Escola Universitaria Politécnica

UNIVERSIDADE DA CORUÑA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A170**

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

AUTOR: **GABRIEL CASAL RODRÍGUEZ**

TUTOR: **JUAN MANUEL RIVAS RODRÍGUEZ**
ESTEBAN JOVE PÉREZ

FECHA: **SEPTIEMBRE DE 2019**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

ÍNDICE

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Listado de figuras

Listado de tablas

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice del documento MEMORIA

1 OBJETO

El objeto de este proyecto es la implementación de un sistema de seguimiento de satélites, basado en la plataforma Arduino, que consistirá en el control de orientación de las antenas conforme a la posición actual del satélite.

Las entradas del sistema serán:

- Los ángulos de azimut y elevación incluidos en una cadena ASCII en formato Easycom, leídos a través de puerto serie. Existirá también la posibilidad de realizar los cálculos de posición del satélite de manera autónoma por el sistema en base a los parámetros keplerianos que definen el movimiento del satélite.
- La realimentación de la posición de las antenas. Esta podrá ser de alguna de las siguientes formas:
  - Con una tensión analógica procedente de potenciómetros acoplados a los ejes de los motores, que debe ser acondicionada para optimizar el convertidor A/D.
  - A través de un encoder digital absoluto.
- Un teclado matricial que nos permitirá introducir numéricamente una consigna para colocar las antenas en la posición deseada.
- Un juego de botones que, además de permitir al usuario desplazarse por los menús que indican el modo de funcionamiento, permitirá aproximar las antenas a una posición determinada de manera manual.

Las salidas del sistema serán:

- La actuación sobre los motores, que podrá ser:
  - De corriente continua: la velocidad será regulada por PWM.
  - De corriente alterna: en este caso, la velocidad vendrá marcada por una tensión analógica para controlar un variador de frecuencia que funcione mediante consigna analógica.
- Un visualizador de cristal liquido, que servirá de interfaz con el usuario, mostrando los distintos modos de funcionamiento del sistema (automático, manual, calibración, etc), además de indicar en tiempo real la posición de las antenas y del satélite en los modos de funcionamiento que proceda.

Este proyecto está enmarcado en una colaboración con el CIFP Ferrolterra, que deberán realizar la construcción de todo el sistema de antenas para lograr la comunicación con la ISS (Internacional Space Station).

Sin embargo, dado que no ha sido posible la realización de este sistema de antenas en el plazo previsto, hemos realizado un pequeño modelo a escala para verificar el correcto funcionamiento del sistema.

Por otra parte, la intención de este proyecto no se ciñe únicamente a la funcionalidad y al objeto anteriormente descrito, sino que tiene también como meta la accesibilidad económica por parte de todo tipo de usuario.

A lo largo de la elaboración del sistema se ha buscado siempre utilizar componentes de bajo coste, empezando por el hardware y software de Arduino, hasta los encoders utilizados para la realimentación de la posición de las antenas, los dispositivos potencialmente más caros de todo el proyecto.

No obstante, no quiere decir que el sistema se encuentre restringido a pequeñas aplicaciones, sino que, con las adaptaciones de potencia pertinentes, podría utilizarse con cualquier tipo de antena.

2 ALCANCE

En este proyecto desarrolla los siguientes puntos:

- Programación del microcontrolador basado en la plataforma Arduino
  - Comunicación con encoders basados en protocolo CANopen
  - Decodificación de datos GPS
  - Decodificación de trama Easycom
  - Elaboración de menús
  - Cálculos keplerianos
- Diseño de un circuito analógico de acondicionamiento de señal
- Diseño de la placa de circuito impreso
- Montaje de componentes sobre la placa de circuito impreso

3 ANTECEDENTES

La Real Academia Española define al satélite artificial [15] como "*Vehículo espacial, tripulado o no, que se coloca en órbita alrededor de la Tierra o de otro astro, y que lleva aparatos apropiados para recoger información y transmitirla*".

El origen de estos *Vehículos espaciales* se remonta a los inicios de la guerra fría entre la URSS y EEUU. Desde entonces, se han desarrollado diversos tipos de satélite con diferentes objetivos (meteorológicos, de telecomunicaciones, de navegación, astronómicos, etc), diferentes órbitas y diferentes altitudes, etc.

3.1. Órbitas satelitales

En la web de la Agencia Espacial Mexicana [17] se proporciona una tabla con las principales órbitas satelitales y sus respectivas aplicaciones, como se muestra en la siguiente figura 3.1, aunque también se utilizan otros tipos distintos, dependiendo del propósito y de las características de cada satélite.

Orbita	Aplicaciones	Características	Ejemplo de misiones
LEO (Low Earth Orbit)	• Observación de la tierra. • Monitoreo del clima • Tecnología • Astronomía • Comunicaciones	Altitud de 300 a 1500 kms.	• CHAMP • SAR-Lupe • BIRD • ROSAT
MEO (Medium Earth Orbit)	• Comunicaciones • Navegación	Altitud de varios miles de kilómetros	• Globalstar • GPS • Galileo
HEO (High Elliptical Orbit)	• Comunicaciones • Astronomía	Altitud de unos cientos de kilómetros hasta 100,000 kilómetros.	• EUTELSAT • ASTRA
GTO (Geoestationary Transfer Orbit)	• Orbita de impulso para lanzamiento de satélites geoestacionarios	Altitud de algunos cientos de kilómetros hasta 35,786 kilómetros	• EUTELSAT • ASTRA
GEO (Geostationary Orbit)	• Comunicaciones	Altitud de 35,786 kilómetros	• EUTELSAT • ASTRA
Puntos de Lagrance	• Astronomía • Investigación	Distancia mayor a un millón de kilómetros	• SOHO • JWST
Orbitas Interplanetarias	• Exploración planetaria	Distancia de varios billones de kilómetros	• Mars Express • Rosetta

Figura 3.1 – *AEM*. Órbitas satelitales [17]

3.2. Bandas de frecuencia de comunicación con satélites

Tal como se explica en la web de la Agencia Espacial Americana [16], en todas las telecomunicaciones es necesaria una frecuencia portadora que separe una comunicación del resto, evitando así posibles interferencias. No obstante, debido a la composición de la atmósfera terrestre, no todo el espectro electromagnético logra atravesarla. A continuación, en la figura 3.2 se muestra un gráfico en el que se representa el espectro electromagnético frente la opacidad atmosférica .

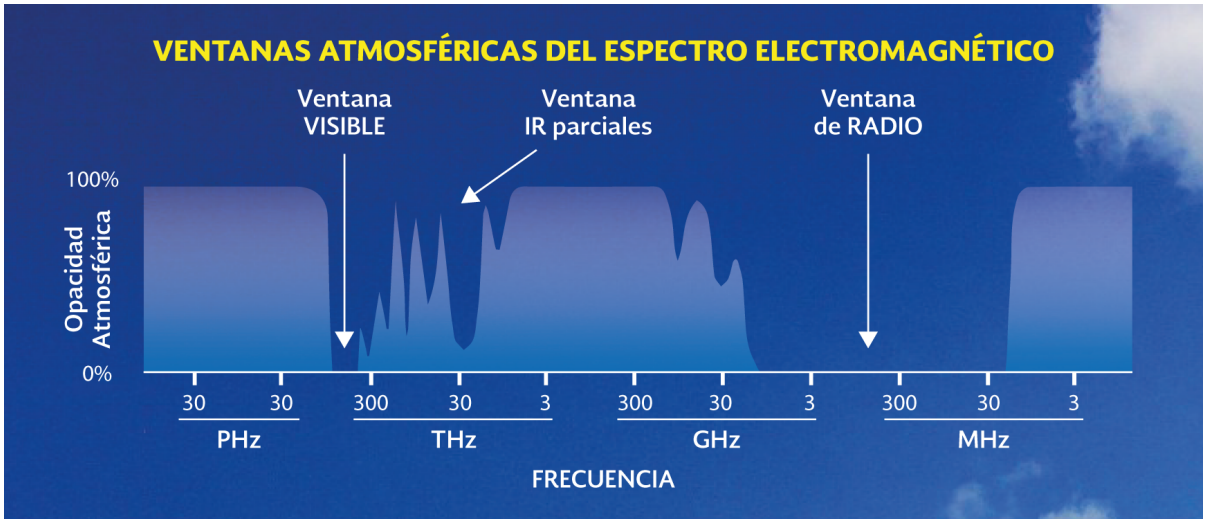


Figura 3.2 – *AEM.* Frecuencias de comunicación satelital [16]

Se puede observar que únicamente existen dos zonas del espectro electromagnético capaces de atravesar la atmósfera terrestre sin atenuación: el espectro visible y una parte del espectro radioeléctrico, aproximadamente de 30MHz a 30 GHz, ventana que se utilizará en las comunicaciones espaciales. El uso de las distintas bandas de este espectro electromagnético satelital está regulado por la Unión Internacional de Telecomunicaciones (UIT).

Consultando también en la web de la Agencia Espacial Europea [47] las principales bandas de frecuencia utilizadas en comunicaciones satelitales se muestran a continuación en la imagen 3.3:

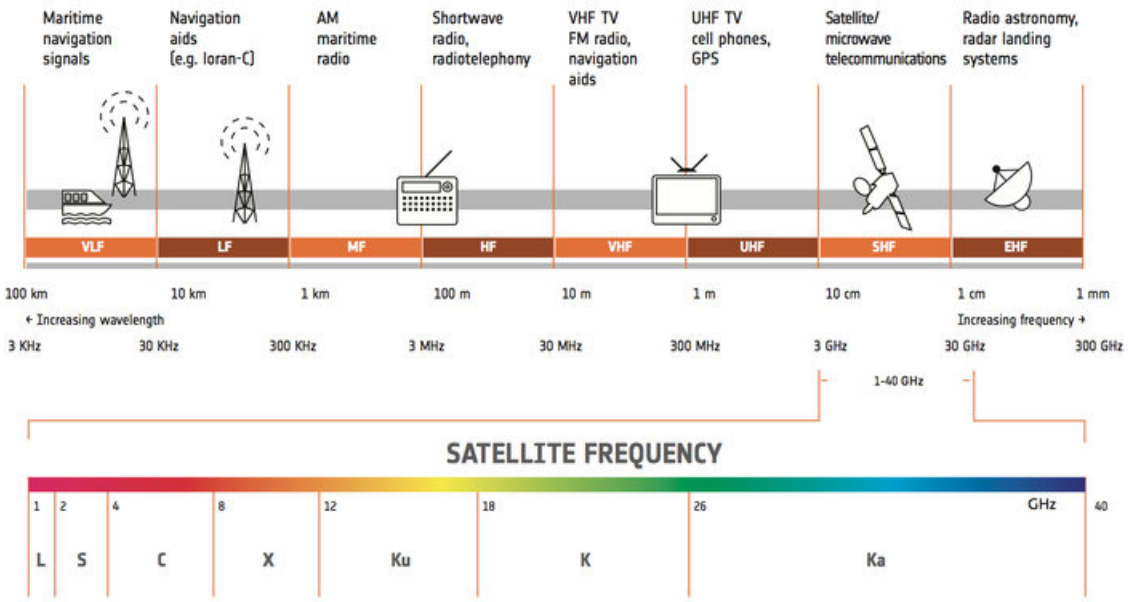


Figura 3.3 – *ESA*. Satellite frequency bands [47]

Sin embargo, otros fragmentos de frecuencia dentro de las bandas UHF y VHF también son utilizados en comunicaciones de satélite, especialmente en el ámbito radioaficionado o incluso militar.

3.3. Tipos de seguimiento

Todos los satélites comparten un requisito común: la necesidad de comunicarse con la Tierra, ya sea con el objetivo de las telecomunicaciones, enviar posiciones GPS, transmitir datos meteorológicos o astronómicos, o cualquier otro tipo de información, dependiendo del tipo de satélite. De esto se puede deducir la necesidad de orientar las antenas de comunicación a la posición orbital del satélite, permitiendo así la transferencia de información.

En el caso de satélites geoestacionarios, es decir, satélites cuya órbita es paralela al ecuador de la Tierra y con la misma velocidad angular, de manera que el satélite se encuentra siempre en la misma posición respecto a la tierra, bastará con orientar la antena a la posición del satélite una única vez, ya que el satélite permanece en una posición fija respecto a la Tierra.

Pero este es un caso muy particular de órbita, en el resto de ellas no es así, por lo que resulta necesario un sistema de ejes móviles capaz de orientarse a la posición del satélite en todo momento de su órbita.

Estos sistemas de ejes son los mismos que los utilizados por los telescopios de observa-

ción, que por lo general se dividen en ecuatoriales y altazimutales:

- Monturas altazimutales:
  Este tipo de monturas están formadas por dos ejes de rotación perpendiculares entre sí, uno perpendicular al plano horizontal (eje de azimut), y el otro perpendicular a este (eje de elevación).

  La rotación del eje de azimut comprende de 0º a 360º, mientras que la del eje de elvación abarca de los 0º a los 90º (hasta 180º en algunos casos).

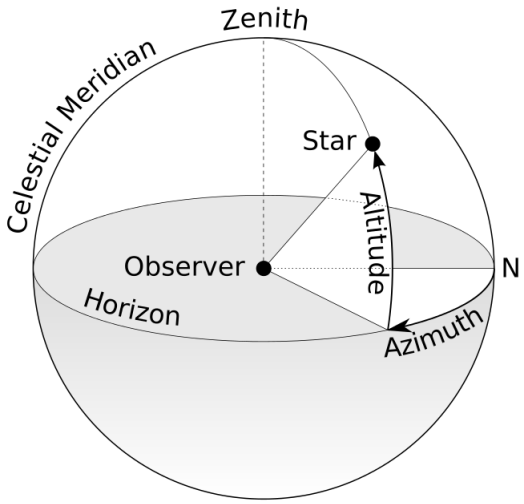


Figura 3.4 – *Wikipedia*. Azimuth-wikipedia
[79]

  A pesar de que es un sistema muy sencillo e intuitivo, dado que este sistema de ejes utiliza coordenadas locales, ambos ejes deberán estar en continuo movimiento para realizar el seguimiento del objeto en cuestión [24, 18].

- Monturas ecuatoriales:
  Este sistema está formado por dos ejes de rotación perpendiculares entre sí, el eje de ascensión recta (eje ecuatorial, paralelo al ecuador celeste) y el eje de declinación. Existe también un tercer eje, el de altitud, que será necesario ajustarlo únicamente en el momento de comenzar un nuevo seguimiento.

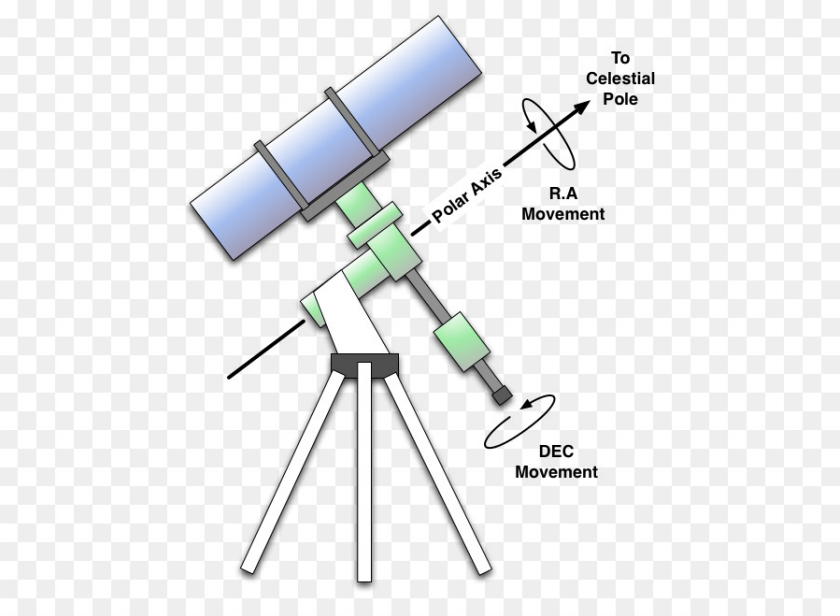


Figura 3.5 – *kisspng*. Equatorial mount Telescope mount Diagram Celestial equator [46]

La montura ecuatorial es más compleja que la anterior ya que, para su correcto funcionamiento, debe de estar perfectamente alineada. Este sistema suele utilizarse mucho en astronomía para observar objetos cuyo movimiento es despreciable respecto al de la tierra debido a la distancia, es decir, para observar objetos aparentemente inmóviles desde la Tierra. De esta manera, lo que se pretende con esta montura es compensar el movimiento de rotación de la Tierra [24, 18].

Tal como se ha mencionado antes, la montura ecuatorial es especialmente útil realizar el para seguimiento de objetos aparentemente inmóviles respecto al movimiento de la Tierra, por lo que no será útil para el seguimiento de satélites artificiales en órbita terrestre, ya que, en este caso, será la Tierra la que permanecerá inmóvil respecto al satélite. Por este motivo y por su sencillez, se ha optado por un sistema de ejes altazimutal para este proyecto.

4 NORMAS Y REFERENCIAS

En este apartado del documento se indica la normativa aplicada a lo largo de la elaboración del proyecto, así como la bibliografía utilizada y cualquier otro tipo de fuente de información consultada.

4.1. Disposiciones legales y normas aplicadas

- Reglamento del Trabajo Fin de Grado de la Escuela Universitaria Politécnica para el Grado en Ingeniería Electrónica Industrial y Automática.
- ISO-690
- UNE-EN 50325-4

4.2. Bibliografía

[1] FERNÁNDEZ CARNERO, José L. y SUÁREZ PERDIGÓN, Antonio, *Televisión y radio analógica y digital*. Santiago de Compostela : Televés, 2004.

[2] MALVINO, Albert Paul, *Principios de electrónica*. México, D.F.: McGraw-Hill, 1986.

[3] PÉREZ GARCÍA, Miguel A.,ÁLVAREZ ANTÓN, Juan C. y CAMPO RODRÍGUEZ, Juan C., *Instrumentación electrónica*. Madrid, D.F.: Thomson, 2004.

[4] URBAN, Sean E. y SEIDELMANN, P. Kenneth, *Explanatory supplement to the Astronomical Almanac*. Mill Valley, Calif. : University Science Books, 2013.

[5] VALLADO, David y MCCLAIN, Wayne D., *Fundamentals of astrodynamics and applications*. Dordrecht : Kluwer Academic Publishers, 2001.

[6] VALLADO, David [en línea], et al.*Revisiting Spacetrack Report #3*. USA: American Institute of Aeronautics and Astronautics, 2006. [Consulta: 4 Septiembre 2019]. Disponible en: https://celestrak.com/publications/AIAA/2006-6753/

4.3. Programas de cálculo

- Arduino IDE
- Notepad++
- CodeBlocks
- Pst Rotator

- KiCad
- AutoCAD (licencia de estudiante)
- Microsoft Excel

4.4. Otras referencias

[7] *3296Y POTENCIOMETRO BOURNS MULTIVUELTA VERTICAL 500K Ohm*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=473231016&cPath=992

[8] *3296Y POTENCIOMETRO BOURNS MULTIVUELTA VERTICAL 5K Ohm*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=473231009&cPath=992

[9] *5-Element-12.5-Ohm-2m-Yagi*. Qsl.net [en línea]. 2019. [Consulta: 15 julio 2019]. Disponible en: https://www.qsl.net/dk7zb/2m-port-SSB/5-Element.htm

[10] *Adjustable Resistor 150W Watt 50 Ohm Ceramic Rotary Rheostat*. En: Amazon.com [en línea], 2019.[Consulta: 10 julio 2019]. Disponible en: https://www.amazon.com/uxcell-Adjustable-Resistor-Ceramic-Rheostat/dp/B008AGVTD8

[11] *Antenna beam width calculator*. En: Satsig.net [en línea]. 2019. [Consulta: 15 julio 2019]. Disponible en: http://www.satsig.net/pointing/antenna-beamwidth-calculator.htm

[12] ARDUINO, *Arduino MEGA2560 R3 (ATmega2560)*. En: aliexpress.com [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://bit.ly/2k2g6ND

[13] ARDUINO, *Arduino Reference*. En: Arduino.cc [en línea], 2019. [Consulta: 4 Septiembre 2019]. Disponible en: https://www.arduino.cc/reference/en/language/variables/data-types/double/

[14] ARDUINO, *Arduino - Compare*. En: Arduino.cc [en línea]. 2019. [Consulta: 21 Julio 2019]. Disponible en: https://www.arduino.cc/en/products.compare

[15] ASALE, RAE, **Satélite**. En: Diccionario de la lengua española [en línea]. Madrid: 2019. [Consulta: 10 julio 2019]. Disponible en: https://dle.rae.es/?id=XKhtxMH

[16] Asociación Espacial Mexicana, *Frecuencias de comunicación satelital*. En: Hacia el espacio [en línea]. México: 2019. [Consulta: 10 julio 2019]. Disponible en: https://haciaelespacio.aem.gob.mx/revistadigital/articul.php?interior=209

[17] Asociación Espacial Mexicana, *Órbitas Satelitales*. En: Hacia el espacio [en línea]. México: 2019. [Consulta: 10 julio 2019]. Disponible en: https://haciaelespacio.aem.gob.mx/revistadigital/articul.php?interior=86

[18] Astronomía Sur, *Monturas de Telescopios - Características, modelos, diseños* [en líena]. 2019. [Consulta: 10 julio 2019]. Disponible en: http://www.astrosurf.com/astronosur/monturas.htm

[19] ATMEL, *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*, En: Microchip.com, [en línea], 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[20] *BC557, 557B - General Purpose Transistor*. En: Farnell.com [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: http://www.farnell.com/datasheets/296678.pdf

[21] BOURNS UNIVERSITY, *Basic Potentiometer Theory*. En: Partnership.bourns.com [en línea]. 2019.[Consulta: 10 julio 2019]. Disponible en: https://partnership.bourns.com/bu/bu_basic_pots.shtml

[22] *BORNA CIRCUITO IMPRESO 2 POLOS PASO 5mm*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999019660&cPath=1175

[23] *BORNA CIRCUITO IMPRESO 3 POLOS PASO 5mm*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999019661&cPath=1175

[24] BRAVO, Roberto, *Tipos de Monturas para un telescopio*. AstroAficion.com [en línea]. 2019. [Consulta: 10 julio 2019]. Disponible en: https://astroaficion.com/2011/03/04/tipos-de-monturas/

[25] *Cable eléctrico 2m, conector M12 4 pin hembra*. En: Ebay.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.amazon.es/sourcing-map-el%C3%A9ctrico-Juntura-Conector/dp/B01MUCYAR0/ref=sr_1_10?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=m12+4+pin&qid=1567441774&s=gateway&sr=8-10

[26] *Cable USB para arduino*. En: aliexpress.com [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://bit.ly/2lGqafz

[27] *CAN Bus Shield - MCP2515-MCP2551*. En: github.com [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://github.com/Seeed-Studio/CAN_BUS_Shield

[28] *CelesTrak: Current NORAD Two-Line Element Sets*. En: Celestrak.com [en línea]. 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://www.celestrak.com/NORAD/elements/

[29] *CIRCUITO INTEGRADO LM324N DIL-14*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=92-324&cPath=875

[30] *CONDENSADOR CERAMICO MULTICAPA R-5 100K pF 50V*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=213004009&cPath=952

[31] *CONDENSADOR ELECTROLITICO RADIAL 100uF 35V 6x11 105º*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=211004046&cPath=968

[32] *CONDENSADOR TANTALO 2.2uF 16V*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=214085025&cPath=954

[33] *CONECTOR DB9 HEMBRA SUB-D CIRCUITO IMPRESO*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999127200&cPath=567

[34] *CONECTOR DB9 MACHO SUB-D ALTA DENSIDAD*. www.cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999127004&cPath=567

[35] COOK, John D., *Latitude doesn't exactly mean what I thought*. En: Johndcook.com [en línea]. 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://www.johndcook.com/blog/2011/09/17/latitude-doesnt-exactly-mean-what-i-thought/

[36] CORONADO RODRÍGUEZ, A y MOUMTADI, F, *Metodología para la obtención del patrón de radiación y prueba de aislamiento en sistemas de comunicaciones vía satélite*. En: Scielo.org.mx [en líena]. México: 2019. [Consulta: 14 julio 2019]. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432006000400005

[37] CSS ELECTRONICS, *CANopen Explained - A Simple Intro*. En: csselectronics.com [en línea]. 2019. [Consulta: 1 Agosto 2019]. Disponible en: https://www.csselectronics.com/screen/page/canopen-tutorial-simple-intro/language/en

[38] *CUBIERTA CONECTOR DB9 SUB-D TORNILLO LARGO*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999019192&cPath=567

[39] *Diagramas de radiacion*. En: Upv.es [en línea]. 2019. [Consulta: 14 julio 2019]. Disponible en: http://www.upv.es/antenas/Tema_1/diagramas_de_radiacion.htm

[40] *DIODO RAPIDO 1N4148 75V 0.3A DO-35*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=71-1N4148&cPath=970

[41] *DIODO RECTIFICADOR 1N4007 1000V 1A DO-41*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=71-1N4007&cPath=970

[42] *DISIPADOR TERMICO TO-220*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999019025&cPath=540

[43] *Display retroiluminado LCD2004 VERDE 20x04 LCM2004A*. En: Ebay.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://bit.ly/2m1wizs

[44] *E6F-AG5C-C 256 2M - ABSOLUTE ROTARY ENCODER, 256CPR, 12-24V*. En: Farnell.com [online]. 2019.[Consulta: 11 julio 2019]. Disponible en: https://es.farnell.com/omron-industrial-automation/e6f-ag5c-c-360-2m/absolute-rotary-encoder-360cpr/dp/2948833?st=E6F-AG5C-C

[45] *E6F-AG5C 360 2M Omron Automation and Safety*. En: Mouser Electronics [en línea]. 2019.[Consulta: 11 julio 2019]. Disponible en: https://www.mouser.es/ProductDetail/Omron-Automation-and-Safety/E6F-AG5C-360-2M?qs=sGAEpiMZZMs%252BIv%2FVy7pdwVU28emBJyzB0mB9n%2F7qxDY%3D

[46] *Equatorial mount Telescope mount Diagram Celestial equator*. En: Kisspng.com [en línea]. 2019. [Consulta: 10 julio 2019]. Disponible en: https://www.kisspng.com/png-equatorial-mount-telescope-mount-diagram-celestial-2138447/

[47] European Space Agency, *Satellite frequency bands*. En: ESA [en línea]. 2018. [Consulta: 10 julio 2019]. Disponible en: https://www.esa.int/Our_Activities/Telecommunications_Integrated_Applications/Satellite_frequency_bands

[48] *GPS, cadenas NMEA y máquinas de estados finitos en Arduino*. En: Fuenteabierta.teubi.co [en línea]. San Salvador: 2015. [Consulta: 2 Septiembre 2019]. Disponible en: http://fuenteabierta.teubi.co/2015/02/gps-cadenas-nmea-y-maquinas-de-estados.html

[49] *GPS - NMEA sentence information*. En: Aprs.gids.nl [en línea]. 2019. [Consulta: 2 Septiembre 2019]. Disponible en: http://aprs.gids.nl/nmea/#rmc

[50] *GY-NEO6MV2 NEO-6M GPS Module Board with Antenna for Arduino and Raspberry Pi*. En: Ebay.es [en línea]. 2019. [Consulta: 20 Julio 2019]. Disponible en: https://bit.ly/2lznfWg

[51] HENRY, Michael F., *The OrbitTools Libraries. NORAD SGP4/SDP4 Implementations in C++ and C#* . En: zeptomoby.com [en línea], 2019. [Consulta: 4 Septiembre 2019]. Disponible en: http://www.zeptomoby.com/satellites/

[52] KELSO, T.S., *Orbital Coordinate Systems, Part I*. En: Satellite Times [en línea]. 1995. [Consulta: 1 Septiembre 2019]. Disponible en:https://celestrak.com/columns/v02n01/

[53] KELSO, T.S., *Orbital Coordinate Systems, Part 2*. En: Satellite Times [en línea], 1995. [Consulta: 1 Septiembre 2019]. Disponible en: https://celestrak.com/columns/v02n02/

[54] KELSO, T.S.,*CelesTrak: Orbital Coordinate Systems, Part 3*. En: Satellite Times [en línea], 1995. [Consulta: 1 Septiembre 2019]. Disponible en: https://celestrak.com/columns/v02n03/

[55] *KIT TIRAS DE PINES COMPATIBLES CON ARDUINO REV3* En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999334116&cPath=1342

[56] *LED AMARILLO 5mm*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=84-5A&cPath=1158

[57] LLAMAS, Luis, *Localización GPS con Arduino y los módulos GPS NEO-6*. En: Luisllamas.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/

[58] *MCP2515 CAN BUS Shield Controller SPI Communication Speed High Arduino*. En: Bay.com [en línea]. 2019. [Consulta 20 Agosto 2019]. Disponible en: https://bit.ly/2k7h4Za

[59] MICROCHIP, *MCP2515*. En: Microchip.com [en línea]. 2019. [Consulta: 2 Septiembre 2019]. Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

[60] MICROCHIP, *MCP2551*. En: Microchip.com [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: http://ww1.microchip.com/downloads/en/devicedoc/21667e.pdf

[61] *Modelo OSI, niveles de modelo, comunicación de redes - Redes informáticas 2016 JFC*. En: Sites.google.com [en línea]. 2019. [Consulta: 1 Agosto 2019]. Disponible en: https://sites.google.com/site/redesinformaticas2016jfc/transmision-de-datos-en-las-redes/modelo-osi-niveles-de-modelo-comunicacion-de-redes

[62] NASA,*Human Space Flight (HSF) - Realtime Data*. En: Spaceflight.nasa.gov [en línea]. 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html

[63] *Orbital elements*. En: Wikipedia: The Free Encyclopedia [en línea], 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://en.wikipedia.org/wiki/Orbital_elements

[64] *PULSADOR DE MEMBRANA B3F-4050*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=484351001&cPath=1014

[65] *¿Qué es y para qué sirve una Antena?*. En: Artchist.blogspot.com [en líena]. 2019. [Consulta: 14 julio 2019]. Disponible en: http://artchist.blogspot.com/2019/04/que-es-y-para-que-sirve-una-antena.html

[66] *REGULADOR FIJO 7805 +5V 1A TO-220*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=86-7805CT&cPath=980

[67] *Rele 5v 10A SPDT - SRD-05VDC-SL-C*. En: Ebay.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en:https://www.ebay.es/itm/Rele-5v-10A-SPDT-SRD-05VDC-SL-C-Arduino-Electronica-DIY/222667235276?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2060353.m2749.l2649

[68] *RESISTENCIA CARBON 1/4W 5 %*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=512310001&cPath=921

[69] *Solar/Sidereal Days*. En: Celestialnorth.org [en línea]. 2019. [Consulta: 1 Septiembre 2019]. Disponible en: http://www.celestialnorth.org/FAQtoids/dazed_about_days_(solar_and_sidereal).htm

[70] *T-series manual 11551*. En: TWK-ELEKTRONIK GmbH [en línea]. Alemania: 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.twk.de/en/support-and-service/documentation/9027/t-series-manual-11551?number=SW10217

[71] *Teclado matricial 4x4*. En: Ebay.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://bit.ly/2k2yHsW

[72] TEXAS INSTRUMENTS, *lm324-n*. En: www.ti.com [en línea], 2019. [Consulta: 1 Septiembre 2019]. Disponible en: http://www.ti.com/lit/ds/symlink/lm324-n.pdf

[73] TEXAS INSTRUMENTS, *LM340,LM340A and LM7805 Family Wide VIN 1.5-A Fixed VoltageRegulators*. En: Texas Instrumenrs [en línea], 2019. [Consulta: 3 Septiembre 2019]. Disponible en: http://www.ti.com/lit/ds/symlink/lm7800.pdf

[74] *TRANSISTOR BC557 PNP 50V 0.1A 0.5W 150MHz TO-92*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=70-BC557&cPath=881

[75] *TWK-Elektronik TBN 50-SA 2048R C2 S N14 Absolute Encoder*. En: Ebay.es [en línea]. 2019. [Consulta: 12 julio 2019]. Disponible en: https://www.ebay.es/itm/TWK-Elektronik-TBN-50-SA-2048R-C2-S-N14-Absolute-Encoder/382299568664?hash=item5902d47a18:g:UykAAOSwcLxbcc6g

[76] VALLADO, David, *Astrodynamics Software*. En: Celestrak.com [en línea], 2018. [Consulta: 5 Septiembre 2019]. Disponible en: https://www.celestrak.com/software/vallado-sw.php

[77] WIKIPEDIA, *Anomalía media*. En: Wikipedia: The Free Encyclopedia [en línea], 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://es.wikipedia.org/wiki/Anomal

[78] WIKIPEDIA, *Antena*. En: Wikipedia: The Free Encyclopedia [en línea]. 2019. [Consulta: 14 julio 2019]. Disponible en: https://es.wikipedia.org/wiki/Antena#Diagrama_de_radiaci%C3%B3n

[79] WIKIPEDIA, *Azimuth*. En: Wikipedia: The Free Encyclopedia [en línea]. 2019. [Consulta: 10 julio 2019]. Disponible en: https://en.wikipedia.org/wiki/Azimuth

[80] WIKIPEDIA, *CANopen*. En: Wikipedia: The Free Encyclopedia [en línea]. 2019. [Consulta: 1 Agosto 2019]. Disponible en: https://en.wikipedia.org/wiki/CANopen

[81] WIKIPEDIA, *Nutación*. En: Wikipedia: The Free Encyclopedia [en línea]. 2019. [Consulta: 1 Septiembre 2019]. Disponible en: https://es.wikipedia.org/wiki/Nutaci%C3%B3n

[82] *ZENER BZX85C 5V6 1W*. En: Cetronic.es [en línea], 2019. [Consulta: 2 Septiembre 2019]. Disponible en: https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=75-Z5V6&cPath=973

[83] *Zener diode*. En: Blog.fossasia.org [en línea]. 2019. [Consulta: 13 julio 2019]. Disponible en: https://blog.fossasia.org/tag/zener-diode/

5 DEFINICIONES Y ABREVIATURAS

- **A/D:** Analógico/Digital
- **AEM** Agencia Espacial Mexicana
- **AFSPC:** Air Force Space Command
- **ASCII:** American Standard Code for Information Interchange
- **CANopen:** Protocolo de comunicaciones industriales
- **ECI:** Earth Centered Inertial
- **ESA** European Space Agency
- **GPS:** Global Positioning System
- **HMI:** Human-Machine Interface
- **IDE:** Integrated Development Environment
- **ISS:** International Space Station
- **JD:** Julian Day
- **NASA:** National Aeronautics and Space Administration
- **NEZ:** Norte Este Zenit
- **NMEA:** National Marine Electronics Association
- **NORAD:** North American Aerospace Defense Command
- **LCD:** Liquid Crystal Display
- **LEO:** Low Earth Orbit
- **PCB:** Printed Circuit Board
- **PDO:** Process Data Object
- **Punto Aries:** punto imaginario en el espacio sobre la la línea de intersección del plano ecuatorial y el plano de la órbita de la Tierra alrededor del Sol, o eclíptica en el equinoccio vernal
- **PWM:** Pulse-Width Modulation
- **SEZ:** Sur Este Zenit

- **Shield:** placa de circuito que se acopla sobre la placa de Arduino, ampliando sus funcionalidades
- **SYNC:** Synchronization object
- **TEME:** True Equator Mean Equinox
- **TLE:** Two Line Elements
- **UIT:** Unión Internacional de Telecomunicaciones
- **UT:** Universal Time
- **UT1:** Universal Time 1
- **UTC:** Universal Time Coordinated
- **VSAT:** Very Small Aperture Terminal

6 REQUISITOS DE DISEÑO

Para implementar correctamente este proyecto deben cumplirse los siguientes requisitos de diseño:

- Leer y decodificar la trama ASCII recibida por puerto serie en formato Easycomm para obtener los ángulos de azimut y elevación a los que debemos direccionar las antenas.
- Cálculo de estos ángulos de azimut y elevación de manera autónoma por el sistema en base a los parámetros keplerianos del satélite.
- Lectura de la señal GPS para obtener el instante y el lugar de observación de manera exacta, parámetros necesarios para los cálculos keplerianos.
- Selección del modo de funcionamiento del sistema a través de una interfaz con el usuario, formada por una pantalla LCD y un sistema de menús, a través de los cuales el usuario podrá moverse mediante una serie de pulsadores.
- En los modos de funcionamiento anteriores deben incluirse
  - Lectura de la posición de apuntamiento por puerto serie.
  - Cálculo de manera autónoma de posición de apuntamiento.
  - Setup inicial que permita configurar determinados aspectos del sistema.
  - Calibración de ambos motores.
- Lectura de la posición de las antenas. Esta podrá ser:
  - Mediante dos potenciómetros analógicos multivuelta solidarios al eje de los motores. La señal procedente de dichos potenciómetros deberá ser adecuadamente acondicionada antes de ser conectada a la placa Arduino.
  - Mediante dos encoders absolutos con protocolo de comunicación industrial CANopen.
- El sistema dispondrá también de un modo de funcionamiento que proporcione la opción de mover ambas antenas manualente:
  - Establecimiento de un ángulo determinado por consigna introducida a través de un teclado matricial.
  - De forma completamente manual mediante la pulsación de teclas.

7 ANÁLISIS DE LAS SOLUCIONES

En este apartado se expondrán las diferentes soluciones barajadas para cada uno de los aspectos del proyecto que se describen en los requisitos de diseño del capítulo 6.

7.1. Posición de satélites

Como es obvio, es necesario conocer la posición del satélite con respecto a la posición del observador en todo momento para poder realizar el seguimiento.

Una opción para obtener dicha posición es la de utilizar un software externo al microcontrolador. La otra vía posible es la de realizar los cálculos software de posición de manera autónoma a través del microcontrolador.

7.1.1. Software externo

En esta opción se dispondrá de un software externo al microcontrolador que comunique en todo momento a este último los parámetros necesarios de posición del satélite a través de puerto serie en forma de una trama de caracteres ASCII.

El cálculo de la posición satelital requiere de diversos parámetros de corrección y actualización de manera frecuente (excentricidad de la órbita, deriva de la órbita terrestre, etc). La principal ventaja de utilizar este software es que, además de evitar al microcontrolador realizar todos los cálculos necesarios, lo que requiere una cierta potencia de cálculo, se mantiene siempre actualizado y con las correcciones pertinentes aplicadas.

Existen diversos formatos de trama para transmitir los parámetros de la posición del satélite, como el GS232 o el Easycom. Este último será el que se empleará en nuestro sistema, ya que es un formato que ha sido utilizado desde los años 80, y es soportado por casi todo el software de seguimiento de satélites.

En cuanto al programa que proporcionará la posición del satélite, a pesar de que existen diversas opciones de software gratuito, se ha seleccionado el PstRotator, un software actual y completo, cuyo precio es meramente simbólico. No obstante, sería perfectamente válido cuaquier otro software que proporcione la posición del satélite por puerto serie en formato Easycom.

7.1.2. Cálculo autónomo

En esta opción, el cálculo de la posición del satélite con respecto al observador en la Tierra se realizará de manera autónoma por el microcontrolador.

La forma más común de realizar dicho cálculo es a través de los parámetros keplerianos del satélite, disponibles en bases de datos en línea, como la del NORAD [28].

Estos datos suelen presentarse en formato TLE (Two Line Elements), de manera que será necesario decodificarlos y hacer los cálculos correspondientes en base a los mismos para

obtener la posición del satélite.

En contraste con lo mencionado en el apartado anterior, esta opción elimina la dependencia del PC. No obstante, sí es cierto que los cálculos a realizar presentan cierta exigencia en lo que a potencia de procesado se refiere.

Además, tal como se explica en el apartado anterior, estos datos TLE son corregidos y actualizados frecuentemente, por lo que sería conveniente la posibilidad de actualizarlos en el microcontrolador, ya sea a través de una memoria externa, a través de puerto serie o a través una conexión directa con la base de datos en la Web. Concretamente para este proyecto se ha optado por la grabación de los datos en la memoria EEPROM del microcontrolador a través de puerto serie.

Cabe destacar que, para poder realizar los cálculos keplerianos, es necesario conocer con exactitud la localización del observador en la Tierra, así como la fecha y la hora. Para esto utilizaremos un receptor GPS que nos proporcione dichos datos. La solución más común para receptores GPS en microcontroladores es módulo receptor GPS GY-NEO6MV2, con un precio aproximado de 5€.

Figura 7.1 – *ebay*. GY-NEO6MV2 NEO-6M GPS Module Board with Antenna [50]

7.2. Motores

Normalmente, para este tipo de sistemas se utilizan motores de corriente continua, ya que, por norma general, son más sencillos de controlar, aunque en sistemas de antenas de grandes dimensiones, que necesiten motores de elevada potencia, podrían requerirse motores de corriente alterna.

Dado que el objetivo de este proyecto es la escalabilidad del mismo, se aportará una solución para las dos variedades de motor. Por lo tanto, se proporciona, por un lado, una señal PWM y, por otra parte, esa misma señal PWM filtrada para poder ser utilizada como señal de control del variador de frecuencia, , como se describe en la sección 7.5.

Esta última señal será una tensión continua de 0V a 5V. En caso de que la señal de control del variador de frecuencia necesite otro rango de voltaje, se delega la adaptación de tensiones al usuario final del sistema.

Además, se añadetambién un circuito de relés de contactos libres de potencial, para poder determinar el sentido de giro de cada motor y actuar sobre el freno del mismo. Serán disponen, por tanto, 3 relés por cada motor: un relé para cada sentido de giro más un relé adicional para activar y desactivar el freno del motor (en caso de tenerlo).

7.3. Realimentación de la posición de los motores

A la hora de realizar la realimentación de la posición de los motores distinguimos dos opciones:

- Potenciómetros
- Encoders

7.3.1. Realimentación mediante potenciómetros

Una posible manera de obtener la posición de un motor es a través de la lectura analógica de la tensión en el cursor de un potenciómetro.

Se incorpora a este proyecto la posibilidad de utilizar este tipo de realimentación para dotar al mismo de retrocompatibilidad, ya que, los equipos antiguos utilizados para este tipo de aplicaciones, emplean normalmente este modo de realimentación.

Para el montaje de la maqueta del proyecto se utilizarán motores que incluyen sus propios potenciómetros, por lo que no será necesario seleccionar un tipo concreto.

Sin embargo, en esta sección se realiza un breve análisis de las distintas opciones, y cuál de ella es la que mejor se adapta a las necesidades del proyecto.

Existe una amplia variedad de potenciómetros disponibles, de mayor o menor tolerancia resistiva, respuestas más o menos lineales, distintas potencias soportadas, ruido generado, etc. Pero antes de analizar este tipo de características eléctricas del producto, describiremos brevemente una serie de características de carácter más general.

La primera clasificación a la que se debe atender divide a los potenciómetros en rotativos o de desplazamiento lineal, siendo la primera opción la necesaria para este proyecto.

Debe tenerse en cuenta también la composición del elemento resistivo por el que se desplaza el cursor, como se muestra a continuación [3, 21]:

- No bobinados:
  - Plástico conductivo
  - Cermet
  - Carbón
  - Híbridos

- De hilo bobinado: en este caso, el elemento resistivo, tal como sugiere su propio nombre, es un hilo arrollado en torno a un soporte. Su respuesta ante cambios de temperatura es excelente, además de ofrecer buenos rangos de potencia, larga vida útil, y poca distorsión de la señal por ruido.

  Por otra parte, al tratarse de un hilo bobinado, el cursor no podrá desplazarse de manera completamente continua sobre él, ya que el bobinado presenta una serie de saltos entre espira y espira. Es decir, el funcionamiento es similar al de un codificador digital de tantos bits como saltos de espira haya en el potenciómetro. Esto supone una desventaja en caso de utilizar señales de alterna debido a las pequeñas capacidades e inductancias que pueden aparecer debidas al elemento bobinado. No obstante, en este proyecto, el potenciómetro soportará únicamente corriente continua, eliminando así este último problema [3].

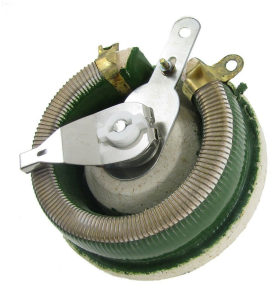


Figura 7.2 – *Amazon*. Wirewound potenciometer [10]

Existe una variedad de potenciómetros que se caracterizan por su buena respuesta en aspectos como linealidad, resolución, ruido y vida rotacional útil. Son los potenciómetros de precisión, que suelen del tipo de hilo bobinado y, en muchos casos, poseen mayores dimensiones, siendo esto una ventaja en lo que a nuestro proyecto se refiere. Por otro lado, consecuentemente a esta serie de características descritas, sus precios suelen ser también más elevados.

Se pueden encontrar también potenciómetros de precisión multivuelta, que ofrecen una mayor precisión ya que, con la multiplicación o desmultiplicación adecuada, puede conseguirse que una vuelta completa de la antena de 0º a 360º se corresponda con el número de vueltas que permite el potenciómetro.

Puede concluirse entonces que los potenciómetros que mejor se adaptan al proyecto son estos últimos, potenciómetros de precisión bobinados y, también, multivuelta.

7.3.2. Realimentación mediante encoders

Otra manera de obtener la posición del motor es a través de encoders rotativos. En contraste con el modo de realimentación por nivel de tensión analógica, los encoder son la opción potencialmente más precisa, aunque también más cara.

Estos dispositivos traducen la posición angular del eje de rotación a un código de pulsos digitales, dependiendo de la resolución.

Ahora bien, la manera de proporcionar la posición angular puede ser absoluta o relativa, según sean encoders absolutos o incrementales, respectivamente.

Los primeros, como su nombre indica, proporcionan la posición de manera absoluta, cada posición del eje de rotación se corresponde con un código binario determinado.

En contraste, los incrementales no proporcionan una posición angular absoluta, sino que generan pulsos binarios con cada unidad de desplazamiento. Esta unidad de desplazamiento dependerá de la resolución del encoder. Esto significa que, para identificar la posición angular, será necesario establecer un punto de referencia inicial que se tomará como posición cero, a partir de la cual la posición se irá incrementando con cada pulso del encoder.

Teniendo en cuenta lo anterior, y sabiendo que este sistema de seguimiento de satélites es un sistema considerablemente lento, ya que, incluso los satélites más rápidos de órbita más baja, tardan más de una hora en completar una vuelta completa en torno a la Tierra (de la cual sólo podremos seguir durante una porción de la misma), se deduce también que el movimiento de los motores debe de ser lento.

Esto se traduce en que, en caso de un error en la lectura de la posición durante el seguimiento, sería necesario calibrar de nuevo el sistema enviando los motores hasta su posición de paso por cero y, dado que se trata de un sistema con movimiento de motores lento, supondría un proceso que perdería mucho tiempo. Es por esto que un encoder absoluto se adapta mejor a las necesidades de este proyecto.

Sin embargo, al consultar en distribuidores como Farnell o Mouser, se observa que los encoders industriales, con cierta robustez mecánica, ascienden a precios excesivamente elevados para la naturaleza de nuestro proyecto, como se ha comentado en el alcance del proyecto del capítulo 2.

Un buen ejemplo de esto es el encoder absoluto del fabricante Omrom E6F-AG5C-C [45, 44], un encoder rotativo absoluto de código gray con un precio de 976,0€ en Farnell y 580,87€ en Mouser.

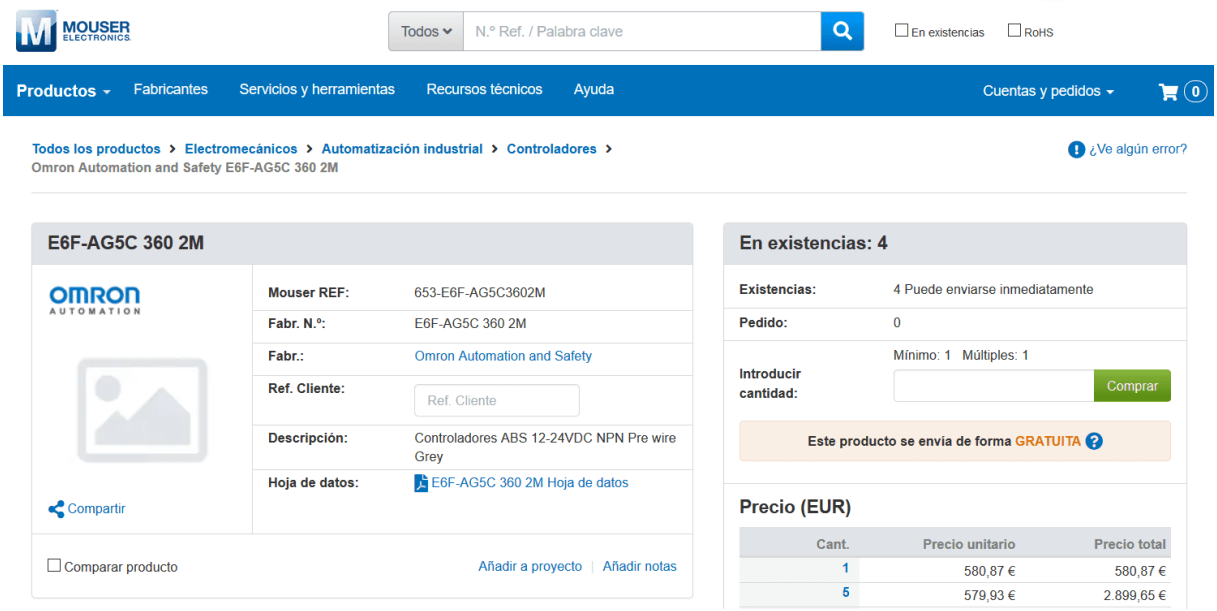


Figura 7.3 – Mouser[45]

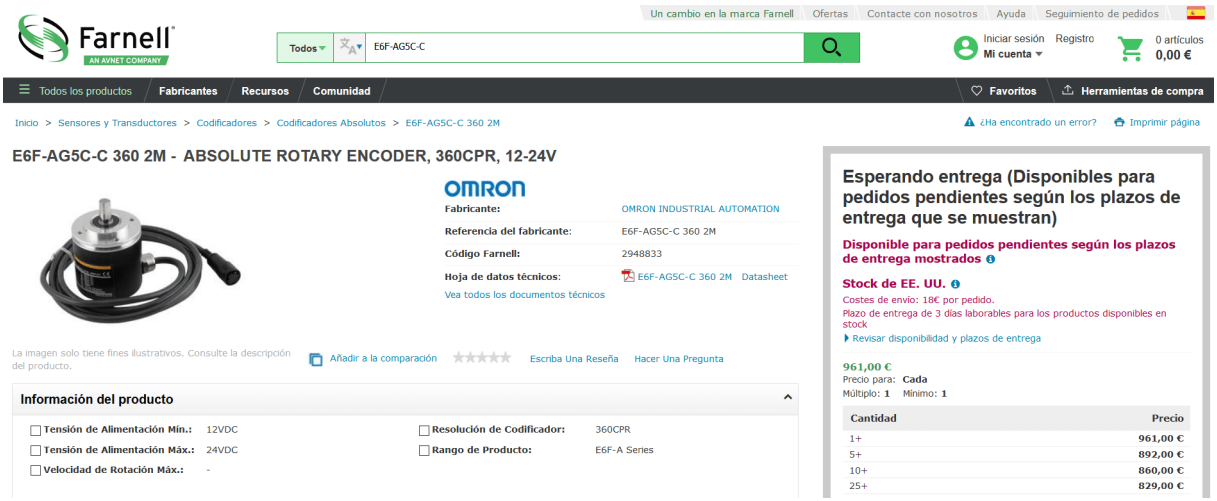


Figura 7.4 – Farnell [44]

Su resolución es de 9 bits, $2^9 = 512$, sin embargo, en la hoja de características se especifica que la resolución es únicamente de 360, es decir, presenta una precisión de 1º.

Este mismo modelo se ha conseguido de segunda mano para el proyecto por un precio bastante inferior al anteriormente indicado, pero excesivamente elevado de todas formas. Además, se ha verificado que dicho encoder, a pesar del elevado precio, llegó defectuoso.

En la búsqueda de este encoder de segunda mano se ha encontrado otro modelo de precio considerablemente más asequible que el anterior, además de ofrecer mayor resolución. Se trata del encoder absoluto TBN 50-SA 2048R C2 S N14 del fabricante alemán TWK-Electronik que se ha conseguido a través de eBay por un precio de 20,99€ [75].

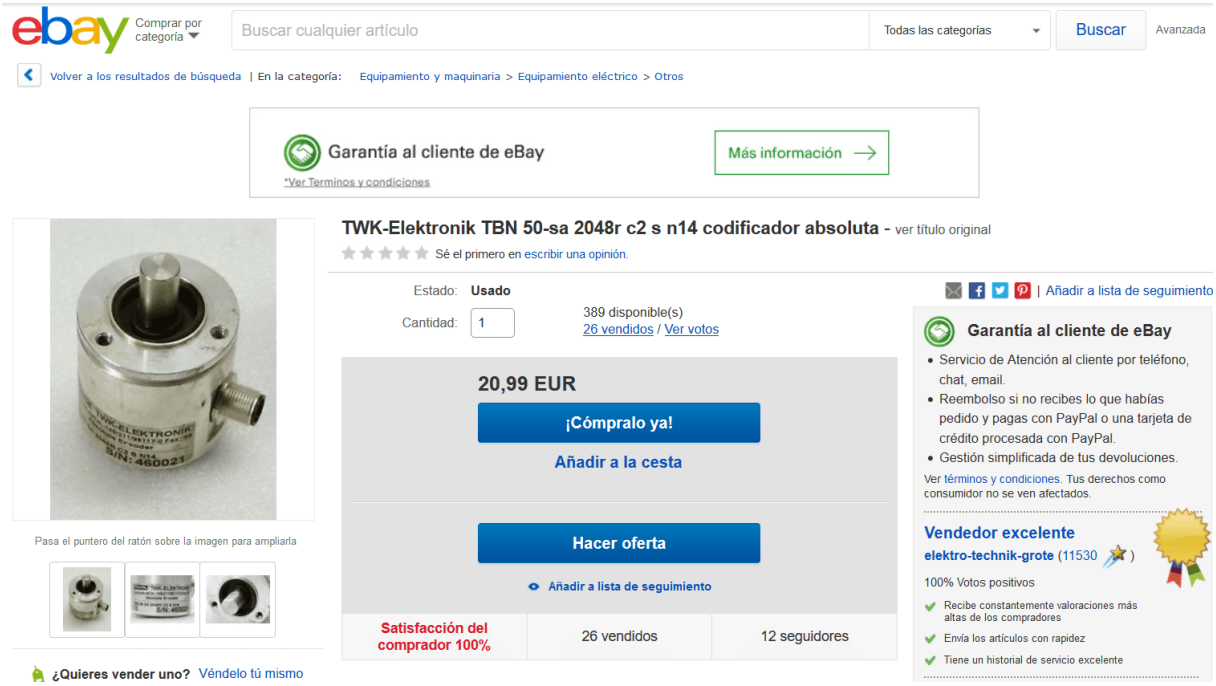


Figura 7.5 – Encoder TWK [75]

Este encoder posee 12 bits de resolución, lo que significa que tendremos una precisión de $2^{12} = 4096$, es decir, 0,09º de precisión. Está basado en el protocolo de comunicación CANopen, por lo tanto su lectura no será trivial, y requerirá un estudio del protocolo para lograr la comunicación con el microcontrolador.

Dado su bajo precio y su alta precisión, se convierte en la mejor opción para implementar en este proyecto.

Por otro lado, puesto que este encoder está basado en el protocolo de comunicación CANopen, es necesaria una interfaz entre el bus de comunicación de CANopen y el microcontrolador Arduino. Se utilizará el Shield de comunicaciones de bus CAN para Arduino.

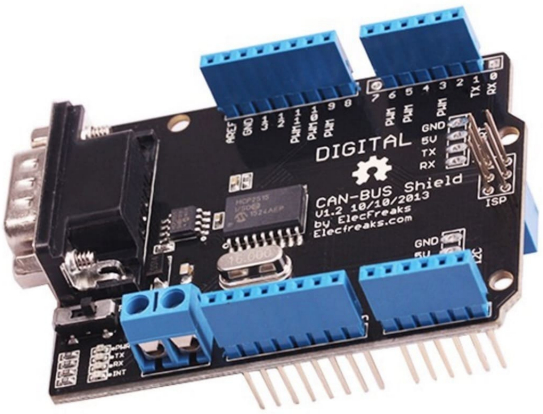


Figura 7.6 – *ebay*. MCP2515 CAN Shield Controller [58]

7.4. Acondicionamiento analógico de los potenciómetros

Para poder leer adecuadamente la posición de las antenas a través de potenciómetros, es necesario diseñar un circuito analógico de acondicionamiento de señal que optimice el funcionamiento del conversor A/D del microcontrolador, de manera que la tensión de entrada de este último se encuentre siempre en el rango óptimo para la conversión analógico/digital, independientemente del rango de voltaje de entrada al circuito de acondicionamiento (siempre que este se encuentre dentro de las limitaciones del circuito).

En la sección 8.3.1 se ofrece una breve descripción del funcionamiento de dicho circuito. Por otra parte, el cálculo de todos los componentes del circuito analógico se realiza en el capítulo 11 (anexo de cálculos).

7.5. Control de posición y velocidad

El propósito de la realimentación de la posición de las antenas no es otro que poder controlar el movimiento de los motores de manera que apunten en todo momento a la posición del satélite.

Este control de velocidad se basará en un arranque y una frenada de carácter lento, ya que los cambios bruscos de potencia son perjudiciales para los motores eléctricos y para la propia estructura del sistema, especialmente si se trata de motores de alta potencia, como sería el caso de un radiotelescopio o de antenas de grandes dimensiones. Además, estos cambios bruscos de potencia provocarían vibraciones en antenas como las que se pretenden utilizar.

Se realizará el cálculo por software de una señal PWM en función de la posición actual y la posición de consigna.

7.6. Interfaz con el usuario

De nuevo, tal como se especifica en los requisitos de diseño del capítulo 6, el sistema debe contar con una interfaz con el usuario que permita a este último pueda visualizar los parámetros necesarios del sistema, desplazarse por los distintos menús, etc.

Se dispondrá de:

- Un display LCD (20x4)
- 5 pulsadores, que servirán tanto para desplazarse por los menús como para el movimiento manual de los motores.
- Un teclado matricial de 4x4, necesario para introducir manualmente consignas de posición para los motores.

7.7. Selección del microcontrolador

Los requisitos de diseño del capítulo 6 restringen la selección del microcontrolador a la plataforma Arduino por su fácil acceso para todo tipo de usuarios. En la tabla de la figura 7.7 se muestran las distintas opciones que ofrece la plataforma Arduino.

En primer lugar se realiza un recuento de las entradas y salidas necesarias, tanto analógicas como digitales:

- Circuito de acondicionamiento analógico: 2 entradas analógicas para los potenciómetros de posición de los motores.
- Placa de relés: 6 salidas digitales para el control de los relés (una salida digital por cada relé).
- Control de velocidad: dos pines digitales para PWM, una por cada motor.
- Interface con el usuario:
  - Teclado: 8 pines digitales para el teclado matricial de 4x4.
  - Botonera: 5 pines digitales, uno por cada pulsador.
  - Display LCD: 7 pines digitales, utilizando el modo de transmisión de 4 hilos para utilizar el menor número de pines.
- Comunicación con encoders: Se ha indicado en la sección 7.3 que se utilizarán encoders basados el protocolo de comunicación CANopen, por lo que será necesario una interfaz entre el bus de comunicaciones CAN y el microcontrolador Arduino. Esta comunicación se realizará a través del bus SPI, utilizando así 4 pines digitales: MISO, MOSI, SCK y un pin adicional, SS, que funcionará como chip select del Shield de comunicaciones CAN para Arduino.
- Receptor GPS: La comunicación entre el receptor GPS y el microcontrolador Arduino se realiza a través de puerto serie, por lo que se requerirán dos pines digitales (TX y TR).

En la siguiente tabla se muestra el recuento total de salidas y entradas necesarias:

Entradas/Salidas

Bloque	Digitales	Analógicas
Acondicionamiento analógico	-	2
Placa de relés	6	-
Control de velocidad	2 (PWM)	-
CANopen	5	-
Receptor GPS	2	-
Teclado	8	-
Botonera	5	-
Display LCD	7	-
Total	35	2

Tabla 7.1 – Recuento de pines utilizados

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V/ 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8 MHz	4/0	9/4	1	2	32	-	-
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	1	2.5	32	Micro	-
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V/ 5V	48MHz	7/1	8/4	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	1	2	32	-	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	-	96	512	2 Micro	4
Esplora	ATmega32U4	5 V / 7-12 V	16 MHz	-	-	1	2.5	32	Micro	-
Ethernet	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/4	1	2	32	Regular	-
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Mini	ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	1	2	32	-	-
Nano	ATmega168 ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	0.512 1	1 2	16 32	Mini	1
Yún	ATmega32U4 AR9331 Linux	5 V	16 MHz 400MHz	12/0	20/7	1	2.5 16MB	32 64MB	Micro	1
Arduino Robot	ATmega32u4	5 V	16 MHz	6/0	20/6	1 KB (ATmega32u4)/ 512 Kbit (I2C)	2.5 KB (ATmega32u4)	32 KB (ATmega32u4) of which 4 KB used by bootloader	1	1
MKRZero	SAMD21 Cortex-M0+ 32bit low power ARM MCU	3.3 V	48 MHz	7 (ADC 8/10/12 bit)/1 (DAC 10 bit)	22/12	No	32 KB	256 KB	1	1

Figura 7.7 – *Arduino.* Compare board specs [14]

Consultando la tabla de la figura 7.7, los únicas placas que presentan un número de entradas y salidas digitales suficientes para este proyecto son la MEGA 2560, la MEGA ADK y la DUE. En el proyecto se utilizará la MEGA 2560, ya que se disponía inicialmente de ella.

7.8. Requisitos precisión del sistema

Uno de los aspectos fundamentales a tener en cuenta es la exigencia del sistema en cuanto a precisión.

Las características de una antena suelen venir definidas y representadas gráficamente en lo que se conoce como diagramas de radiación. Estos representan, normalmente en función de la dirección expresada en coordenadas angulares, la densidad de potencia radiada o absorbida por la antena. Se expresa en dB en relación a la dirección de máxima potencia radiada o recibida. Aunque este diagrama de radiación representa la densidad de radiación en el espacio tridimensional, suele representarse en gráficos de dos dimensiones [78, 36, 39].

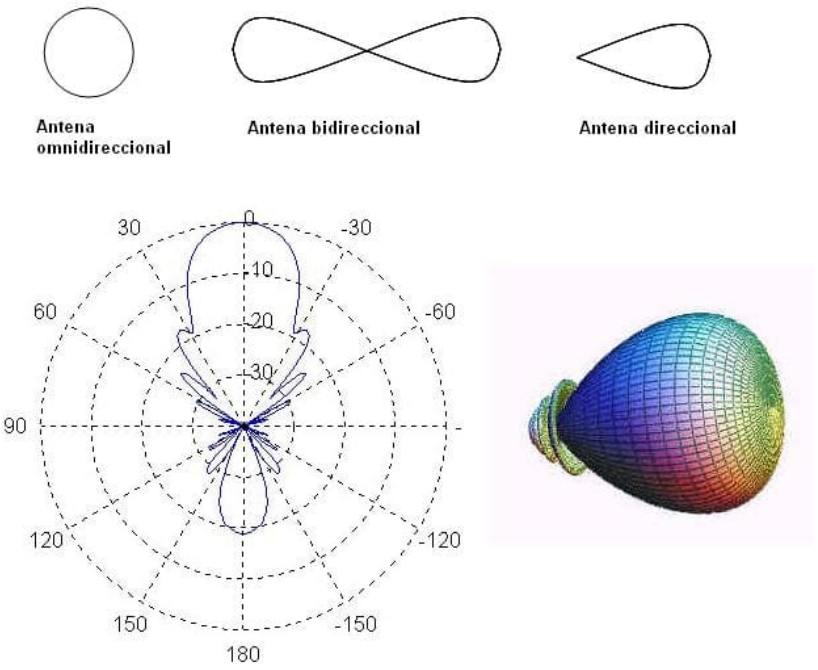


Figura 7.8 – *wikipedia*. Diagrama de radiación [78]

Así pues, una de las características fundamentales de las antenas es su directividad. Tal como explica Televés [1], la directividad de una antena es "*la capacidad de concentrar la potencia radiada en una dirección determinada del espacio*"para el caso de emisión de señal y, para el caso de recepción, "*capacidad de absorber la potencia incidente en esa dirección*".

Existen dos parámetros fundamentales que determinan la directividad de cada antena, el ancho de haz y la relación delante/atrás.

- **Relación delante/atrás**:

Televés [1] define la relación delante/atrás como "*la relación entre la ganancia de la antena en el punto de máxima radiación, y la ganancia de la antena en cualquier otra dirección comprendida entre 90º y 270º respecto a ella*".

- **Ancho de haz**:

  De nuevo, Televés [1] define ancho de haz como "*el ángulo formado por los dos ejes imaginarios de unión de la antena con los puntos donde la ganancia ha caído 3dB respecto del punto de máxima potencia*", es decir, los puntos donde la potencia es la mitad con respecto al punto de máxima potencia. Por tanto, este ángulo que define el ancho de haz determinará la precisión necesaria que deberá tener nuestro sistema.

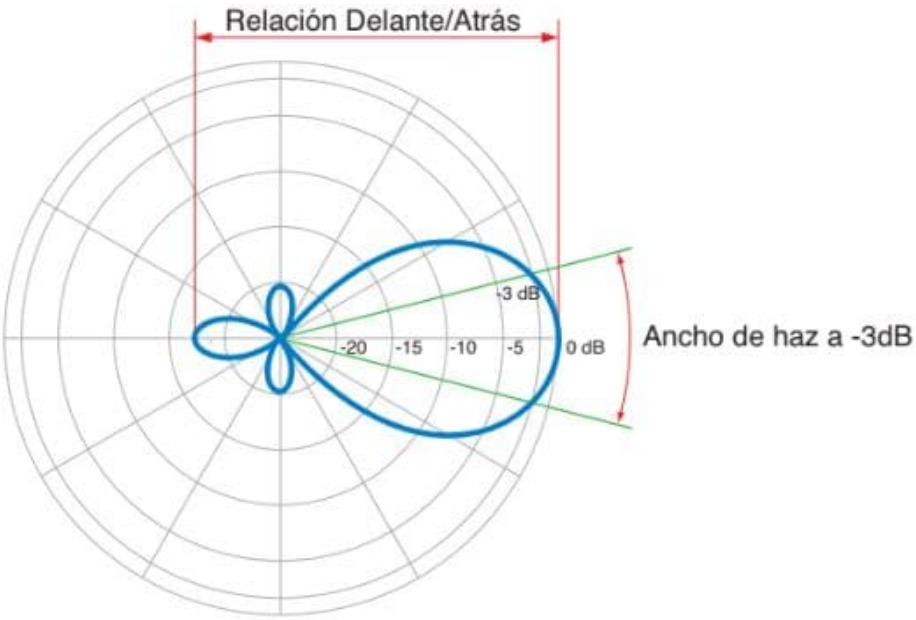


Figura 7.9 – *Artchist.blogspot.com*. Directividad de antenas [65]

7.8.1. Ancho de haz

Para determinar la precisión que requiere este sistema, se estudiará el ancho de haz requerido en una serie de ejemplos de aplicación, utilizando páginas web que calculan los anchos de haz en función del diámetro de nuestra antena y la frecuencia deseada [11].

7.8.1.1. Línea de hidrógeno

Se trata de una radiación electromagnética ampliamente observada en el ámbito de la radioastronomía. Su frecuencia es de 1,42 GHz.

Diameter of your satellite dish	Frequency	Efficiency 0.6-0.8
18 metres	1.42 GHz	0.65

Click to calculate result | Click to default values

dB contour down	Full beam width deg.	Half beam width degr.	Gain dBi
0	0	0	46.68
0.25	0.23	0.11	46.43
0.5	0.32	0.16	46.18
1	0.46	0.23	45.68
2	0.65	0.32	44.68
3	0.79	0.4	43.681
4	0.91	0.46	42.68
4.5	0.97	0.48	42.18
5	1.02	0.51	41.68
6	1.12	0.56	40.68
7	1.21	0.6	39.68
8	1.29	0.65	38.68
9	1.37	0.69	37.68

Figura 7.10 – Ancho de haz para la línea de hidrógeno [11]

Así, en el caso de utilizar un diámetro de disco de 18 metros (ya que estamos hablando en términos de radioastronomía), el ancho de haz a media potencia, es decir, para una atenuación de 3 dB respecto al punto de máxima potencia (0 dB), será de 0.79º.

7.8.1.2. Antena VSAT

Este tipo de antenas, cuyas siglas son Very Small Aperture Terminal, se utilizan en comunicaciones por satélite, como es el caso de internet por satélite. Un caso típico sería el de una antena de 1,2 metros de diámetro y 14 GHz.

Diameter of your satellite dish		Frequency		Efficiency 0.6-0.8
1.8	metres	14	GHz	0.65

Click to calculate result | Click to default values

dB contour down	Full beam width deg.	Half beam width degr.	Gain dBi
0	0	0	46.56
0.25	0.23	0.12	46.31
0.5	0.33	0.16	46.06
1	0.46	0.23	45.56
2	0.66	0.33	44.56
3	0.8	0.4	43.558
4	0.93	0.46	42.56
4.5	0.98	0.49	42.06
5	1.04	0.52	41.56
6	1.14	0.57	40.56
7	1.23	0.61	39.56
8	1.31	0.66	38.56
9	1.39	0.7	37.56

Figura 7.11 – Ancho de haz para una antena VSAT [11]

Se observa que el ancho de haz a media potencia es similar al del caso anterior, 0,8º.

7.8.1.3. Comunicaciones con la Estación Espacial Internacional

Tal como se ha mencionado en el capítulo 2 (alcance del proyecto), este se encuentra enmarcado en una colaboración con el CIFP Ferrolterra, cuyo objetivo es lograr la comunicación con la ISS (International Space Station).

Para este caso se prentende instalar una antena de tipo Yagi de 2 metros de largo en 145 MHz, resultando un ancho de haz de 44,4º [9].

7.8.2. Conclusión

El microcontrolador que incropora la placa Arduino MEGA 2560 es un Atmega 2560. En su manual [19] se indica que su convertidor A/D es de 10 bits, por lo que la precisión de la realimentación por potenciómetros, para azimut y elevación, será, respectivamente, $\frac{360}{2^{10}}$ y $\frac{180}{2^{10}}$, es decir, 0,35º para azimut y 0,17º para elevación.

Por otro lado, el sistema ofrece mayor precisión utilizando la realimentación de posición a través de los encoders, 0,09º, tal como se ha descrito en la sección 7.3.

De las secciones anteriores puede deducirse que para un mayor tamaño de antena (u otro tipo de antena más directiva), los requisitos de precisión se incrementan.

Este tipo de antenas que requieren precisión elevada son utilizadas normalmente para observaciones de objetos lejanos a la Tierra (como nebulosas, estrellas lejanas, sondas de ex-

ploración espacial, etc), cuya posición respecto a la tierra sufre variaciones lo suficientemente pequeñas como para no requerir un sistema de seguimiento.

No obstante, el objetivo de este proyecto es el seguimiento de satélites de órbita LEO, en el que se utilizarán antenas de comunicación que requieren, como acabamos de ver, una precisión muy escasa, del orden de decenas de grados.

La precisión de este sistema queda determinada, por tanto, por la precisión de la realimentación de los motores y por el software de posicionamiento que, en cualquier caso, superan la precisión requerida por el objetivo principal del proyecto, que sería el apuntamiento a la ISS.

8 RESULTADOS FINALES

En este capítulo se describe el resultado final del conjunto del proyecto, cuya implementación queda reflejada en el diseño de una placa de circuito impreso, que puede consultarse en los planos adjuntos en este documento, así como los resultados de cada uno de los bloques por los que está formado el mismo.

A esta placa se conectarán los distintos dispositivos mencionados en el capítulo 7 (análisis de soluciones).

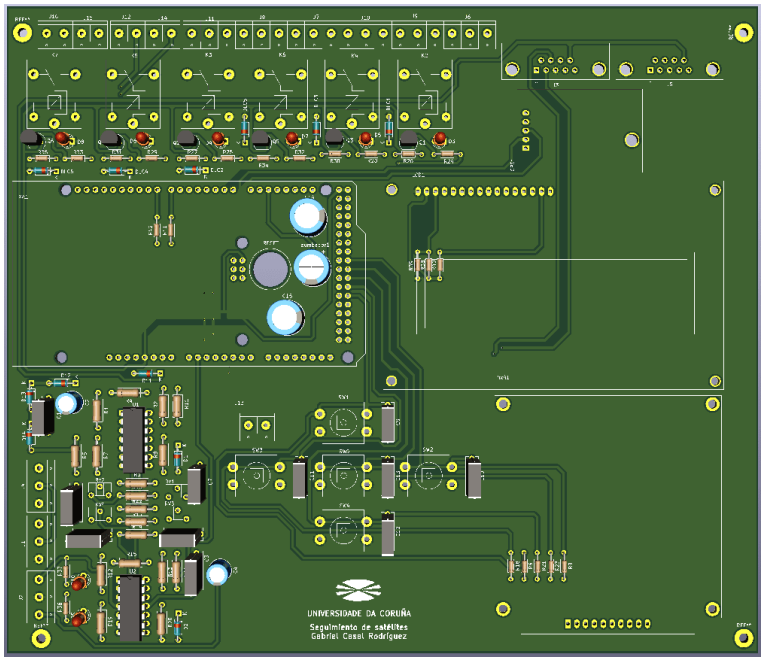
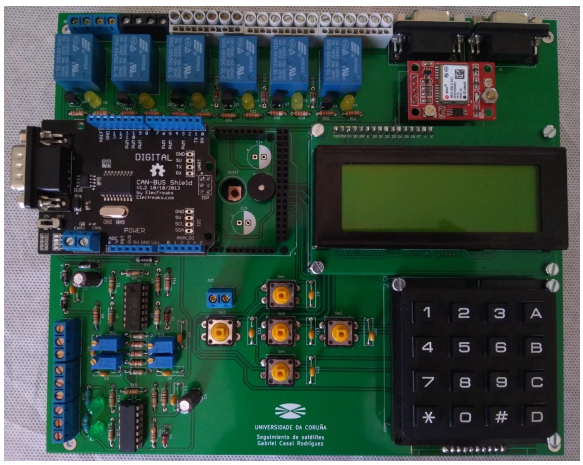
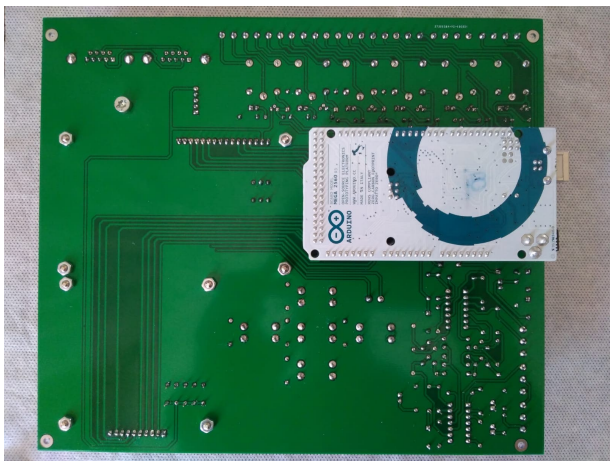


Figura 8.1 – Placa de circuito impreso



(a) Cara front PCB



(b) Cara bottom PCB

Figura 8.2 – Montaje de circuito impreso

Por otro lado, el análisis detallado de cada una de las partes en las que se divide el proyecto quedan descritas en los anexos del documento.

8.1. Diagrama de bloques general del sistema

Con las soluciones aportadas en el capítulo 7, el diagrama de bloques general del sistema queda como se ilustra en la figura 8.3.

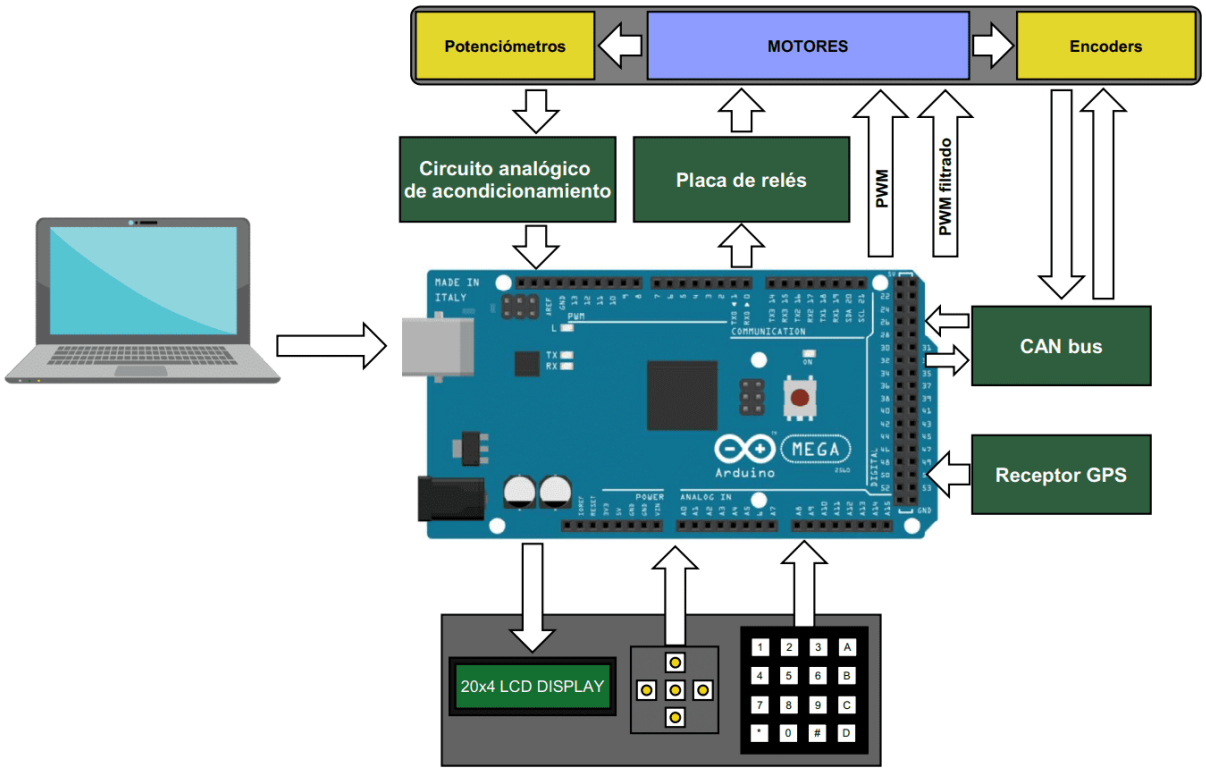


Figura 8.3 – Diagrama de bloques del sistema

A partir de él se describirá en los siguientes apartados la solución final adoptada y el modo de implementación de la misma para cada bloque.

8.2. HMI. Interfaz con el usuario

Todo el sistema queda gestionado por el software cargado sobre el microcontrolador. Este está basado en un sistema de menús que permiten al usuario acceder a distintos modos de funcionamiento y configuración que se ilustra en la figura 8.4:

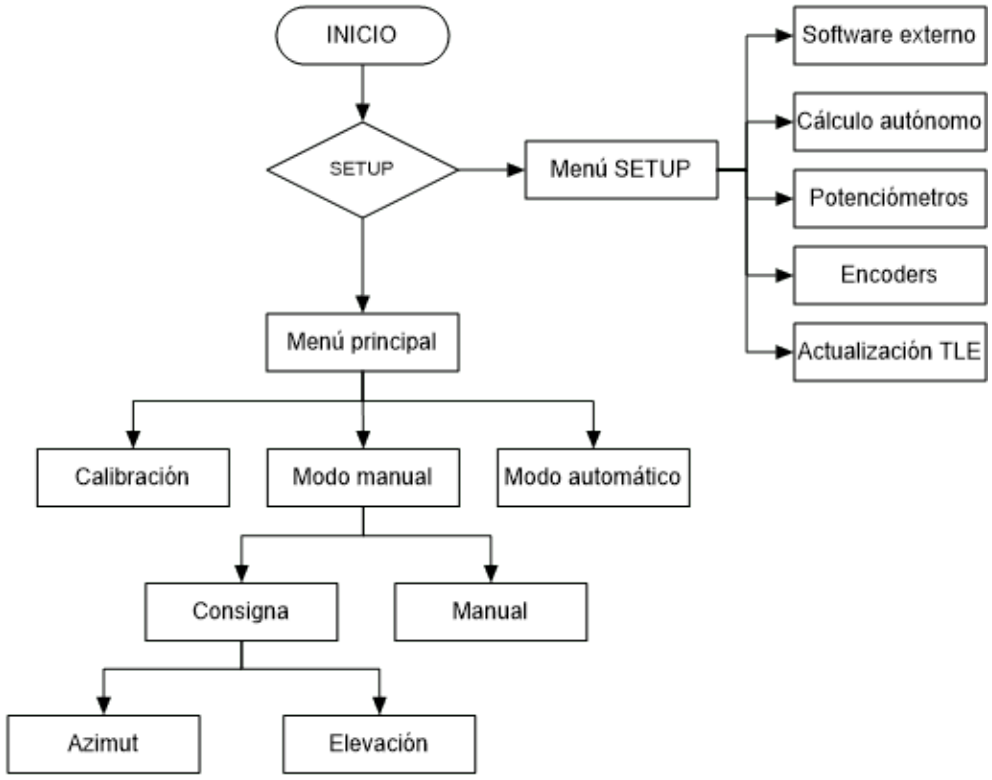


Figura 8.4 – Flujograma principal del software

Entrando en el setup del sistema, el usuario puede seleccionar:

- Modo de realimentación de los motores: encoders o potenciómetros
- Modo de obtención de la posición de apuntamiento: software externo o cálculo autónomo
- Actualización de los parámetros TLE

En caso de no acceder a la configuración en el setup, el sistema guardará los valores por defecto de la misma.

Una vez configurado el sistema, el usuario puede acceder a través del menú a tres opciones principales:

- Calibración:
  Permite realizar el ajuste de offset y ganancia del circuito analógico para adaptar la señal del potenciómetro de realimentación

  El procedimiento consistirá, de igual manera para los dos motores, en llevar el motor manualmente a la posición de 0º y realizar seguidamente el ajuste de offset de manera que se obtengan 0V a la salida del circuito analógico.

  Ajustado el offset, se llevará el motor a la posición extrema contraria, para ajustar la ganancia de manera que se obtengan 5V a la salida del circuito de acondicionamiento.

- Modo manual:
  Este modo de funcionamiento permite al usuario posicionar las antenas manualmente de dos maneras:

- Introducción de consigna: el usuario introducirá una consigna numérica a través del teclado
- Manual: el usuario podrá accionar los motores de manera completamente manual a través de la botonera

- Modo automático:
  Es el modo de funcionamiento principal del sistema. En él se realiza el apuntamiento a la posición del satélite en todo momento.

  Por seguridad, se ha añadido a este modo de funcionamiento un sistema de detección de cambio de sentido.

  Un cambio de sentido sucede, por ejemplo, si la antena se pasa de la posición de consigna, o bien si hay un cambio de consigna que obligue a los motores a girar en sentido contrario al que actualmente se encuentra girando. Este último caso puede darse, por ejemplo, si en el software de cálculo externo se cambia el satélite que se desea seguir sin antes haber detenido los motores.

  Si el sistema detecta un cambio de sentido, frenará los motores antes de realizar el cambio.

El sistema cuenta con una pantalla LCD en la que el usuario visualizará los distintos menús y configuraciones disponibles, así como los parámetros necesarios en cada modo de funcionamiento.

Para seleccionar los modos de funcionamiento y configuraciones deseadas (o salir de ellos cuando sea preciso), el usuario cuenta con una botonera de cinco botones (arriba, abajo, derecha, izquierda y centro).

8.3. Realimentación de la posición de los motores

8.3.1. Acondicionamiento analógico de los potenciómetros de posición

Se ha indicado anteriormente en el capítulo 7 (análisis de soluciones) que se precisa de un circuito analógico que acondicione la señal de los potenciómetros de posición de las antenas, de manera que se optimice el funcionamiento del convertidor A/D del microcontrolador.

En primer lugar debe garantizarse que en la posición de 0º, los potenciómetros de posición de las antenas proporcionen 0V. Esto se logra con el ajuste de offset.

Posteriormente, es necesario que la posición más alejada de la inicial de 0º, es decir, 180º para elevación y 360º para azimut, la salida del circuito analógico sea de 5V, el máximo valor legible por el A/D. Esto debe lograrse independientemente de la alimentación de los potenciómetros de posición. Expresado en otras palabras, la máxima posición del motor debe corresponderse con 5V a la salida del circuito, al margen de que los potenciómetros se encuentren alimentados a 3V, a 5V, a 15V, o a cualquier otro valor que se encuentre dentro de este rango. Esto se consigue ajustando la ganancia del circuito.

El diagrama de bloques es el siguiente:

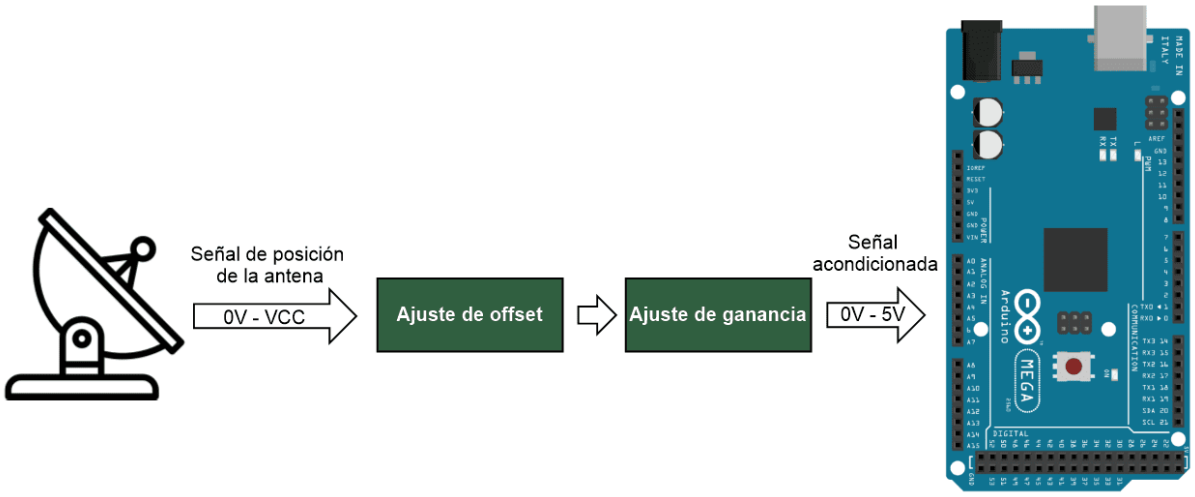


Figura 8.5 – Diagrama de bloques del acondicionamiento de señal

En el anexo de cálculos 11 se indican las explicaciones y cálculos que se han realizado para concluir con el diseño final del circuito de acondicionamiento analógico (necesario para cada motor), que se muestra en la figura 8.6.

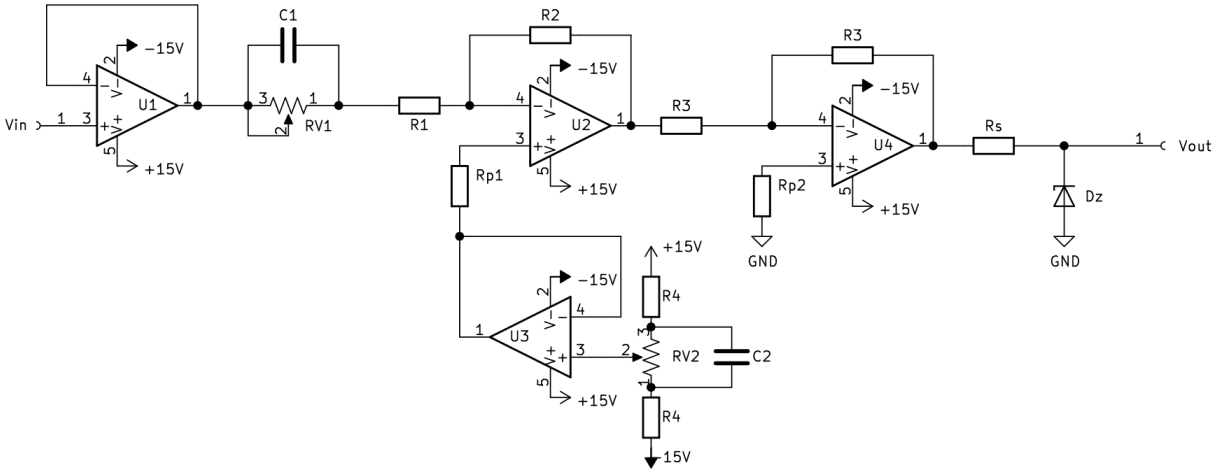


Figura 8.6 – Circuito analógico acondicionamiento de señal

Este circuito se implementa físicamente sobre la placa de circuito impreso, cuyo aspecto final se ilustra en la figura 8.7.

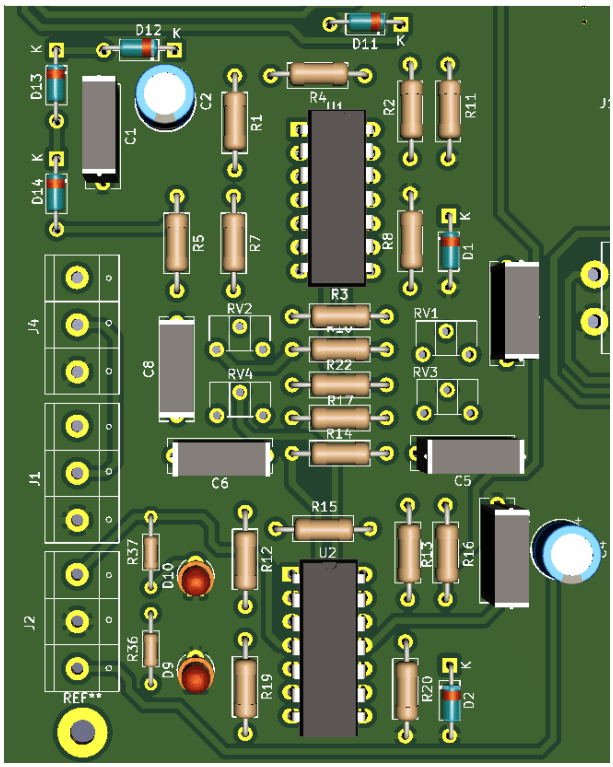


Figura 8.7 – Zona de acondicionamiento de señal del PCB

8.3.2. Comunicación con los encoders

Como se ha indiciado en el la sección 7.3, se utilizan los encoder TBN 50-SA 2048R C2 S N14 del fabricante alemán TWK-Electronik, basados en el protocolo de comunicaciones CANopen.

Para lograr la comunicación entre el microcontrolador y los encoders, se ha indicado también en la sección 7.3 que se utiliza el shield para comunicaciones CAN de Arduino.

En cuanto a la programación del software para lograr la comunicación entre el microcontrolador y los encoder, esta puede encontrarse en los anexos 12 del documento, así como una serie de conocimientos previos a cerca del protocolo de comunicaciones CANopen, necesarios para comprender dicha programación.

Se verifica, a través del puerto serie, el correcto funcionamiento de los encoder actuando simultáneamente, como se puede ver en la figura 8.8:

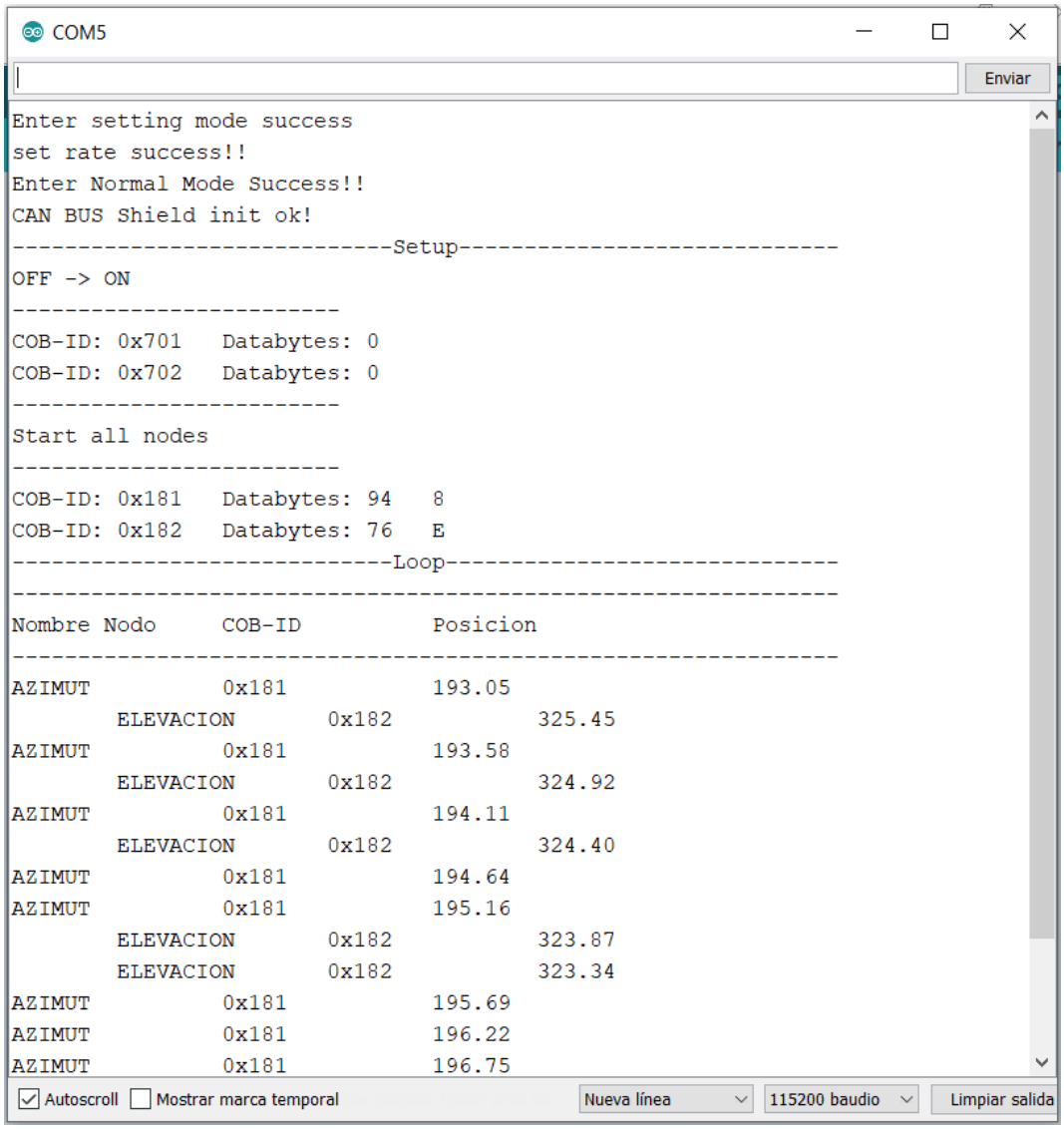


Figura 8.8 – Encoders proporcionando la ubicación sumultáneamente

Para poder visualizar estos resultados por puerto serie, se han realizado las modificaciones necesarias al código referente al protocolo CANopen del código principal.

8.4. Cálculo autónomo de la posición de apuntamiento

Para realizar el cálculo de los ángulos de azimut y elevación de manera autónoma, se indica en la sección 7.1.2 que los cálculos se realizarán en base a los parámetros TLE del satélite que se desea seguir.

El software de cálculo aplica el procedimiento SGP4 (desarrollado por la NASA y NORAD y que es de libre acceso para su uso en aplicaciones) y está basado en los códigos de David Vallado [76] y Michael F. Henry [51].

Las bases teóricas para comprender los cálculos realizados pueden consultarse en el anexo 15, y la explicación del software implementado junto con las modificaciones realizadas sobre el mismo respecto al original, en el anexo 16.

8.4.1. Obtención de datos GPS

En la sección 7.1.2 se ha establecido que, para realizar los cálculos keplerianos, es necesario conocer la ubicación del observador, así como el instante de observación. Para esto se utilizara el receptor GPS indicado en esa misma sección.

Este receptor GPS transmite al microcontrolador a través de puerto serie los datos en formato NMEA, cuya decodificación se encuentra detallada en el anexo 13.

En el código principal del software se puede encontrar la programación referente a la decodificación del formato GPS, que se ha adaptado para poder visualizar la decodificación por puerto serie (figura 8.9).

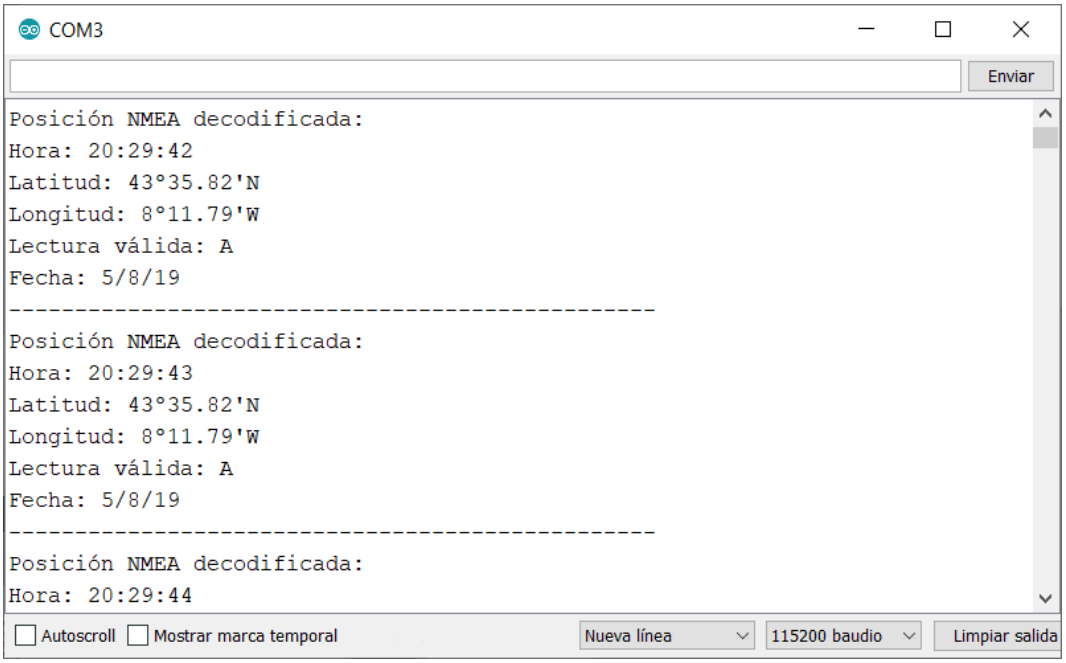


Figura 8.9 – Resultado de la decodificación GPS

8.4.2. Verificación de la posición de apuntamiento

La verificación del funcionamiento consiste en comparar los datos obtenidos con los proporcionados por el software PstRotator, y se puede comprobar, comparando las figuras 8.10 y 8.11, que los resultados son satisfactorios.

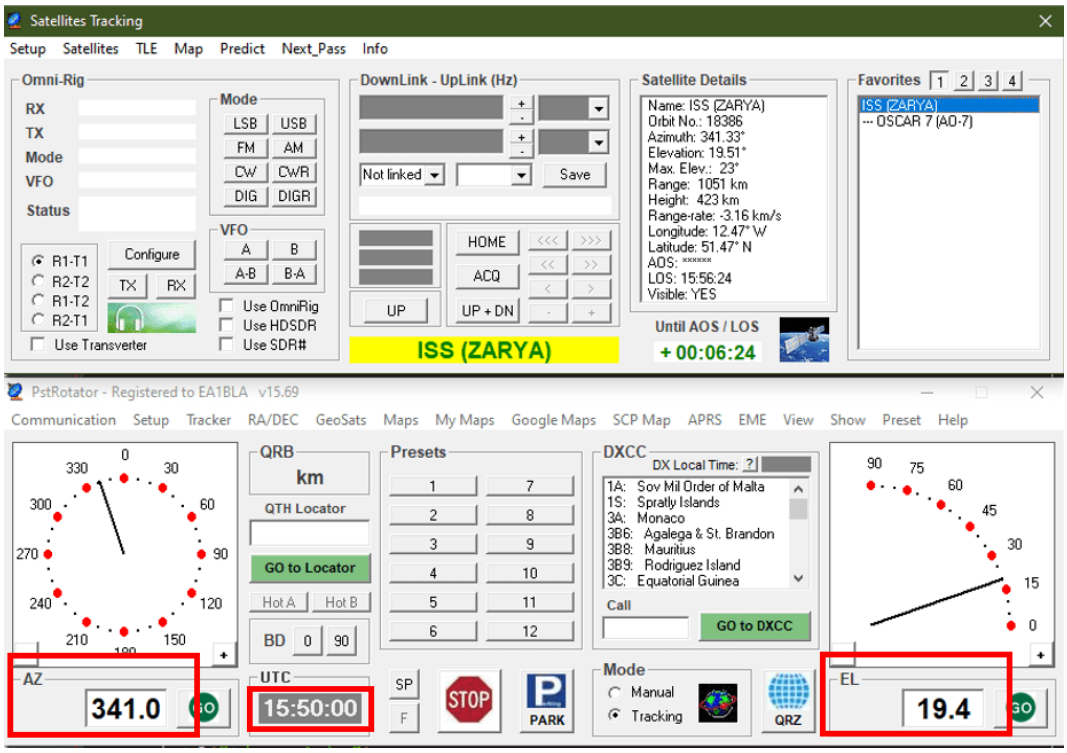


Figura 8.10 – Posición de apuntamiento PstRotator

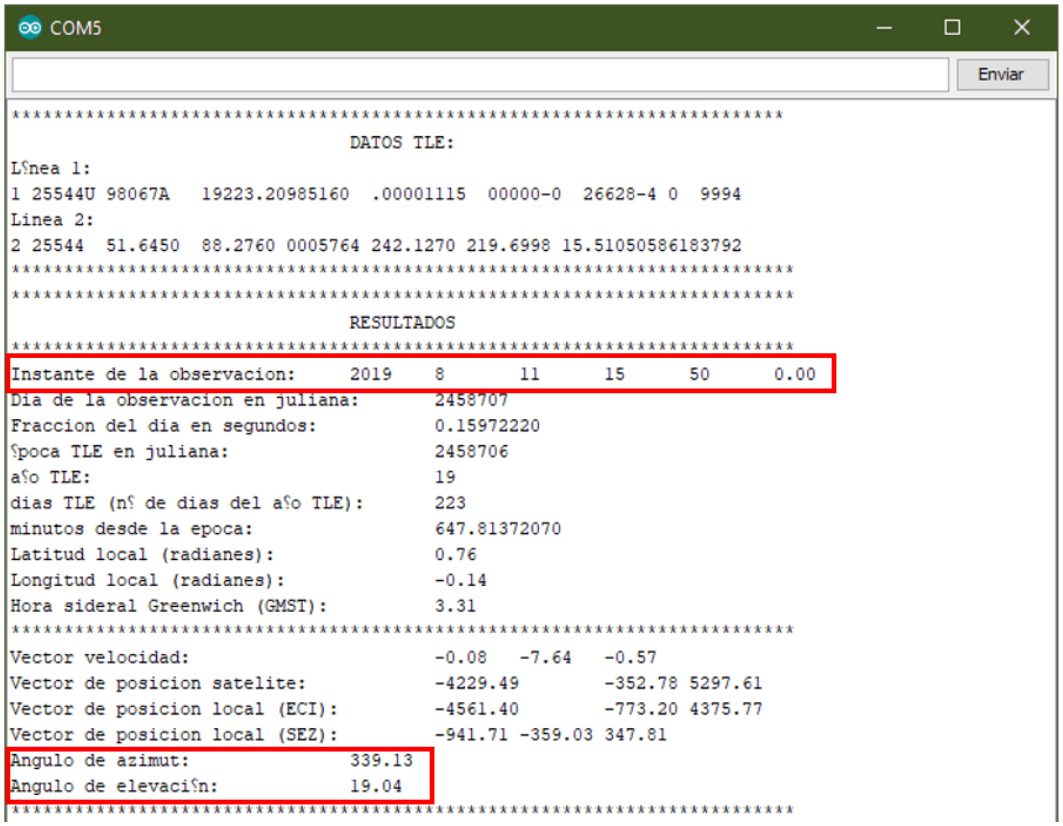


Figura 8.11 – Posición de apuntamiento Arduino

No obstante, debido a ciertos problemas de precisión ocasionados por las limitaciones del microcontrolador, cuyo tratamiento se aborda en el anexo 16, puede observarse que, aunque

muy aproximados, los resultados no son completamente exactos.

8.5. Comunicación con el PC

El diseño de la placa de circuito impreso proporciona fácil acceso al puerto USB de la placa Arduino (figura 8.1), necesario para cargar el programa final en el microcontrolador, así como para comunicar por puerto serie los datos de apuntamiento del satélite en el caso de no estar realizando los cálculos de direccionamiento de manera autónoma o, por otra parte, actualizar los parámetros TLE para los cálculos keplerianos.

8.5.1. Dirección de apuntamiento por software externo

Se ha indicado en la sección 7.1.1 que se utilizará el software PstRotator para transmitir los ángulos de apuntamiento al microcontrolador en formato Easycom por puerto serie. En el anexo 14.1 se indica la configuración necesaria de este software para transmitir la posición, así como la explicación de la decodificación Easycom del programa principal para descifrar la trama recibida (adaptado para poder verificar su funcionamiento en el monitor serie del IDE Arduino). El resultado de dicha decodificación se muestra en la figura 8.12.

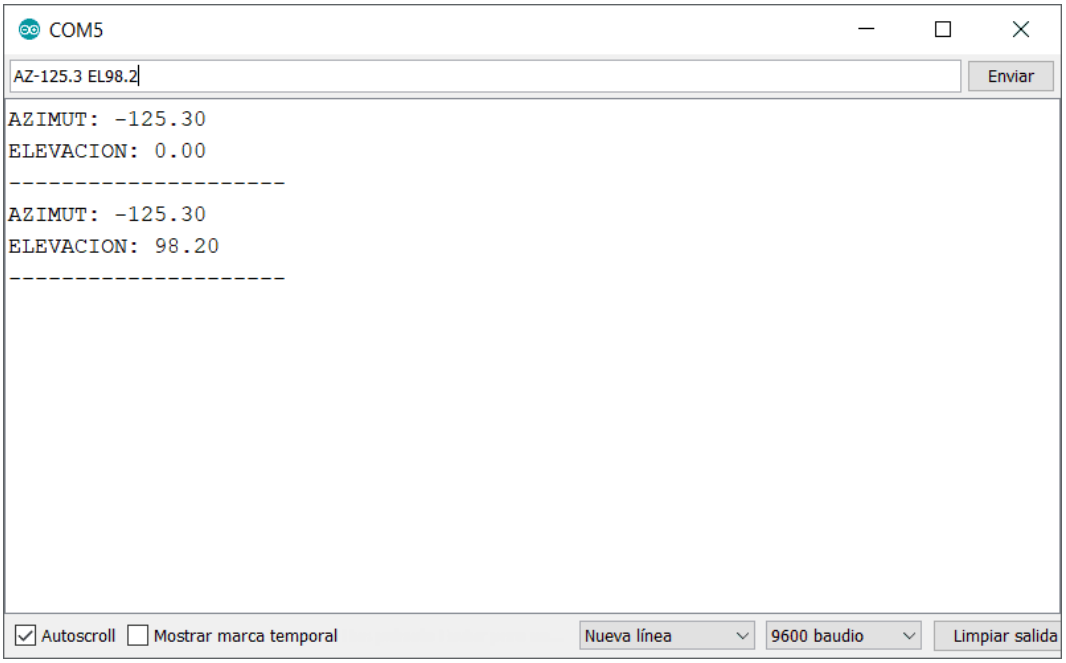


Figura 8.12 – Posición de apuntamiento Easycom decodificada

8.5.2. Actualización de parámetros TLE

Dado que los parámetros TLE son actualizados con frecuencia, es necesario establecer un modo de actualizar los datos TLE guardados en el microcontrolador. Para ello, en el menú del setup, tal como se observa en la figura 8.4, existe la opción de actualizarlos desde dicho menú. Será necesario abrir el monitor serie del IDE Arduino, en el que se muestra al usuario los pasos

a seguir, además del enlace a la web donde se encuentran los parámetros TLE actualizados [28].

Una vez el usuario haya enviado los datos por el puerto serie, estos se gravarán en la EEPROM del microcontrolador, y serán los que se utilicen en los cálculos keplerianos. Este proceso se ilustra en la figura 8.13.

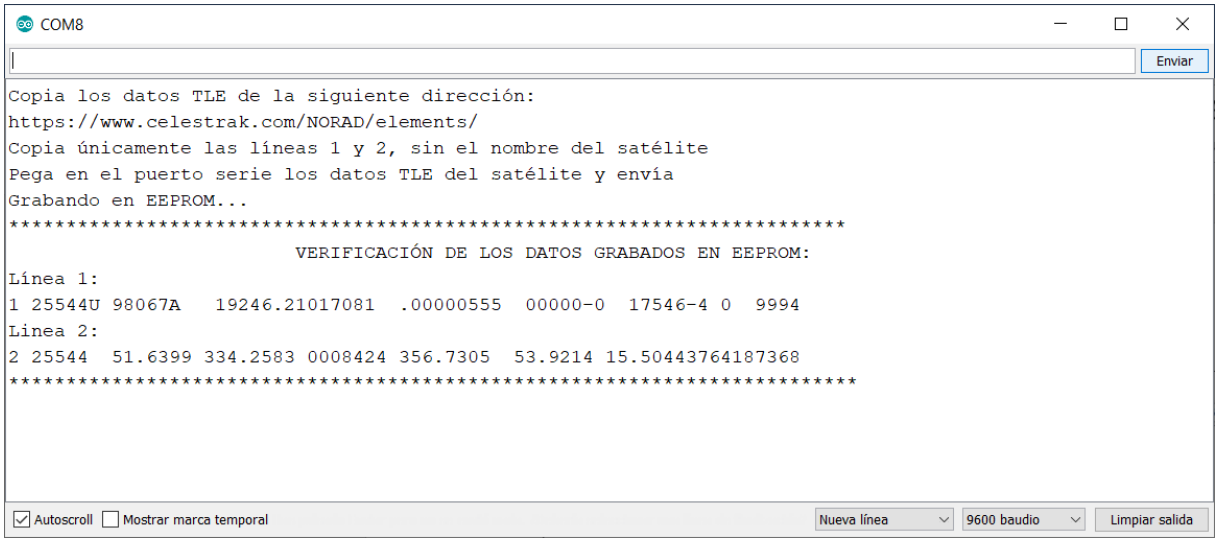


Figura 8.13 – Actualización TLE en monitor serie

Cabe destacar que, en caso de abrir el monitor serie del IDE Arduino con el sistema en funcionamiento, la propia acción de abrir el monitor serie reseteará el sistema y será necesario acceder de nuevo a la opción de actualización de TLE (con el monitor serie ya abierto).

8.6. Accionamiento sobre los motores de la antena

Como se indica en la sección 7.2, la placa de circuito impreso incorpora seis relés (de contactos libres de potencial) para actuar sobre los motores, además de seis LEDs, uno por cada relé, para indicar en cada instante que relé se encuentra activado.

Se incorpora también en la placa dos salidas de control de velocidad por cada motor, una de PWM para motores de corriente continua y otra de PWM filtrado, pensada para motores de corriente alterna gobernados por variador de frecuencia con consigna analógica, ambas de 0V a 5V.

Esta señal PWM es generada por el software de control de posición y velocidad, detallado en el anexo 17.3.

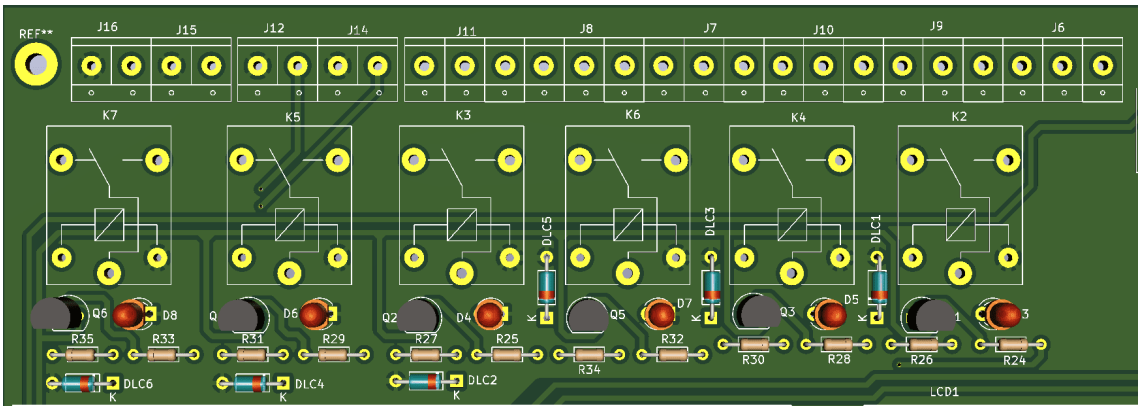


Figura 8.14 – Zona de accionamiento sobre motores del PCB

8.7. Posibles mejoras

A continuación se describen una serie de aspectos que se podrían añadir o mejorar en este proyecto:

- Un indicador del tiempo restante hasta que el satélite vuelva a estar en rango.

  Los satélites de órbita LEO (como la ISS), tienen un periodo orbital que abarca una o dos horas. No obstante, en todo ese período, el tiempo que permanece en rango el satélite para el observador es nada más que unos minutos.

  Es por esto que sería de utilidad dotar al sistema con la capacidad de calcular el tiempo restante hasta que el satélite vuelva a estar en rango para el observador.

- Indicador de la fecha de actualización de TLE.

  Se menciona en diversos apartados la importancia de mantener actualizados los datos TLE del satélite que se desea seguir. Así, la posibilidad de consultar la última fecha de actualización de los TLE grabados en el controlador sería una aportación útil al proyecto.

- Mejora en la precisión de los cálculos keplerianos.

  Las limitaciones del propio microcontrolador, aspecto que se trata en el anexo 16, producen imprecisión en el cálculo de la posición de apuntamiento, en torno a 1º. Este aspecto podría solventarse realizando el diseño del sistema que emplee un microcontrolador más potente.

  El único microcontrolador de la plataforma Arduino que presenta una potencia mayor que el utilizado en este proyecto, y, al mismo tiempo, un número de entradas y salidas suficiente, es el ATSAM3X8E de la placa DUE (tabla comparativa de la figura 7.7).

  No obstante, el hecho de que su tensión de operación sea 3,3V en lugar de 5V hace necesario el redimensionamiento eléctrico de todo el proyecto.

- Anticipación de la posición de apuntamiento.

El sistema diseñado de seguimiento obtiene siempre la posición actual del satélite, por lo que puede entenderse que la antena irá siempre por detrás del satélite.

Una posible mejora sería dotar al sistema con una capacidad de anticipación de la posición de apuntamiento, calculando dicha posición para un instante lo suficientemente adelantado al actual como para que el la antena apunte en todo momento a la posición que ocupa el satélite en su órbita.

Esta modificación es fácilmente implementable con la opción del cálculo autónomo, en el que habría que, simplemente, sumar el lapso de tiempo necesario a la hora proporcionada por el GPS.

Sin embargo, no se ha llevado a cabo dado que no se disponía del montaje definitivo del sistema de antenas, necesario para determinar el lapso de tiempo óptimo.

9 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

Frente a posibles discrepancias, se establece el orden de prioridad de los documentos del TFG como se describe a continuación:

1. Planos
2. Pliego de Condiciones
3. Presupuesto
4. Memoria
5. Anexos

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice del documento ANEXOS

10 DOCUMENTACIÓN DE PARTIDA

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Casal Rodríguez,Gabriel*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** Feb2019
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Sistema de seguimiento de satélites

Número TFG: *770G01A170*

TUTOR:(Titor) *Rivas Rodriguez,Juan Manuel*

COTUTOR/CODIRECTOR: *Esteban Jove Pérez*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Viernes, 28 de Junio del 2019

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Casal Rodríguez,Gabriel

DESCRIPCIÓN Y OBJETIVO:OBJETO:

El sistema debe estar basado en la plataforma Arduino. Las entradas y salidas serán:

- Recibirá a través de un puerto serie, o una cadena ASCII en formato GS232 o Easycomm indicando acimut y elevación, o los parámetros keplerianos que definen el movimiento del satélite. En este segundo caso el programa debe hacer los cálculos correspondientes.

- La realimentación de la posición de las antenas. Esta podrá de alguna de las siguientes formas ser:

  o Con una tensión analógica que debe ser acondicionada para optimizar el convertidor A/D.
  o A través de un encoder digital absoluto, en este caso la resolución mínima debe ser de 10 bits.
  o A través de un encoder incremental con control de paso por cero.
  o Otro posible sistema aceptado por el director del proyecto.

- La actuación sobre los motores que podrá ser:

  o De corriente continua. La velocidad se regulará por PWM.
  o De corriente alterna. La velocidad vendrá marcada por una tensión analógica para controlar el variador de frecuencia.
  o Deberá variar la velocidad en función de la distancia a inicio y al final del movimiento.

- Un display que nos indicará en tiempo real de la posición de las antenas, además de servir de interface para los menús de calibración.

- Un teclado que permitirá el movimiento manual del sistema e introducir los datos de configuración.

ALCANCE:

Este proyecto está enmarcado en una colaboración con el CIFP Ferrolterra que deberán realizar la construcción de todo el sistema de antenas.

En caso de que terminen en plazo su construcción el alumno debe montar un prototipo e integrarlo con dicho sistema.

En caso contrario se deberá realizar un modelo a escala para verificar el correcto funcionamiento.

11 CÁLCULOS

11.1. Circuito analógico de acondicionamiento de la señal del potenciómetro

Para realizar la realimentación de la posición del los motores mediante potenciómetros, será necesario realizar el un acondicionamiento de la señal, de manera que esta se encuentre en todo momento dentro de los rangos de entrada de la placa Arduino para optimizar la conversión analógico-digital. Se debe de poder ajustar el offset y la ganancia del circuito para que este pueda adaptarse a distintas alimentaciones del potenciómetro, de manera que a la salida del mismo, el rango de la señal sea siempre el establecido por el rango de entrada de la placa Arduino, de 0V a 5V. El rango de alimentaciones posible que se ha establecido para los potenciómetros es de X a Y.

Además, este ajuste de ganancia y de offset nos permitirá calibrar el sistema de antenas:

- El ajuste de offset permitirá obtener a la salida del circuito 0V cuando el motor se encuentre en la posición de 0º.
- El ajuste de ganancia permitirá obtener 5V a la salida del circuito cuando el motor se encuentre en la posición de 360º (180º para el motor de elevación).

Para realizar estos ajustes se ha diseñado un circuito analógico con amplificadores operacionales.

11.1.1. Ajuste de ganancia

Para realizar el ajuste de ganancia será necesario poder conseguir, por un lado, incrementar el voltaje máximo de salida hasta 5V, y por otro será necesario poder conseguir también reducir el voltaje máximo de salia a 5V. Es por esto que el montaje no inversor (figura [11.1]) queda descartado, ya que su ganancia será siempre mayor que la unidad: $Vout = Vin(\frac{R2}{R1} + 1)$

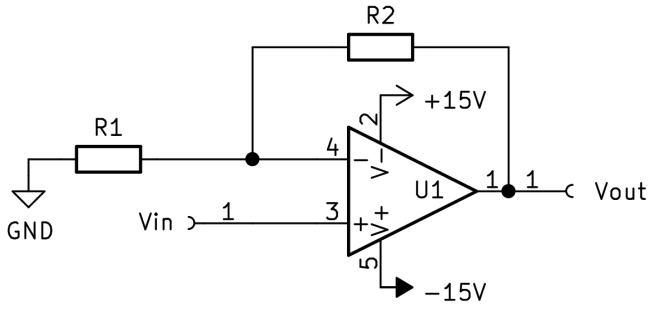


Figura 11.1 – Circuito no inversor

La configuración inversora (figura 11.2), por el contrario, sí nos permite obtener nuestro rango de ganancias, aunque, como el nombre indica, de manera invertida: $Vout = -Vin\frac{R2}{R1}$

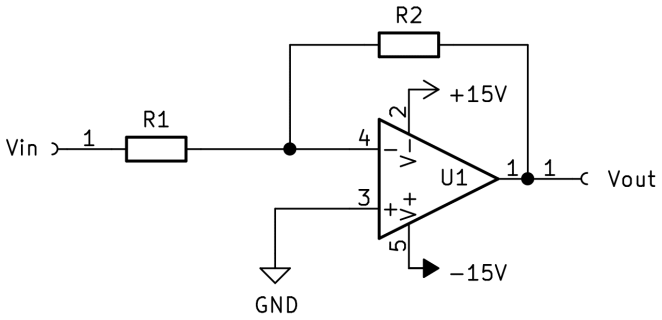


Figura 11.2 – Circuito inversor

Por lo tanto, se requieren dos etapas inversoras (figura [11.3]), una de ganancia variable y otra de ganancia unitaria, es decir, $R2 = R1$.

Cabe destacar que, con el fin de evitar corrientes de polarización en el amplificador operacional, se utilizará una resistencia compensadora en la entrada no inversora, que se corresponde con la resistencia del equivalente de Thevenin visto desde la entrada inversora, el paralelo, $R_{TH} = R1||R2$

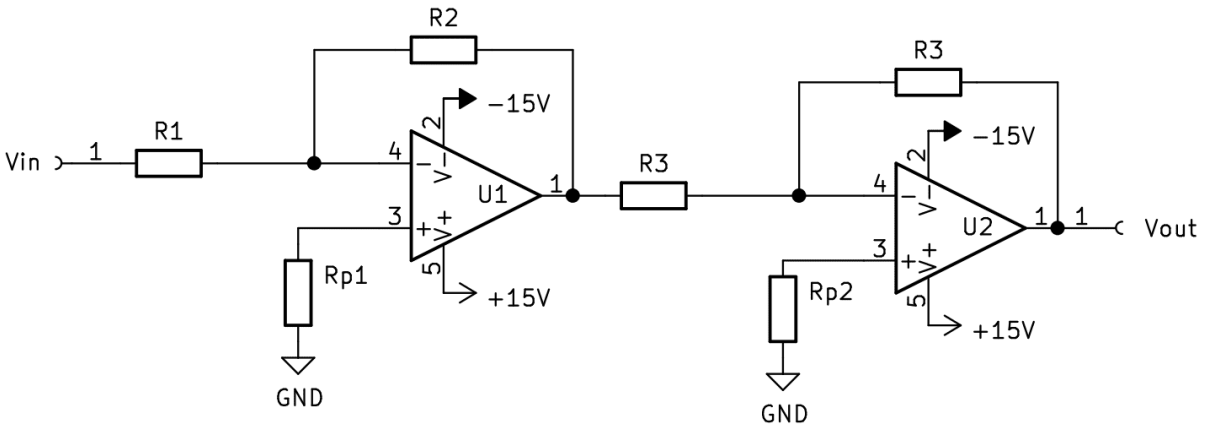


Figura 11.3 – Circuito de dos etapas inversoras

La ganancia total será:

$$V_{out} = -V_{in} \cdot \frac{R2}{R1} \cdot (-\frac{R}{R}) \quad (11.1)$$

$$= V_{in} \cdot \frac{R2}{R1} \quad (11.2)$$

Para poder variar la ganancia, sustituiremos R1 por un potenciómetro con una resistencia serie (para evitar que, en el caso de que el potenciómetro se ajuste a 0Ω, la ganancia sea $V_{in} \cdot \frac{R2}{0}$).

11.1.2. Ajuste de offset

Para realizar el ajuste de offset, se coloca el circuito de compensación en la entrada no inversora de la primera etapa del circuito.

El circuito de compensación se muestra en la siguiente figura [11.4]:

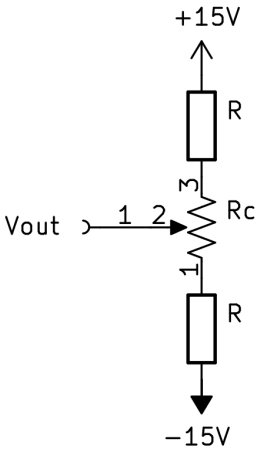


Figura 11.4 – Compensación de offset

Con el objetivo de que el divisor este equilibrado, las dos resistencias será del mismo valor óhmico. Para realizar el cálculo de los valores de los componentes del valor de tensión, establecemos en primer lugar el rango de variación de la tensión de offset, que será, de manera muy aproximada, el $\pm 10\,\%$ de la tensión de alimentación.

11.1.3. Selección del circuito integrado

Teniendo en cuenta que nuestro circuito necesita dos amplificadores operacionales, uno para cada etapa, y que además contamos con el circuito de ajuste de offset, se ha seleccionado el circuito integrado LM324N [72], que cuenta con 4 amplificadores operacionales. Se utilizan los dos operacionales restantes como seguidores de tensión para la entrada y para el circuito de ajuste de offset. De esta manera se proporciona una impedancia de entrada elevada, tanto para la entrada como para la tensión de ajuste de offset, evitando así posibles pérdidas de tensión por corriente.

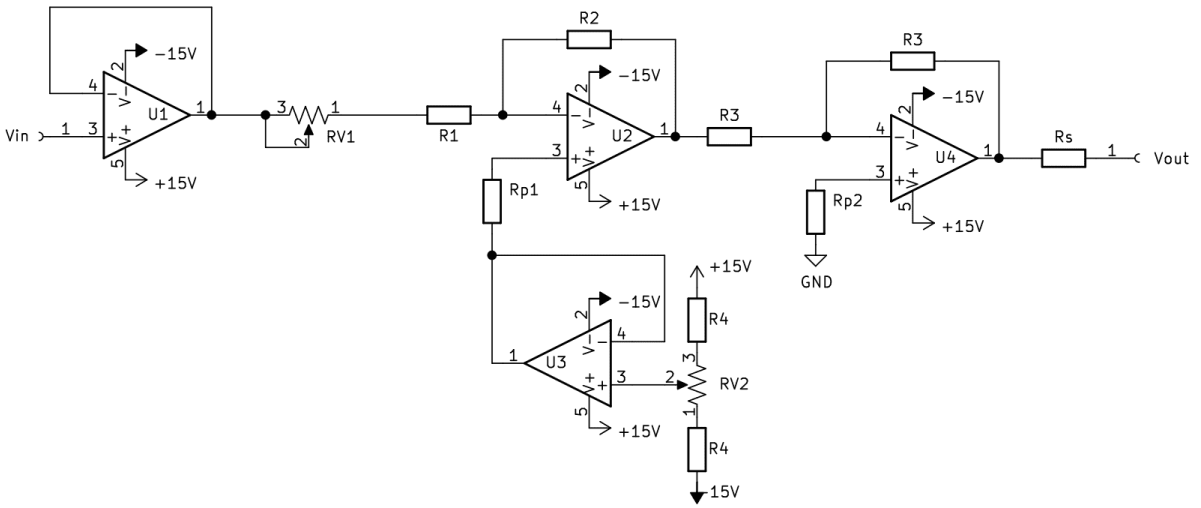


Figura 11.5 – Montaje con el circuito integrado LM324

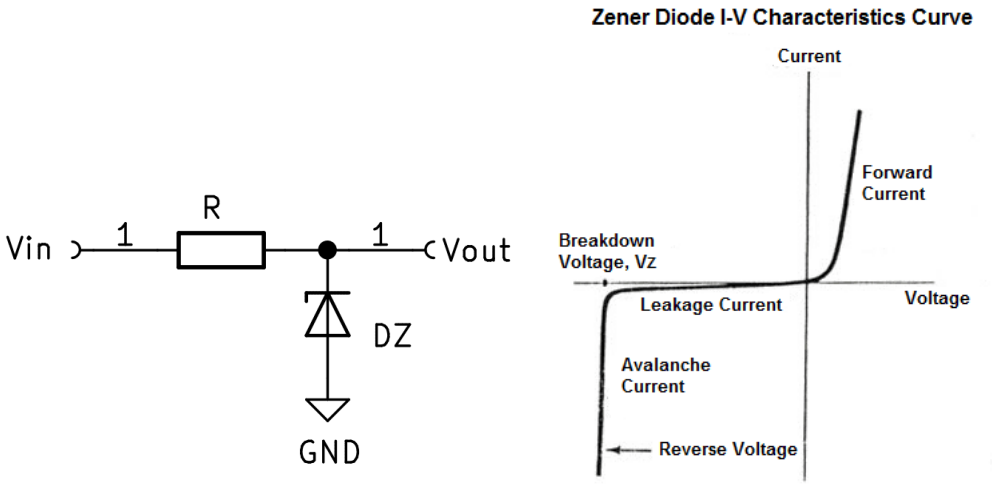
11.1.4. Ecuación general del circuito

$$V_{out} = V_{in} \cdot (-\frac{R2}{R1 + Rv1}) \cdot (-\frac{R}{R}) + V_{offset} \cdot (1 + \frac{R2}{R1}) \cdot (-\frac{R}{R}) \quad (11.3)$$

$$= V_{in} \cdot (\frac{R2}{R1 + Rv1}) \pm V_{offset} \cdot (1 + \frac{R2}{R1}) \quad (11.4)$$

11.1.5. Limitador de tensión

Por último, para asegurarnos de no dañar en ningún momento la placa Arduino, se diseñará un circuito de protección limitador de tensión [2]:



(a) Circuito limitador (b) blog.fossasia.org.*Zener diode* [83]

Figura 11.6 – Limitador de tensión

Se observa que el diodo zener se encuentra polarizado inversamente. Esto provocará que en el momento que la tensión de polarización inversa, es decir, Vout, supere la tensión zener Vz, el diodo entrará en corte, manteniendo la tensión constante e igual a la tensión zener Vz. De esa manera se evitan tensiones de entrada excesivas en la entrada analógica de la placa Arduino.

11.1.6. Circuito completo

Al unificar los apartados anteriores, se obtiene el circuito completo de acondicionamiento analógico de señal.

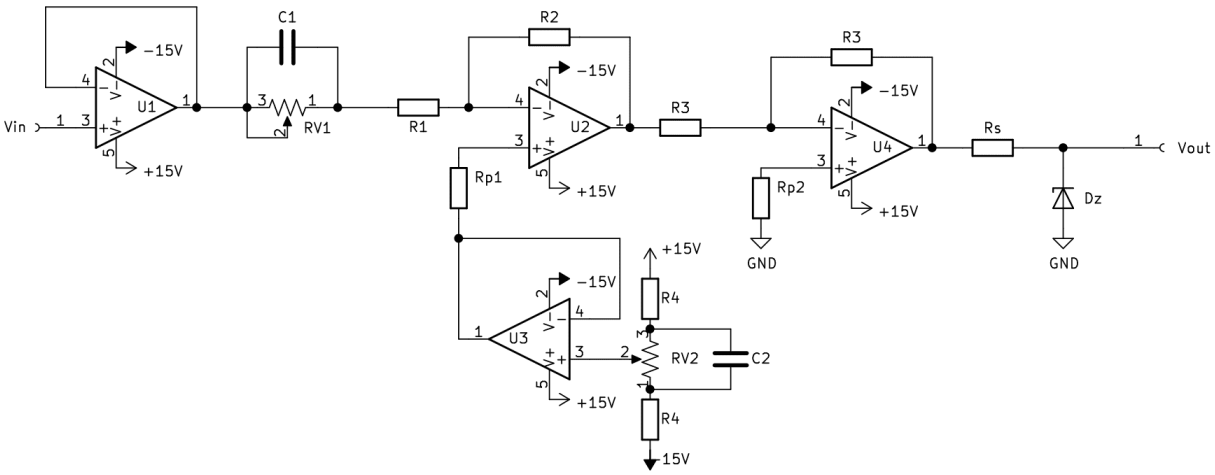


Figura 11.7 – Circuito analógico completo

Adicionalmente se han añadido al circuito los condensadores que se ven en la figura 11.7 en paralelo a ambos potenciómetros. De esta manera corregimos el ruido que producen estos últimos al mover el cursor.

11.1.7. Dimensionado de los componentes

11.1.7.1. Ajuste de ganancia

El circuito integrado LM324N [72] permite alimentaciones de $\pm 16V$, por lo que se podrán acondicionar señales cuyo máximo no supere $\pm 15V$. Esta señal debe ajustarse de manera que a la salida del circuito obtengamos siempre 5 V en la situación de voltaje más elevado. Para determinar los valores de las resistencias se estudiarán los casos extremos, separando por un lado el efecto del ajuste de ganancia y por otro el ajuste de offset:

- Alimentación de mínima de 3V:
  En esta situación requiere que la ganancia del circuito sea máxima, Rv1 en posición de 0Ω.

$$\begin{aligned} 5V &= 3V \cdot \frac{R2}{R1} \\ R1 &= \frac{3}{5} \cdot R2 \end{aligned} \tag{11.5}$$

  Si se establece un valor de $100k\Omega$ para R1, R2 serán $60k\Omega$, siendo $56k\Omega$ el valor comercial más cercano.

  Así, el valor mínimo de V_{in} es

$$\begin{aligned} 5V &= V_{in,min} \cdot \frac{100k}{56k} \\ V_{in,min} &= 2,8V \end{aligned} \tag{11.6}$$

- Alimentación de 15 V:
  De la ecuación 11.3 se deduce que:

$$\begin{aligned} 5V &= 15V \cdot \frac{R2}{R1 + Rv1} \\ \frac{R2}{Rv1 + R1} &= \frac{1}{3} \\ 3 \cdot R2 &= Rv1 + R1 \end{aligned} \tag{11.7}$$

  Sustituyendo valores,

$$\begin{aligned} 3 \cdot 100k &= Rv1 + 56k \\ Rv1 &= 244k\omega \end{aligned} \tag{11.8}$$

  Con $Rv1 = 500k\omega$ se obtiene un ajuste más holgado, de manera que el valor máximo de entrada es el siguiente:

$$\begin{aligned} 5V &= V_{in,max} \cdot \frac{100k}{56k + 500k} \\ V_{in,max} &= 27,8V \end{aligned} \tag{11.9}$$

Esta $V_{in,max}$ de 27,8V supera la establecida por la hoja de características del circuito integrado, de manera que seguirá siendo 1V por debajo de la tensión de alimentación.

Por tanto, las resistencias del ajuste de ganancia quedan de la siguietne manera:

$$R2 = R1 \tag{11.10}$$
$$Rv1 = 2 \cdot R2 \tag{11.11}$$
$$R1 = 100k\Omega \tag{11.12}$$
$$Rv1 = 200k\Omega \tag{11.13}$$

Por otra parte, cabe destacar que los valores de los condensadores utilizados en la eliminación del ruido de los potenciómetros se han obtenido experimentalmente. Se ha comprobado que que con una capacidad de 100 nF, se ha comprobado que se elimina el ruido de manera aceptable y con un tiempo de carga del condensador inapreciable.

Etapa de amplificación

Componente	Valor
R1	100 kΩ
Rv1	500 kΩ
R2	100 kΩ
C1	100 nF
C2	100 nF

Tabla 11.1 – Componentes de la etapa de amplificación

11.1.7.2. Ajuste de offset

Se desea que el divisor de tensión de ajuste de offset (figura 11.4) esté equilibrado, por lo que las dos resistencias será del mismo valor óhmico. Para realizar el cálculo de los valores de los componentes, establecemos en primer lugar el rango de variación de la tensión de offset, que será, de manera muy aproximada, el $\pm 10\,\%$ de la tensión de alimentación.

Así, para 15V, se aproxima el valor de ajuste de offset a $\pm 2V$ (13 % de 15V), y se establece también una intensidad Ic de 1mA. Llamaremos a la tensión en los extremos del potenciómetro Va y Vb:

$$Va - Vb = Ic \cdot Rc \tag{11.14}$$
$$4V = 1mA \cdot Rc \tag{11.15}$$
$$Rc = 4k\Omega \tag{11.16}$$

Para utilizar valores comerciales, se utilizará un potenciómetro de 5kΩ. Por último, se muestra el cálculo de las resistencias en las siguientes ecuaciones:

$$Vcc - (-Vcc) = Ic \cdot (Rc + R + R) \quad (11.17)$$
$$15V - (-15V) = 1mA \cdot (5k\Omega + 2R) \quad (11.18)$$
$$R = 12,5k\Omega \quad (11.19)$$

De nuevo, se utilizarán resistencias de 12kΩ para utilizar valores óhmicos comerciales. Se añadirá también un condensador de 100nF en paralelo con el potenciómetro para evitar el posible ruido al desplazar el cursor.

11.1.8. Limitador de tensión

Ya se ha mencionado anteriormente el propósito de este circuito limitador (figura 11.6) no es otro que el de evitar que llegue a la placa Arduino una tensión superior a la que esta última soporta [2].

El manual de usuario del microcontrolador ATmega2560 [19] establece que la máxima tensión de entrada del convertidor AD será la máxima alimentación del propio microcontrolador, es decir $V_{in}(max) = V_{cc}(max) = 6V$. Por tanto, la utilización de un diodo zener de 5,6 V como protección garantiza la seguridad del microcontrolador.

La intensidad que circulará por el diodo zener (figura 11.6) será:

$$Iz = \frac{Vin - Vz}{Rs} \quad (11.20)$$

Si se utiliza un zener de 0,5 W, la intensidad máxima será:

$$P = I \cdot V \quad (11.21)$$
$$I = \frac{P}{V} \quad (11.22)$$
$$I = \frac{0,5W}{5,6V} \quad (11.23)$$
$$I = 89mA \quad (11.24)$$

Esta intensidad máxima se dará cuando Vin, que se corresponde con la salida de las dos etapas de amplificación, sea máxima, es decir, 15 V con ganancia unitaria. Para mayor seguridad, aplicaremos un margen a la tensión de alimentación, por ejemplo, un 20 %, que se corresponde con 18 V. Despejando y sustituyendo en la ecuación 11.20, se obtiene que:

$$89mA = \frac{18V - 5,6V}{Rs} \quad (11.25)$$
$$Rs = 139,3\Omega \quad (11.26)$$

Por tanto, cualquier resistencia mayor de 140 Ω garantiza una corriente inferior a 89 mA. Se utilizará una resistencia Rs de 1 kΩ.

Por otro lado, se indica también en el manual de usuario del ATmega2560 [19] que el convertidor AD está optimizado para señales analógicas cuya resistencia equivalente sea igual o menor a 10 kΩ. Esto significa que habrá que verificar que el equivalente de Thevenin del limitador visto desde la entrada del convertidor AD es menor o igual a 10 kΩ.

Para calcular el equivalente Thevenin del limitador de la figura 11.6, se substituye el diodo zener por su circuito equivalente:

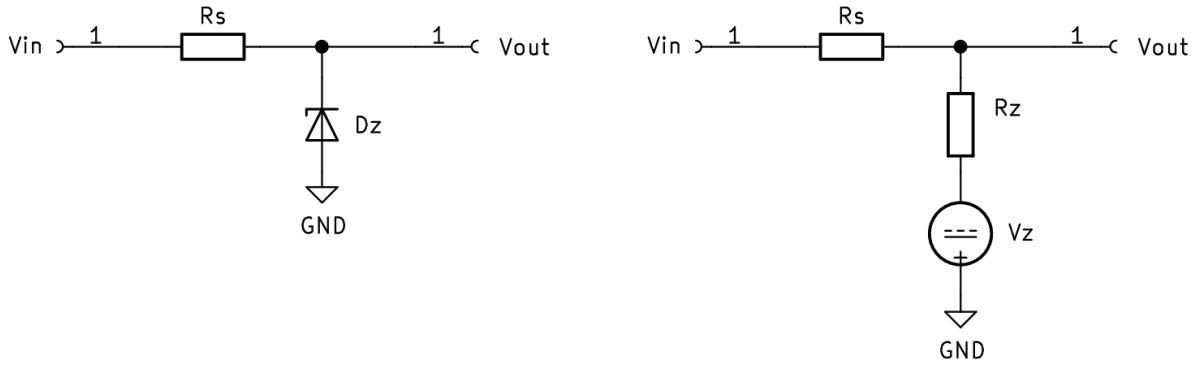


Figura 11.8 – Circuito zener equivalente

El siguiente paso será anular las fuentes de tensión y calcular la impedancia de Thevenin:

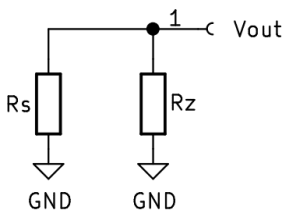


Figura 11.9 – Equivalente Thevenin

Como se observa, la resistencia de Thevenin será el paralelo de la resistencia de polarización del diodo con la resistencia del propio zener: $R_{TH} = Rs||Rz$. Por tanto, teniendo en cuenta que R_S es una resistencia de 1 kΩ y que la resistencia equivalente del diodo zener R_Z es muy pequeña, se puede asegurar que la R_{TH} es considerablemente menor que 10 kΩ, por se cumplen las exigencias para el buen funcionamiento del conversor A/D.

11.1.8.1. Tabla de componentes

Circuito analógico	
Componente	Valor
R1	56 kΩ
R2	100 kΩ
R3	100 kΩ
R4	12 kΩ
Rs	1 kΩ
Rp1	72 Ω
Rp2	47 Ω
Rv1	500 kΩ
Rv2	5 kΩ
Dz	5,6 V
C1	100 nF
C2	100 nF

Tabla 11.2 – Componentes del circuito analógico

11.2. Controlador de relé

Para activar la bobina del relé, se utiliza un transistor NPN, concretamente un BC557B, puesto que se disponía inicialmente. Al tratarse de un transistor tipo NPN, es necesario activarlo a nivel bajo desde Arduino para saturarlo, como se muestra en la siguiente figura 11.10. Además se añade un diodo LED que indicará el funcionamiento del relé, junto con un diodo de libre circulación (DLC) para descargar la bobina y evitar picos de tensión que dañen el transistor.

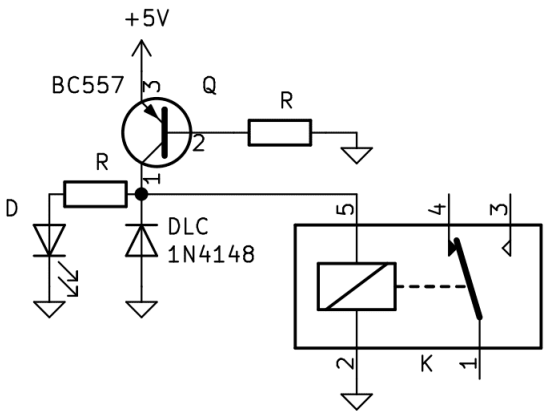


Figura 11.10 – Controlador de relé

Se ha comprobado empíricamente que el relé consume aproximadamente 100mA, intensi-

dad para la que la hoja de características del transistor [20] establece una $\beta = 200$, por tanto:

$$\beta = 200 \tag{11.27}$$
$$I_C = 100mA \tag{11.28}$$
$$I_C = I_B \cdot \beta \tag{11.29}$$
$$I_B = 0,5mA \tag{11.30}$$

Para saturar el transistor, la corriente de base $0,5mA \leq I_B$. Sabiendo esto, puede obtenerse el valor mínimo de la resistencia de base:

$$0,5mA \leq I_B \tag{11.31}$$
$$0,5mA \leq \frac{(5V - 0,7V) - 0}{R} \tag{11.32}$$
$$R \leq 8k6\Omega \tag{11.33}$$

Para el circuito se utilizará una $R = 2k2\Omega$, tanto para la resistencia de base como para la del indicador LED.

11.3. Filtro PWM

Para lograr obtener un nivel de continua a partir del PWM generado por Arduino, se utiliza un filtro RC pasa-bajo. Esta señal será utilizada por el variador de frecuenia en caso de utilizar motores de corriente alterna.

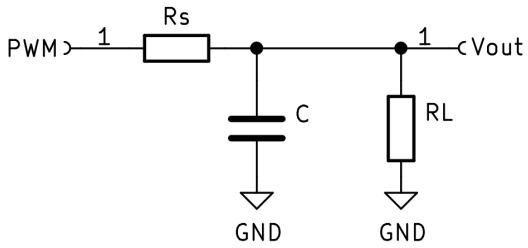


Figura 11.11 – Filtro PWM

Se supondrá una impedancia de entrada del variador de frecuencia de $R_L \geq 100k\Omega$.

Teniendo en cuenta que la tolerancia de las resistencias utilizadas es del 5 %, cualquier resistencia en serie Rs que cumpla que $R_S \leq 20 \cdot R_L$ será despreciable frente la impedancia de entrada R_L [2].

Por tanto, se selecciona una $R_S = 3,9k\Omega \leq 100k\Omega$.

Por otra parte, si se pretende filtrar la pate alterna de la señal PWM y obtener el nivel medio a la salida del filtro, la impedancia del condesador para la frecuencia deseada deberá ser, de igual forma, despreciable frente a la resistencia serie R_L. Por tanto, se calcula que, para la frecuencia de 500 Hz del PWM de Arduino:

$$Z_c \leq \frac{R_S}{20} \tag{11.34}$$

$$\frac{20}{2 \cdot \pi \cdot f \cdot C} \leq R_S \tag{11.35}$$

$$C \geq \frac{20}{2 \cdot \pi \cdot 500 \cdot R_S} \tag{11.36}$$

$$C \geq 1,63\mu F \tag{11.37}$$

11.4. Alimentación del circuito

Con el fin de apagar el ruido eléctrico producido por la alimentación, se han añadido dos condensadores de filtro, tanto en la entrada positiva como en la negativa, como se observa en la figura 11.12.

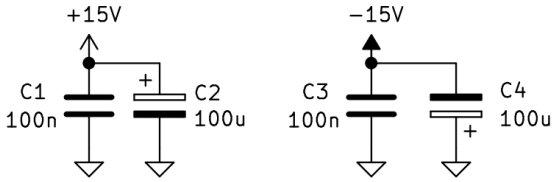


Figura 11.12 – Filtro de entrada de alimentación

Además se han añadido dos indicadores LED en las entradas de alimentación, como se ilustra en la figura 11.13.

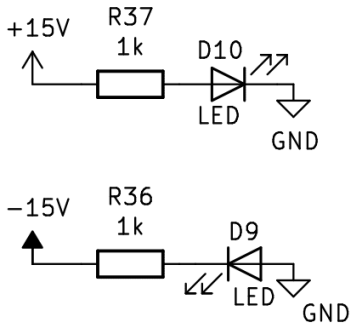


Figura 11.13 – Indicadores LED de alimentación

Es necesario adaptar la tensión positiva de entrada para alimentar la placa de Arduino. La figura 7.7 del capítulo 7 (análisis de soluciones) establece una tensión de entrada de 7V a 12V. Para poder adaptar la tensión de entrada de del circuito analógico (15V) a la entrada de la placa Arduino se colocan 4 diodos 1N4007 en serie. De esta manera se baja el voltaje 2,8V aproximadamente.

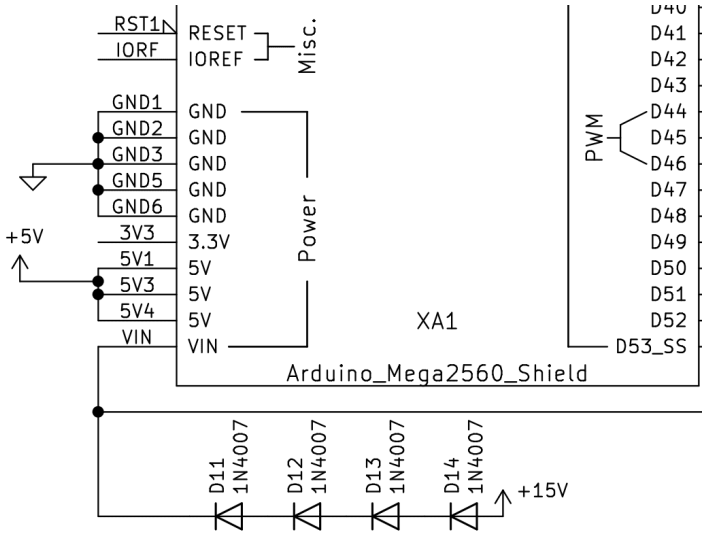


Figura 11.14 – Adaptación de tensiones

Por último, con el objetivo de reforzar la capacidad de corriente del sistema, se añade, en paralelo con el regulador de tensión de entrada de Arduino, un LM7805 [73] (regulador de tensión fijo) junto con un disipador de calor.

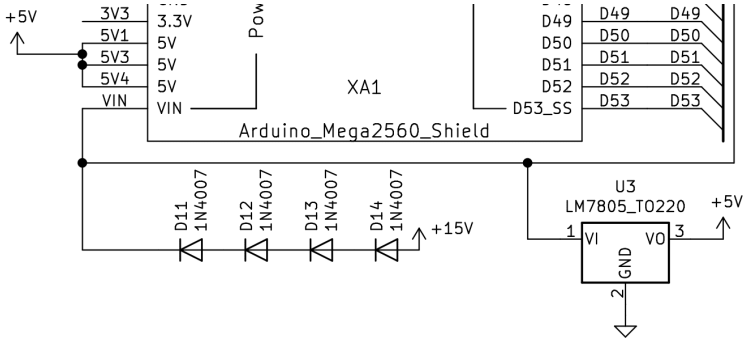


Figura 11.15 – Refuerzo de la capacidad de corriente

12 PROTOCOLO CANOPEN

Se ha indicado en el capitulo 7 (análisis de soluciones) que se utilizará un módulo que servirá de interfaz entre el microcontrolador Arduino y el bus CAN al que se conectarán los encoder. Este módulo está formado por el controlador de bus CAN MCP2515, con interfaz SPI entre el propio controlador y la placa Arduino, así como un transceptor CAN MCP2551, ambos de Microchip:

- MCP2551: Tal como se describe en la hoja de características del integrado [60], se trata de un chip transceptor, que funciona como interfaz ente el controlador de protocolo CAN y el bus físico, funcionando a velocidades de hasta 1 Mb/s. Lo más habitual en un sistema de comunicaciones CAN es que cada dispositivo conectado al bus disponga de un transceptor de estas características, cuya función principal será la de adaptar las señales digitales del controlador CAN de cada nodo, de manera que puedan transmitirse adecuadamente a través del bus físico, controlando los niveles lógicos de tensión.

- MCP2515: Tal como se especifica en la hoja de características [59], el MCP2515 es un microcontrolador que incorpora herramientas como máscaras, filtros, buffers de recepción y transmisión, etc, que facilitan el envío y transmisión de mensajes a través del bus CAN. La lectura y transmisión de los registros de este controlador utiliza el interfaz SPI.

Para lograr la comunicación entre el microcontrolador Arduino y los encoders a través del módulo CAN se ha utilizado la biblioteca SPI.h y la biblioteca CAN_BUS_Shield-master.h [27], respaldada por licencia MIT. Concretamente, de la librería CAN_BUS_Shield-master.h se han utilizado las siguientes funciones:

- CAN.begin: init can and set speed

- CAN.checkReceive: check if got something

- CAN.getCanId(): when receive something, you can get the can id!!

- CAN.readMsgBuf(&len, buf): read message buf

- CAN.sendMsgBuf: send buf

Estas funciones únicamente facilitan el envío y recepción de mensajes a través del bus CAN, sin embargo, se requerirá un estudio del protocolo CANopen para configurar el contenido de estos mensajes y garantizar la comunicación con los encodres. En primer lugar, se realizará una breve introducción al protocolo de comunicación CANopen.

12.1. Modelo OSI

El protocolo de comunicaciones CANopen, desarrollado por CiA (CAN in Automation), y que se acoje a la norma UNE-EN 50325-4, está basado en el protocolo CAN. Este último se ciñe al modelo de referencia OSI de 7 capas.

Se trata de un modelo de referencia para las comunicaciones industriales, que divide la red de comunicaciones en 7 capas independientes. Cada una de estas capas realiza una función perfectamente definida, de manera que se reducirá el flujo de información entre capas.

Figura 12.1 – Modelo OSI [61]

12.2. Protocolo CANopen

El protocolo CAN utiliza únicamente las dos capas inferiores de la pirámide OSI, la capa física y la capa de enlace, es decir, simplemente permite la transmisión de paquetes básicos. Sin embargo, el protocolo CANopen incorpora por encima de la capa física y de enlace la capa de aplicación.

12.2.1. Diccionario de objetos

La adición de esta última capa permite la introducción del concepto de objetos para la organización de la comunicación. Así, existen siete protocolos, implementados en objetos [37,

80]:

- Gestión de red (NMT): la finalidad de este protocolo es la de cambiar el estado de los nodos (inicialización, detención, etc).
- Sincronización (SYNC): la transmisión de un mensaje SYNC desde el nodo maestro provoca la respuesta de uno o varios nodos (dependiendo de la configuración).
- Emergencia (EMCY): se utiliza para notificar al resto de nodos de la red la existencia de un fallo en el nodo emisor del mensaje EMCY.
- Marca de tiempo (Timestamp): relacionado con PDO
- Process Data Object (PDO): se trata de un objeto pensado para optimizar la transmisión de datos de los nodos de la red en tiempo real. Existen dos tipos de objetos PDO:
  - TPDO para la transmisión
  - RPDO para la recepción
- System Data Object (SDO): es utilizado para acceder y cambiar la configuración de los nodos de la red de comunicaciones.
- Monitoreo de nodos (Heartbeat): relacionado con SDO. Se utiliza para verificar la capacidad de comunicación de todos los nodos conectados a la red.

En el diccionario de objetos de cada dispositivo, a cada objeto le corresponde un índice, un subíndice y un rango de valores:

TWK **6. Programming and diagnosis (object directory)**

6.2.13 Object 1800_h - First transmit PDO

Index	Sub	Name	Data type	Access	Range/Value	Default
1800_h	00	Largest supported subindex	Unsigned8	ro	3	
	01	COB-ID	Unsigned32	rw	0 ... 0x7FF	0x180 + Node-ID
	02	Transmission type	Unsigned8	rw	252,253,254	253
	03	Inhibit time	Unsigned16	rw	0 ... 65535	0

Figura 12.2 – Ejemplo de objeto del diccionario CANopen [70]

12.2.2. Modos de comunicación

Por otro lado, CANopen permite utilizar tres modelos de comunicación:

- Maestro/esclavo
- Cliente/servidor
- Productor/consumidor

12.2.3. Trama CANopen

La trama CANopen presenta la configuración que se muestra en la figura 12.3:

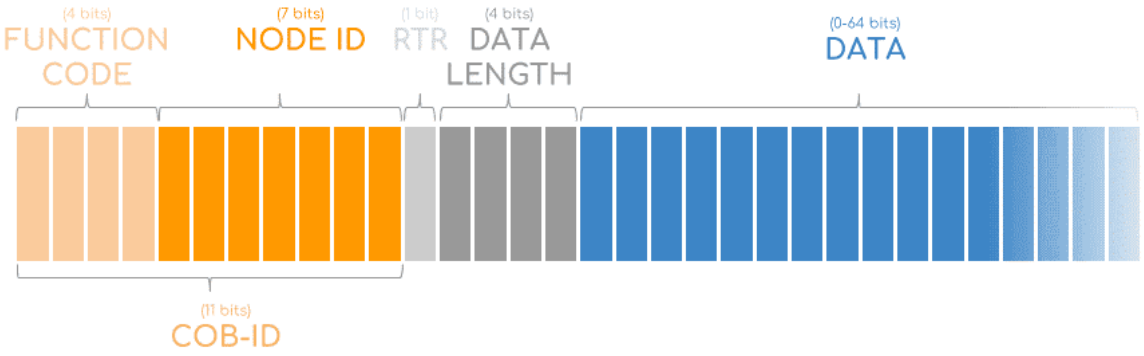


Figura 12.3 – *csselectronics*. CANopen CAN frame [37]

Los primeros 11 bits de la trama conforman el identificador de objeto de comunicación (COB-ID). De estos 11 bits, los 4 primeros indican el código de función, mientras que los 7 últimos representan la ID del nodo. De esta manera, el COB-ID determina qué nodo está emitiendo o recibiendo el mensaje y a través de qué objeto.

El octavo bit de la trama es el bit RTR (Remote Transmission Request), que se utiliza en transmisiones de objetos PDO de manera asíncrona. Más adelante se profundizará en este aspecto. Por último, los siguientes 4 bits indican la longitud del mensaje que se transmitirá o recibirá, entre 0 y 64 bits.

Para realizar las comunicaciones mediante cualquiera de los protocolos de comunicación de CANopen, existen diccionarios de objetos propios de cada dispositivo con una estructura estandarizada, que definen todos los parámetros del funcionamiento de esos nodos en la red. Así, dependiendo del tipo de comunicación que queramos realizar, utilizaremos un protocolo de comunicación CANopen distinto.

A cada protocolo se le asocia un objeto de comunicación, y a cada objeto, un código de función. Si recordamos el formato del COB-ID, los primeros 4 bits de este definían el código de función, es decir, el objeto de comunicación. En la figura 12.4 se muestra la lista de códigos de función asociados a cada objeto del diccionario. Esta tabla será de gran utilidad más adelante, a la hora de configurar los diálogos con los encoders.

	COMMUNICATION OBJECT	FUNCTION CODE (4 bit, bin)	NODE IDs (7 bit, bin)	COB-IDs (hex)	COB-IDs (dec)	#
1	NMT	0000	0000000	0	0	1
2	SYNC	0001	0000000	80	128	1
3	EMCY	0001	0000001-1111111	81 - FF	129 - 255	127
4	TIME	0010	0000000	100	256	1
5	Transmit PDO 1	0011	0000001-1111111	181 - 1FF	385 - 511	127
	Receive PDO 1	0100	0000001-1111111	201 - 27F	513 - 639	127
	Transmit PDO 2	0101	0000001-1111111	281 - 2FF	641 - 767	127
	Receive PDO 2	0110	0000001-1111111	301 - 37F	769 - 895	127
	Transmit PDO 3	0110	0000001-1111111	381 - 3FF	897 - 1023	127
	Receive PDO 3	1000	0000001-1111111	401 - 47F	1025 - 1151	127
	Transmit PDO 4	1001	0000001-1111111	481 - 4FF	1153 - 1279	127
	Receive PDO 4	1110	0000001-1111111	501 - 57F	1281 - 1407	127
6	Transmit SDO	1011	0000001-1111111	581 - 5FF	1409 - 1535	127
	Receive SDO	1100	0000001-1111111	601 - 67F	1537 - 1693	127
7	HEARTBEAT	1110	0000001-1111111	701 - 77F	1793 - 1919	127

Figura 12.4 – *csselectronics.* Function code [37]

12.3. Comunicación con los encoders

En este apartado se describirá el proceso seguido hasta lograr la comunicación con los encoders. El primer paso consistió en realizar una comunicación simple con un único encoder conectado al bus CAN, siguiendo los pasos indicados en el manual de usuario del encoder [70].

7.1 Boot-up

The following Table shows encoder boot-up, from switching on the supply voltage to initial transmission of the position value. The position value is subsequently polled via a Sync command.

Action	Id	Rx/Tx	DLC	Databytes								Remark
				00	01	02	03	04	05	06	07	
Bus active, encoder in the bus with node address 1												
Voltage off -> on	701	Rx	1	00								Boot up node 1
Start all nodes	0	Tx	2	1	0							Operational for all nodes
	181	Rx	2	xx LSB	xx MSB							Response from TBN (PDO1)
Master (user) transmits a Sync												
Sync from the master	80	Tx	0									
	281	Rx	2	xx LSB	xx MSB							Response from TBN (PDO2)

All values in hex!

Figura 12.5 – Ejemplo de boot-up [70]

Esta comunicación fue establecida mediante el envío desde Arduino de un objeto SYNC, al cual el encoder responde enviando un PDO con la posición del mismo.

Puede verse que en el arranque, el encoder transmite un mensaje con un objeto HEARTBEAT con COB-ID 701 (tabla de la figura 12.4). A continuación se transmite desde Arduino un mensaje de inicialización de nodos, al que el encoder responde a través del PDO1, en el cual ya va incluida la posición. Una vez arrancado, el encoder envía un PDO2 con la posición desde la Id 281 (COB-ID 280+ ID del nodo) como respuesta al mensaje SYNC (COB-ID 80) trasmitido desde Arduino.

El código puede consultarse en el anexo 18.1.

En el monitor serie del IDE de Arduino se pueden ver las respuestas del encoder tal como se indica en el manual de usuario [70]:

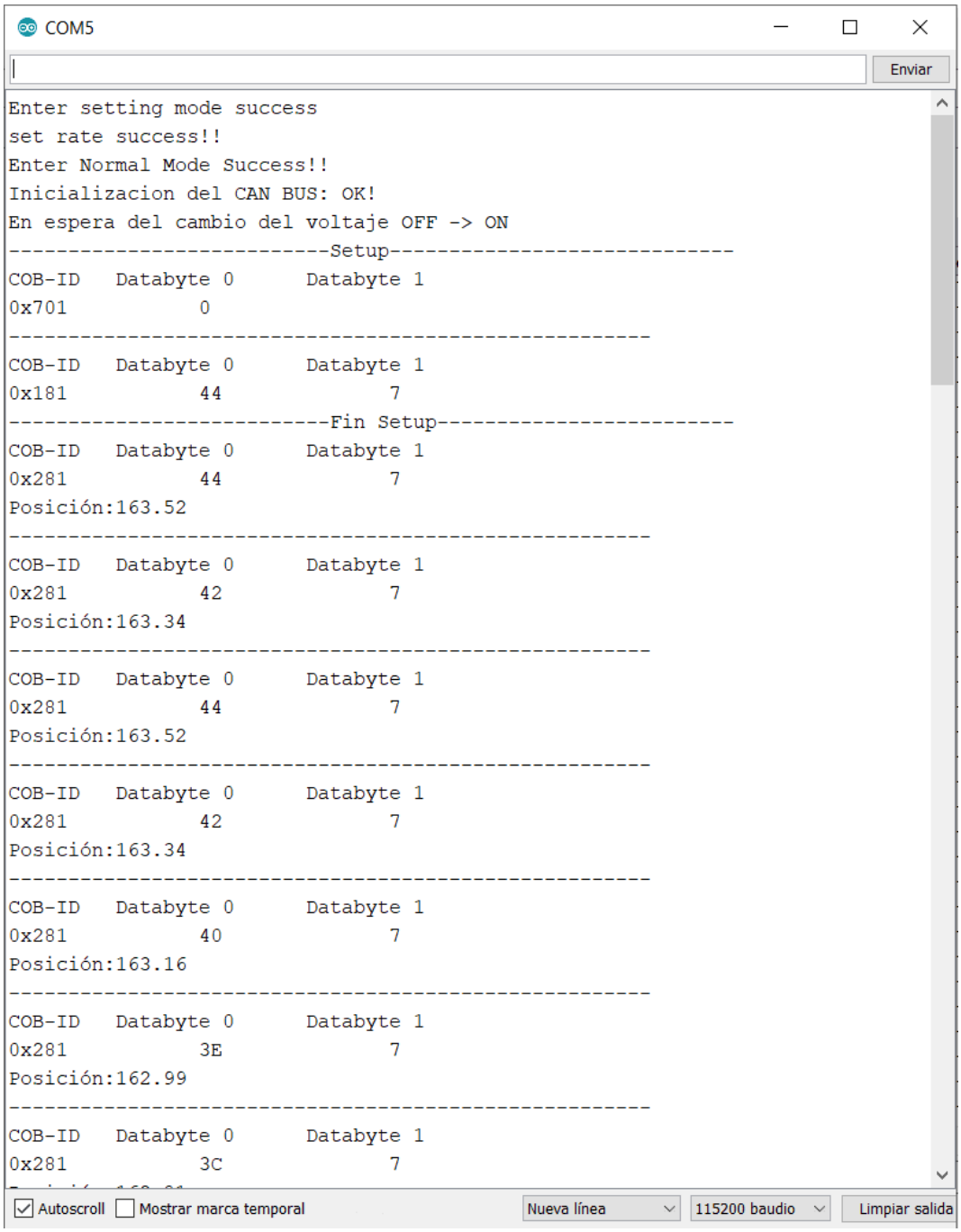


Figura 12.6 – Funcionamiento mediante mensajes SYNC

El siguiente paso fue realizar el mismo procedimiento conectando los dos encoders, pero dado que, por defecto, estos vienen configurados con la dirección Id 1, fue necesario modificar la dirección de uno de ellos. Para ello se han seguido los pasos indicados en el manual de usuario, desde el modo de configuración LSS (Layer Setting Services).

Aktion	Id	Rx/Tx	DLC	Databytes								Comment
				00	01	02	03	04	05	06	07	
Stop Node	0	Tx	2	02	00							Stop node for all nodes
LSS-Switch Mode Selective	7E5	Tx	8	40	0D	01	00	00	00	00	00	1. Transmission of the manufacturer name
LSS-Switch Mode Selective	7E5	Tx	8	41	00	60	00	00	00	00	00	2. Transmission of the product number
LSS-Switch Mode Selective	7E5	Tx	8	42	03	00	01	00	00	00	00	3. Transmission of the revision number
LSS-Switch Mode Selective	7E5	Tx	8	43	66	BE	02	00	00	00	00	4. Transmission of the serial number (in this case: 179814)
	7E4	Rx	8	44	00	00	00	00	00	00	00	Success message from the sensor, which is now in LSS Configuration-Mode
LSS-Configure Modul ID	7E5	Tx	8	11	02	00	00	00	00	00	00	Node address 2 programming
	7E4	Rx	8	11	00	00	00	00	00	00	00	Success message from the sensor
LSS-Store Configuration	7E5	Tx	8	17	00	00	00	00	00	00	00	Zero-voltage-protected saving
	7E4	Rx	8	17	00	00	00	00	00	00	00	Success message from the sensor
LSS-Switch Mode Global: Operation Mode	7E5	Tx	8	04	00	00	00	00	00	00	00	Sensor is reset to LSS-Operation-Mode
	702	Rx	1	00								Boot-up node with new node address

All values in hex!

Figura 12.7 – Cambio de dirección vía LSS [70]

La tabla de la figura 12.7 muestra los mensajes emitidos y recibidos por el encoder necesarios para modificar la Id a la número dos. En el código confeccionado 18.2 se replica este mismo diálogo, al que se le ha añadido los dos primeros pasos realizados en apartado anterior, es decir, espera del mensaje transmitido por el encoder en la transición de 0 V a VCC y, posteriormente, inicialización del nodo, a la que el encoder responde con el objeto PDO1.

Además, es necesario modificar los datos propios del encoder (número de serie, número de producto y número de revisión). Estos los podemos encontrar en la hoja de características del producto y en la etiqueta del mismo.

Cabe destacar que ha sido imposible encontrar la hoja de características en la web del fabricante, por lo que ha sido necesario ponerse en contacto con el servicio de atención al cliente de la propia empresa y solicitar la hoja de características, que se adjunta en el anexo 20.

Lieferform

Wenn nicht anders vereinbart, werden die Winkelcodierer mit folgenden Werten eingestellt geliefert:

- Knoten-Nummer: 1
  Einstellbar über LSS oder Objekt 2000
- Baudrate: 20 kBaud

PDO 1:

Index	Sub-Index	Kommentar	Vorgabewert
1800h	1	COB-ID für PDO 1	180h + Knotenadresse
	2	transmission type (asynchron, herstellerdefiniert)	253
	3	inhibit time	0

PDO 2:

Index	Sub-Index	Kommentar	Vorgabewert
1801h	1	COB-ID für PDO 2	280h + Knotenadresse
	2	transmission type (zyklisch synchron)	1

SDO:

SDO (tx) (Codierer -> Master): 580h (1408) + Knotenadresse
SDO (rx) (Master -> Codierer): 600h (1536) + Knotenadresse
Hinweis: Die SDO-Identifier sind nicht änderbar

LSS-Adresse (siehe Identity-Objekt 1018):
Vendor ID = 0x0000 010D
Produkt ID = 0x0000 06000
Revisionsnummer: = 0x0001 0003
Seriennummer = siehe Typenschild

Bemerkung: LSS = Layer Setting Service DSP 305 V1.1.1

Programmed values as supplied

Unless agreed otherwise, the encoders are supplied with the following adjusted values

- *Node number: 1*
  Adjustable via LSS or object 2000
- *Baud rate: 20 kBaud*

PDO1:

Index	*Sub-Index*	*Comment*	*Default value*
1800h	*1*	*COB-ID for PDO 1*	*180h + Node-ID*
	2	*transmission type (asynchronous, manufacturer defined)*	*253*
	3	*inhibit time*	*0*

PDO 2:

Index	*Sub-Index*	*Comment*	*Default value*
1801h	*1*	*COB-ID for PDO 2*	*280h + Node-ID*
	2	*transmission type (cyclic synchronous)*	*1*

SDO:

SDO (tx) (Encoder -> Master): 580h (1408) + Node-ID
SDO (rx) (Master -> Encoder): 600h (1536) + Node-ID
Hint: SDO-identifiers are not changeable

LSS-adress (see identity-object 1018):
Vendor ID = 0x0000 010D
Product ID = 0x0000 6000
Revision number = 0x0001 0003
Serial number = see nameplate

Note: LSS = Layer setting service DSP 305 V1.1.1

Figura 12.8 – Datos del encoder [20]

Figura 12.9 – Número de serie

Dado que los datos se transmiten en código hexadecimal, el número de serie correspondiente al encoder será:

- S/N: 459764 = 0x0703F4

El código puede consultarse en el anexo 18.2.

En el monitor serie se verifica el cambio de dirección ID del nodo:

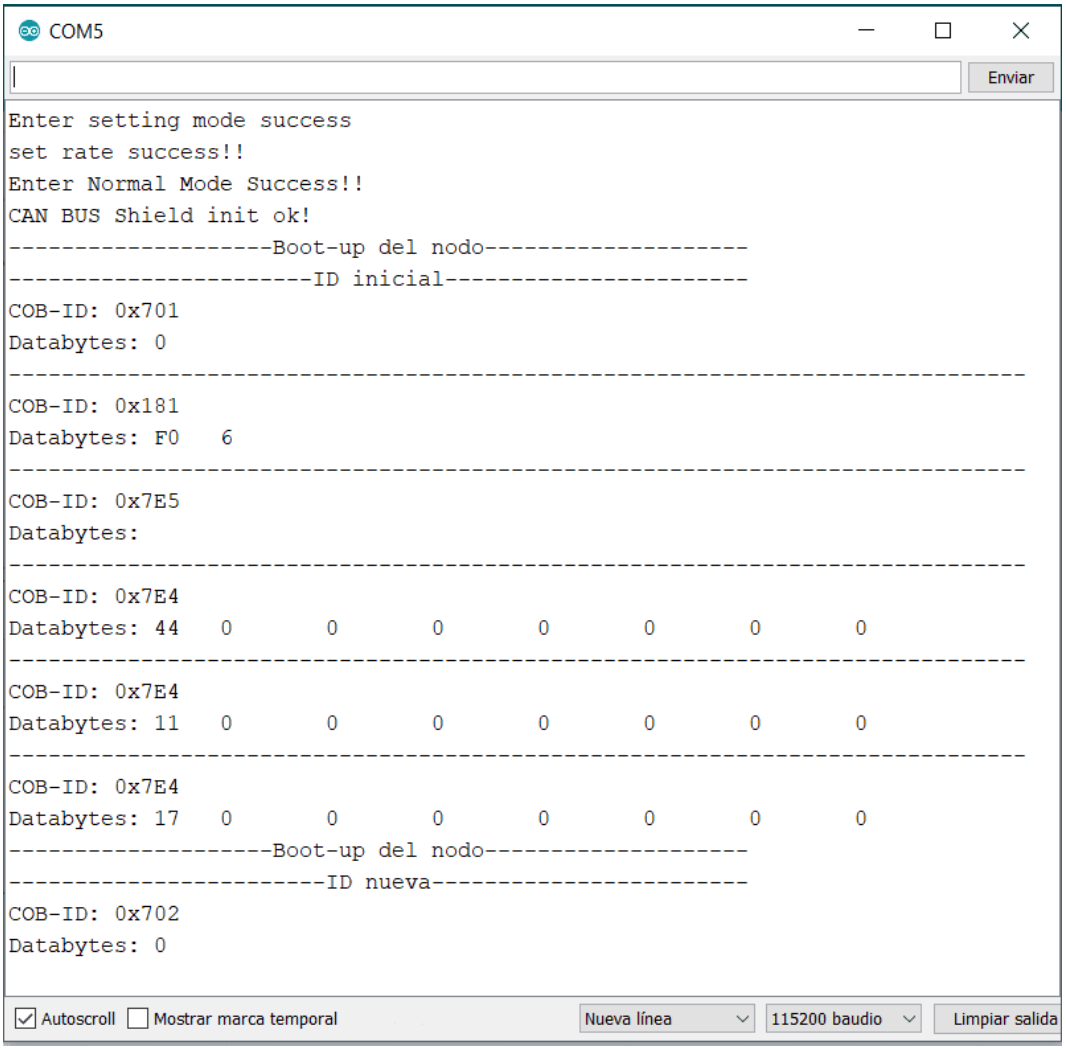


Figura 12.10 – Verificación del cambio de ID

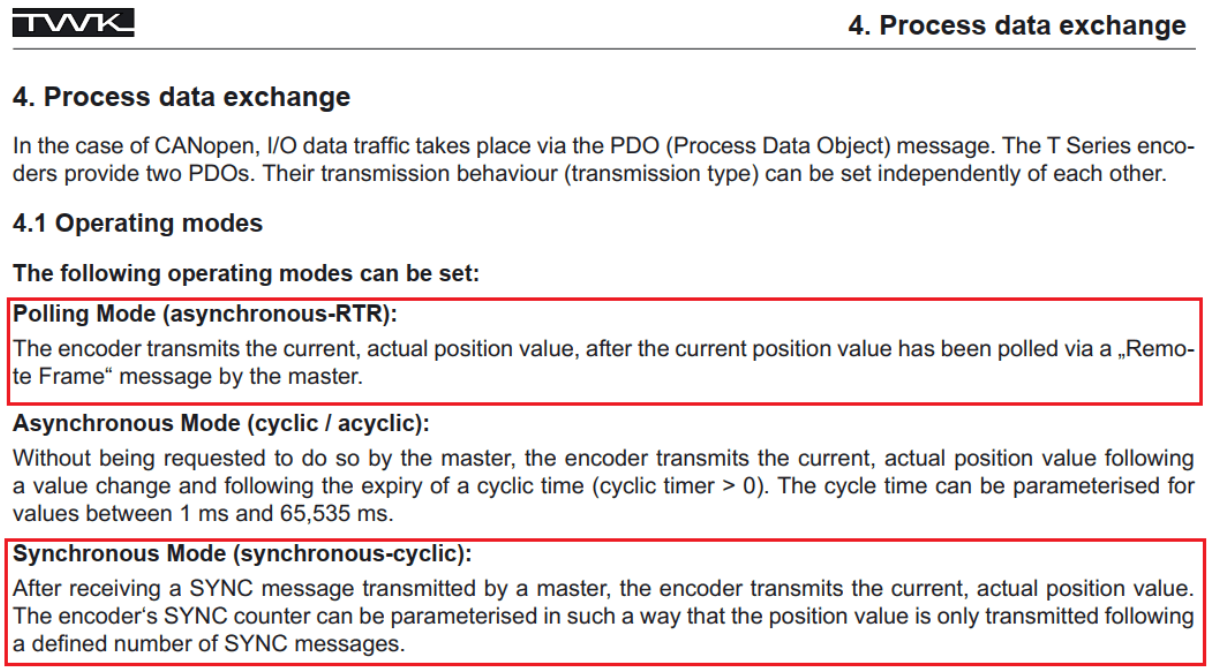
12.3.1. Comunicación simultánea con los dos encoder

Una vez cambiada la ID de uno de los dos encoders, podemos conectarlos al bus. En el manual de usuario [70] se especifican distintos modos de funcionamiento, de los cuales podrían adaptarse a las necesidades del proyecto los dos siguientes: Polling Mode (asynchronous-RTR) y Synchronous Mode (synchronous-cyclic).

El modo síncrono (cíclico) ofrece un contador de mensajes SYNC, de manera que se puede configurar para que sólo transmita el mensaje PDO con la posición después de un número de mensajes SYNC establecidos por el usuario, mientras que el modo asíncrono (RTR), el encoder transmite la posición al recibir una solicitud del maestro. Este último modo de funcionamiento permite direccionar la solicitud del maestro a cada nodo del bus, mientras que en el modo síncrono-cíclico el mensaje SYNC se envía a todos los nodos, y cada uno de ellos res-

ponde con el envío de la posición únicamente cuando el contador de mensajes SYNC alcance un número de mensajes determinado por el usuario.

Después de realizar varias pruebas, el método más cómodo y que mejor se adapta al proyecto es el modo polling (RTR).



TWK 4. Process data exchange

4. Process data exchange

In the case of CANopen, I/O data traffic takes place via the PDO (Process Data Object) message. The T Series encoders provide two PDOs. Their transmission behaviour (transmission type) can be set independently of each other.

4.1 Operating modes

The following operating modes can be set:

Polling Mode (asynchronous-RTR):
The encoder transmits the current, actual position value, after the current position value has been polled via a „Remote Frame“ message by the master.

Asynchronous Mode (cyclic / acyclic):
Without being requested to do so by the master, the encoder transmits the current, actual position value following a value change and following the expiry of a cyclic time (cyclic timer > 0). The cycle time can be parameterised for values between 1 ms and 65,535 ms.

Synchronous Mode (synchronous-cyclic):
After receiving a SYNC message transmitted by a master, the encoder transmits the current, actual position value. The encoder‘s SYNC counter can be parameterised in such a way that the position value is only transmitted following a defined number of SYNC messages.

Figura 12.11 – Modos de transmisión [70]

Se ha indicado anteriormente que existen dos tipos de objeto PDO:

- TPDO: se utiliza para transmitir datos al nodo
- RPDO: se utiliza para recibir datos del nodo

Para hacer una solicitud a un determinado dispositivo desde el maestro, se envía un TPDO vacío con el bit RTR activado [80]. Para poder realizar este tipo de transmisiones asíncronas con solicitud RTR, será necesario configurar los dispositivos con este tipo de transmisión. En la tabla del apartado 4.1 del manual de usuario [70] se indican los códigos de transmisión de los distintos modos de transmisión :

Transmission Type					
Code	Transmission type				
	Cyclic	Acyclic	Synchron	Asynchronous	RTR
0		x	x		
1-240	x		x		
241-251	Reser				
252			x		x
253				x	x
254				x	
Meaning					
0	After SYNC, but only if the value has changed since the last SYNC.				
1-240	Transmit value after 1st or 240th SYNC message.				
252	Cycle Timer = 0	Position integration on SYNC; output of the stored position following request (Remote Frame).			
	Cycle Timer ≠ 0	Current position is transmitted in the timer's cycle. Position integration on SYNC; output of the stored position following request (Remote Frame) remains active.			
253	Cycle Timer = 0	Current position is transmitted upon request (Remote Frame).			
	Cycle Timer ≠ 0	Current position is transmitted in the timer's cycle. Current position is also transmitted following request (Remote Frame).			
254	Cycle Timer = 0	Data output occurs in the event of a position change. Current position is also transmitted following request (Remote Frame).			
	Cycle Timer ≠ 0	Current position is transmitted in the timer's cycle. Data output also occurs in the event of a position change. Current position is also transmitted following request (Remote Frame).			

Figura 12.12 – Códigos de los modos de transmisión [70]

En la tabla de la figura 12.12 se observa que el código del modo polling es el 253. Por lo tanto, habrá que configurar los dispositivos con este código para establecer este modo de transmisión. Sin embargo, más adelante, la tabla del objeto 1800h del diccionario (PDO1) indica que el modo de transmisión por defecto de este objeto es precisamente el correspondiente al código 253, de manera que no será necesaria la configuración del dispositivo.

6.2.13 Object 1800_h - First transmit PDO

Index	Sub	Name	Data type	Access	Range/Value	Default
1800_h	00	Largest supported subindex	Unsigned8	ro	3	
	01	COB-ID	Unsigned32	rw	0 ... 0x7FF	0x180 + Node-ID
	02	Transmission type	Unsigned8	rw	252,253,254	253
	03	Inhibit time	Unsigned16	rw	0 ... 65535	0

Object 1800_h defines the first PDO's communication data. Only transmission types 252,253,254 are supported.

Sub-index 01 (COB ID) contains the identifier for PDO1.

In default status, this has the value 0x180 + node address. If the object is written, the node address is no longer added. The default status can be restored via „load default" (object 1011_h).

The inhibit time (ms) is the time before the PDO is permitted to be transmitted again.

(See operating modes in Chapter 4.1)

Figura 12.13 – Objeto PDO1

Por tanto, para obtener la posición de cada encoder bastará con enviar un TPDO1 (First transmit PDO) sin datos y con el bit RTR activado. El COB-ID correspondiente se obtiene de la tabla de la figura 12.4:

- Código de función: 00112
- ID del nodo: 00000012 - 00000112

El COB-ID correspondiente en hexadecimal será entonces 0x180+ ID, es decir, 0x181 o 0x182.

En el monitor serie se visualiza el encío de datos de los encoders:

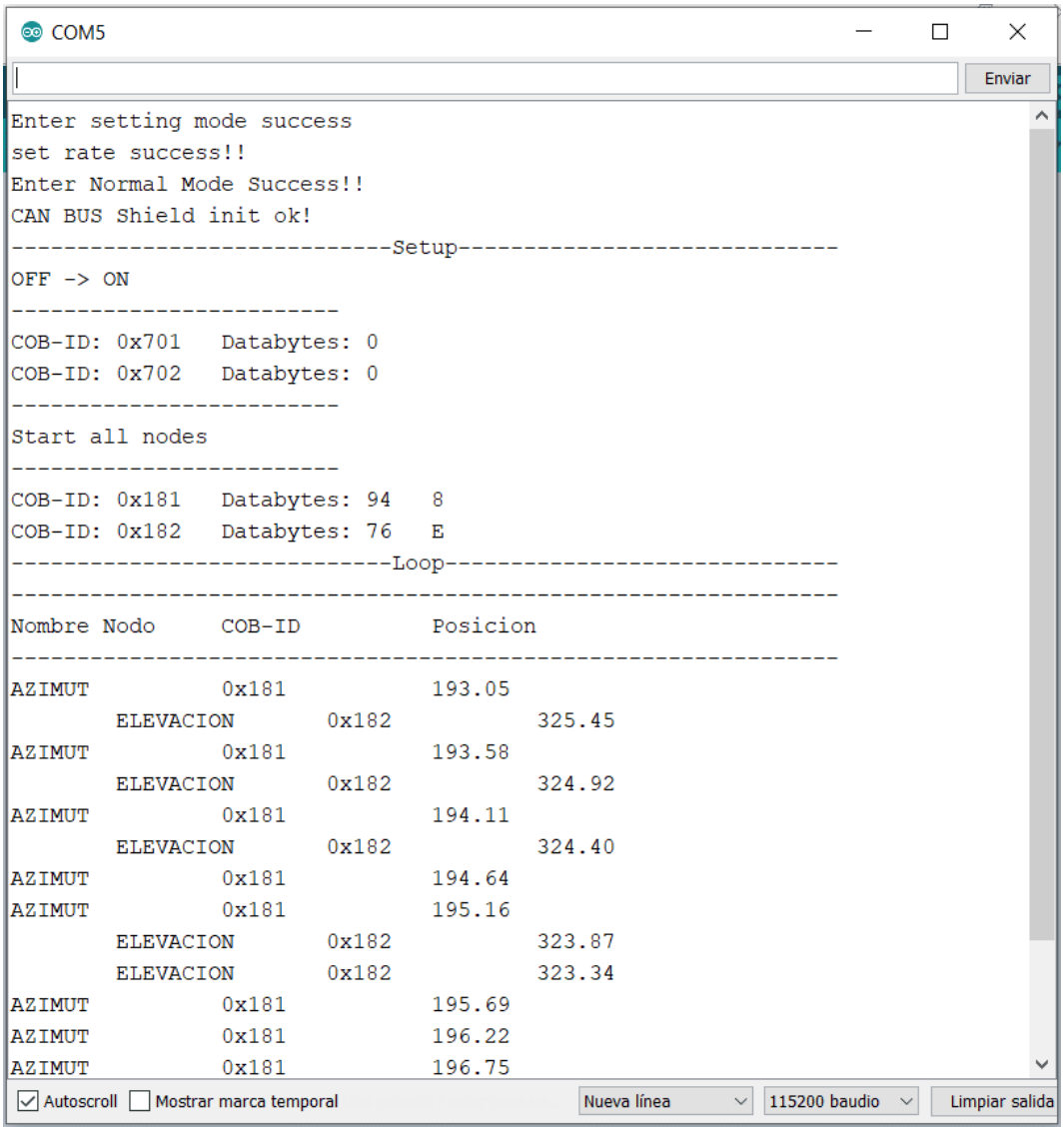


Figura 12.14 – Funcionamiento en modo Polling (RTR activado)

Este resultado se obtiene a partir del código del programa principal referente al protocolo CANopen, adjunto en el anexo 17.

13 RECEPTOR GPS

En este anexo se describe el proceso de decodificación de la información procedente del receptor GPS NEO6M.

13.1. Estándar de comunicaciones NMEA

El formato NMEA (National Marine Electronic Asociation) es un estándar de comunicaciones, utilizado originalmente para establecer la comunicación entre equipos de navegación. Como ya se ha indicado en la sección 7.1.2 , se utiliza el módulo NEO 6M para realizar la lectura GPS. Este se comunica con la placa Arduino a través de puerto serie, por lo que se utilizará la biblioteca SoftwareSerial.h, incluida en el propio IDE Arduino.

Con este pequeño programa, que se adjunta en el anexo 18.3, se visualiza en el puerto serie la información transmitida por el receptor GPS.

Tal como se aprecia en la figura 13.1, el receptor que se utiliza transmite la información en distintos formatos.

El resultado que se visualiza a través del monitor serie del IDE de Arduino es el siguiente:

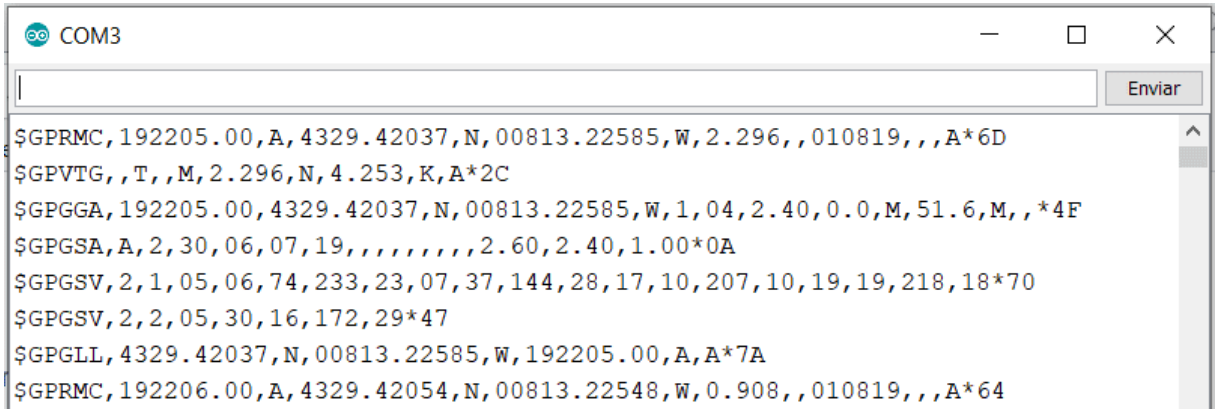


Figura 13.1 – Datos serie del receptor GPS

Dentro del estándar NMEA existen distintos formatos, aunque utilizaremos el GPRMC, ya que es el más utilizado para este tipo de aplicaciones. Presenta la siguiente estructura [57, 49]:

$GPRMC,hhmmss.ss,A,lll l.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh

Cada uno de los parámetros que se transmiten dentro de la trama va separado por comas. Así, en primer lugar, nos encontramos con el formato de la trama, GPRMC.

- hhmmss.ss: Hora UTC (Coordinated Universal Time)
- A: Estado de validez (A=válido, V=inválido)
- llll.ll: longitud

- a: norte(N)/sur(S)
- yyyyy.yy: latitud
- a: este(E)/oeste(W)
- x.x: velocidad en nudos
- x.x: curso en grados
- ddmmyy: fecha UT (Universal Time)
- x.x,a: variación magnética en grados
- a: este(E)/oesete(W)
- *hh: Checksum

13.2. Máquina de estados

Para decodificar esta trama se ha optado por implementar una máquina de estados [48], de manera que con cada carácter recibido por puerto serie, se comprueba si concuerda con el formato seleccionado y se guarda en la variable correspondiente cuando proceda:

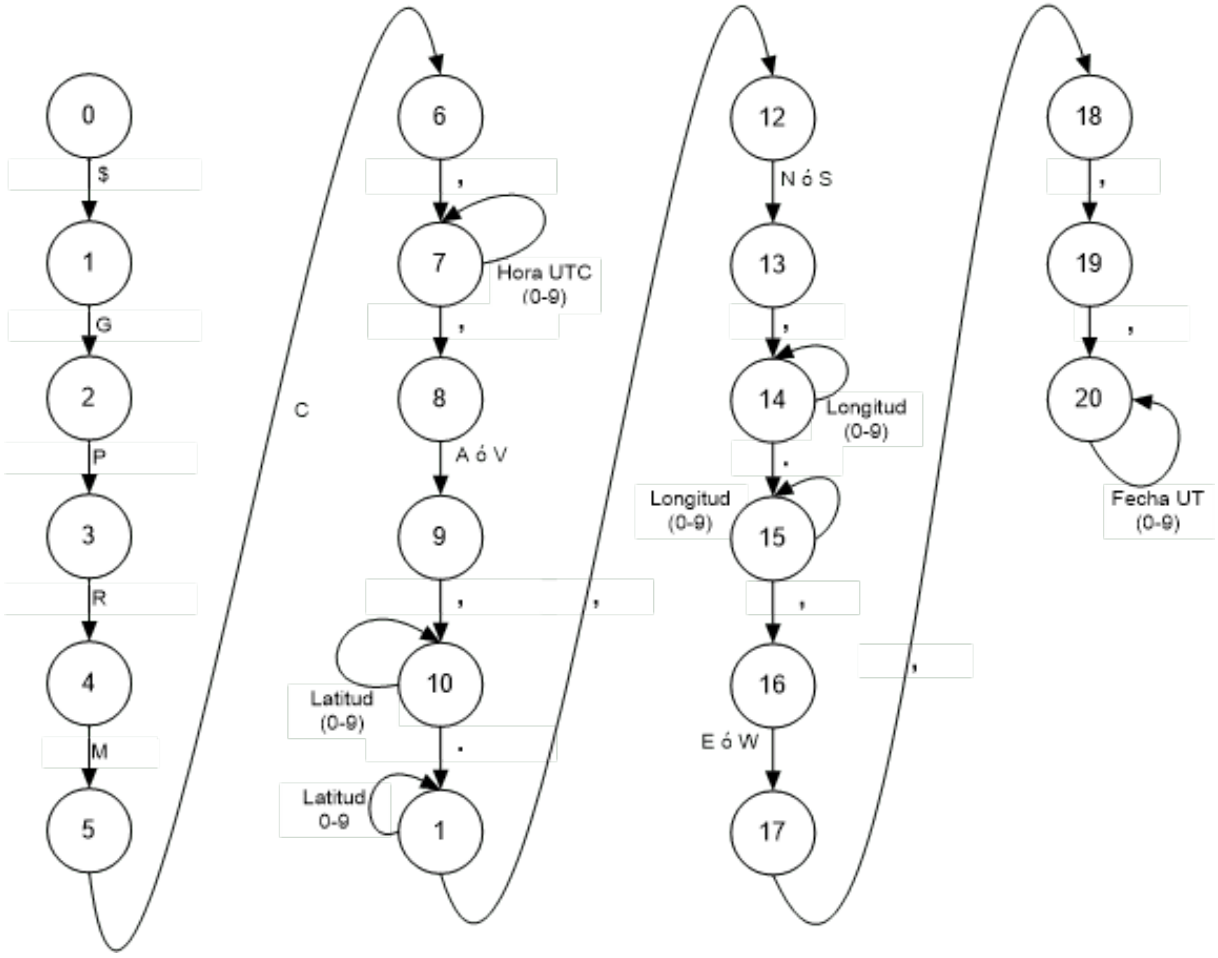


Figura 13.2 – Máquina de estados GPS

En cada estado, en caso de que el carácter recibido no se corresponda con el que establece el formato GPRMC, se devuelve la máquina de estados al estado inicial (en el diagrama no se representa la devolución en cada estado al estado inicial para mayor simplicidad visual). De esta manera se garantiza que sólo se decodificará completamente la trama si esta es la correcta. Esto quiere decir que no hay posibilidad de confusión con otro formato, la máquina de estados decodificará la trama completa únicamente cuando esta sea en el formato indicado.

En el monitor serie se verifica la decodificación:

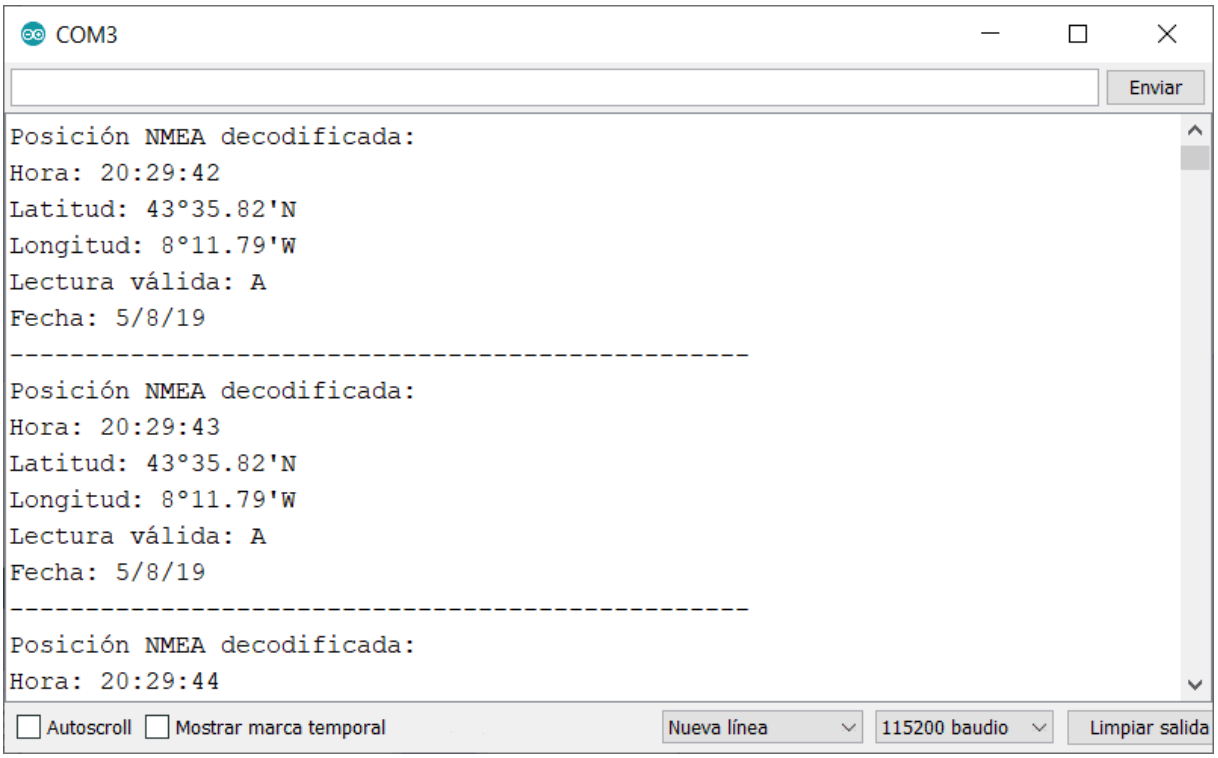


Figura 13.3 – Trama NMEA decodificada

El código con el que se ha obtenido este resultado se encuentra en el programa principal (anexo 18) en las funciones referentes a la lectura GPS. En él se realizaron las modificaciones necesarias para poder visualizar en el monitor serie los resultados.

14 TRANSMISIÓN DE LA POSICIÓN DEL SATÉLITE POR SOFTWARE EXTERNO

En el capítulo 7 de análisis de soluciones se establece que para este proyecto se utilizará el software PstRotator para transmitir al microcontrolador los parámetros de posición del satélite a seguir. Se indica también que esta transmisión se realizará en formato Easycom. Una vez instalado el software en el equipo, se procede a configurarlo.

14.1. Guía de configuración del software PstRotator

En primer lugar, debemos establecer nuestra ubicación en coordenadas de latitud y longitud, clicando en Setup - My Location:

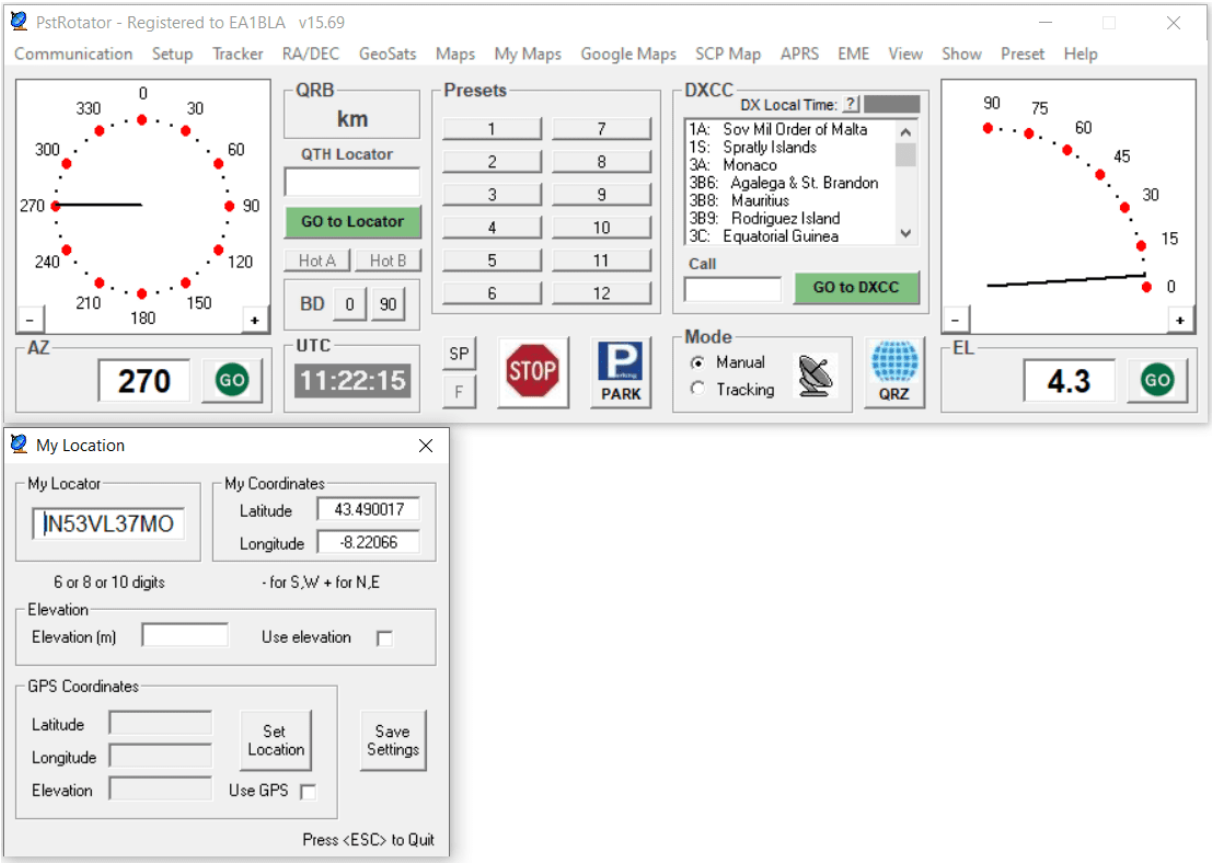


Figura 14.1 – Configuración de la ubicación

A continuación, al clicar en la pestaña Setup - Satellites Tracking se observa que aparece una nueva ventana del programa con el nombre de Satellites Tracking:

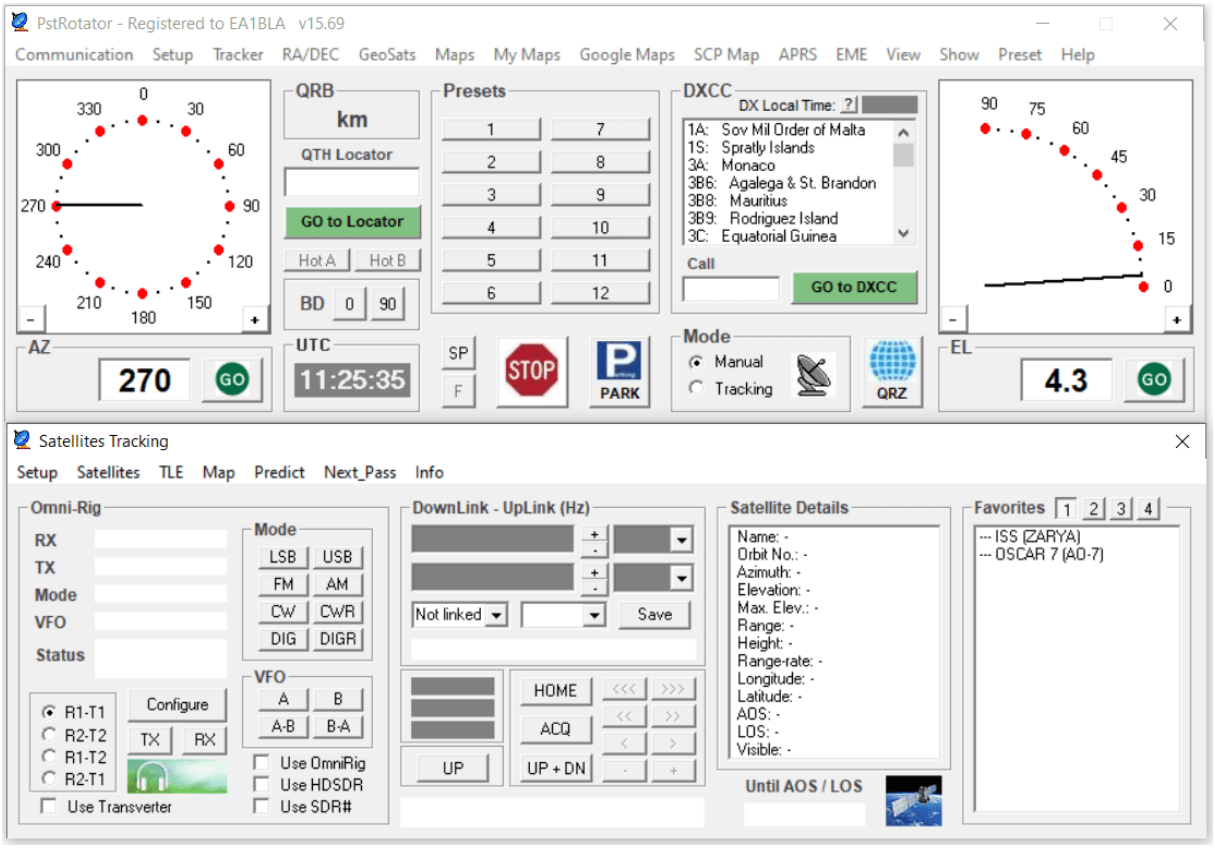


Figura 14.2 – Configuración de la ventana Satellites Tracking

Esta nueva ventana del programa permite, entre otras opciones, seleccionar los satélites de una determinada base de datos. Para ello, basta con clicar sobre la pestaña Satellites, seleccionar el satélite que se desea seguir, y añadirlo a la lista:

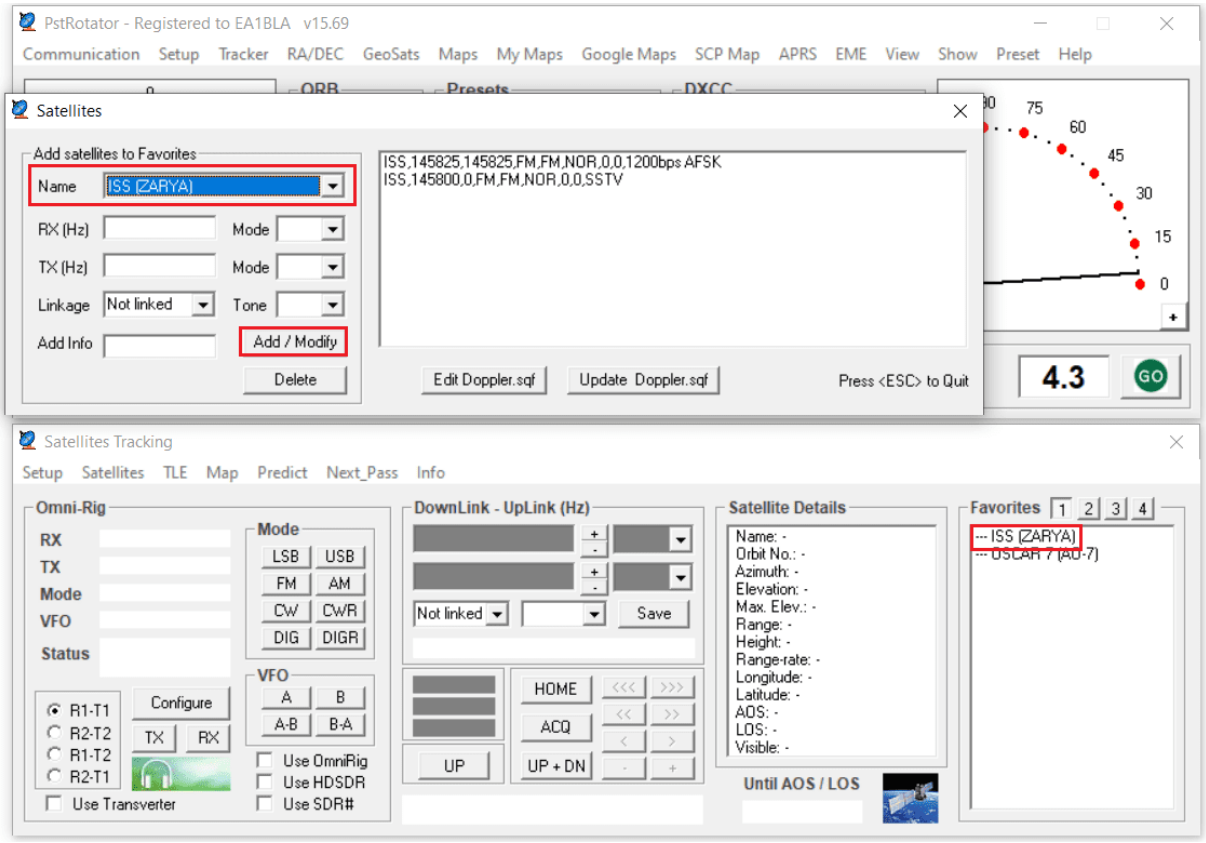


Figura 14.3 – Configuración de la lista de satélites

Es muy importante también mantener los datos TLE actualizados para que la posición de apuntamiento proporcionada por el programa sea correcta. Esto se logra en la pestaña TLE de la segunda ventana del PstRotator. En la figura 14.4 se indica cómo se deben actualizar estos parámetros TLE.

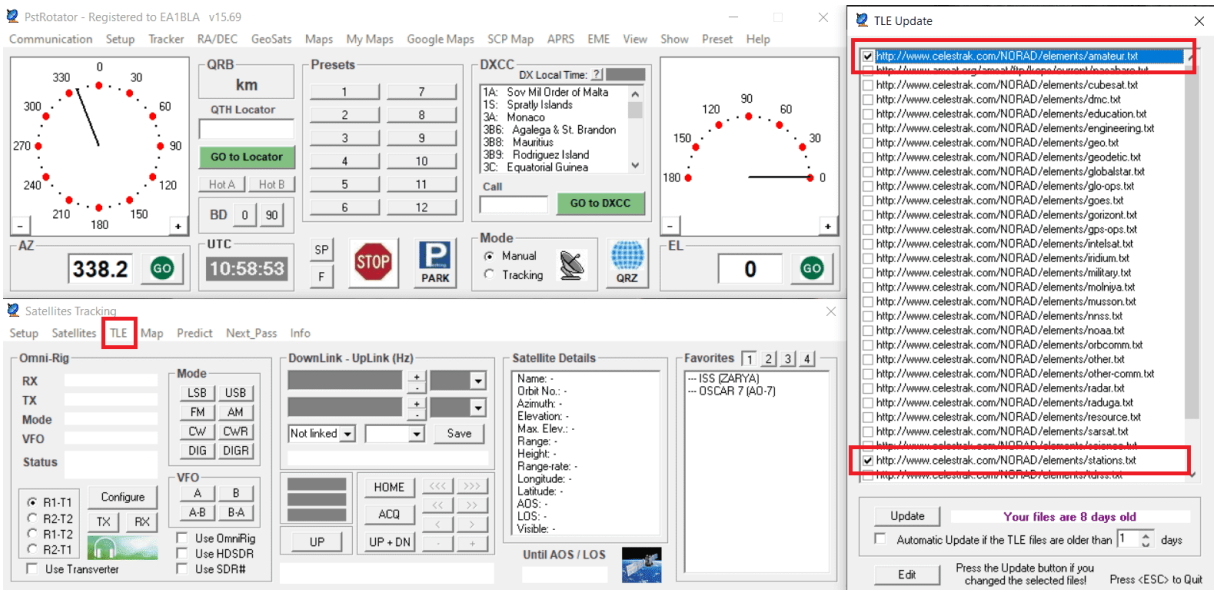
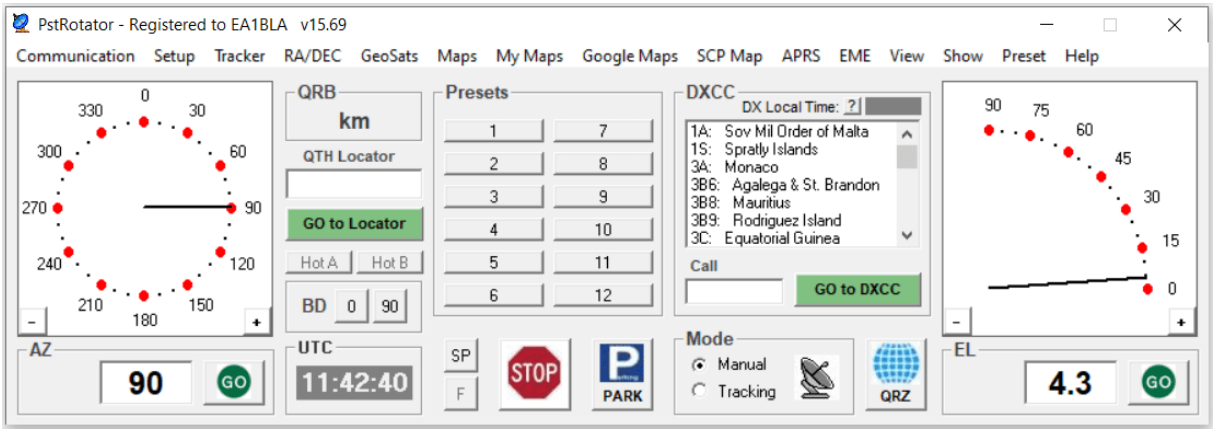


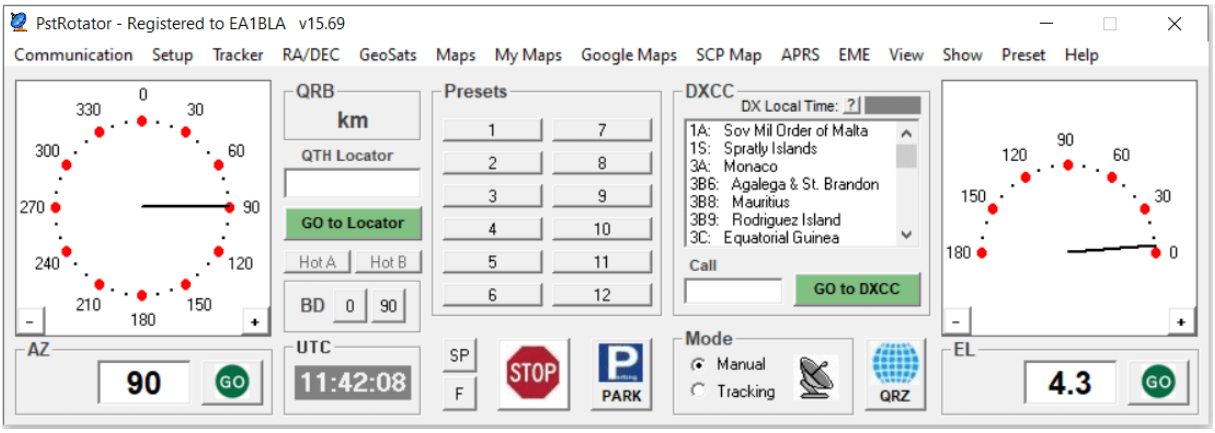
Figura 14.4 – Actualización TLE en PstRotator

Estos datos TLE, como ya se ha mencionado en la sección 7.1.2, contienen la información orbital del satélite necesaria para su seguimiento. El análisis de estos datos TLE se encuentra en el anexo 15.

De nuevo en la ventana principal del programa, debe establecerse el formato de transmisión de los datos de posición del satélite. Esta opción se encuentra en la pestaña Setup - EL/AZ + EL Controller. Dentro de esta pestaña, tal como se ha indicado en el capítulo de análisis de soluciones 7, se seleccionará la opción K3NG (Easycom). Cabe destacar que, dependiendo del sistema implementado, el ángulo de elevación puede indicarse en un rango de 0º a 90º, o bien de 0º a 180º. Este último rango es el que se ha seleccionado para este proyecto, por lo tanto, se seleccionará esta opción en la pestaña Setup - 180 deg Elevation. Se observa que el indicador gráfico de la pestaña principal del programa correspondiente al ángulo de elevación se modifica:



(a) Rango de elevación de 0º a 90º



(b) Rango de elevación de 0º a 180º

Figura 14.5 – Configuración del rango del ángulo de elevación

El último paso es comenzar la transmisión por puerto serie, se selecciona el modo Tracking en lugar del modo Manual, se selecciona el satélite que se desea seguir y, en la ventana principal del programa, se hace clic sobre Communication - EL/AZ +EL COM Port y se selecciona el puerto al que se encuentre conectado el microcontrolador:

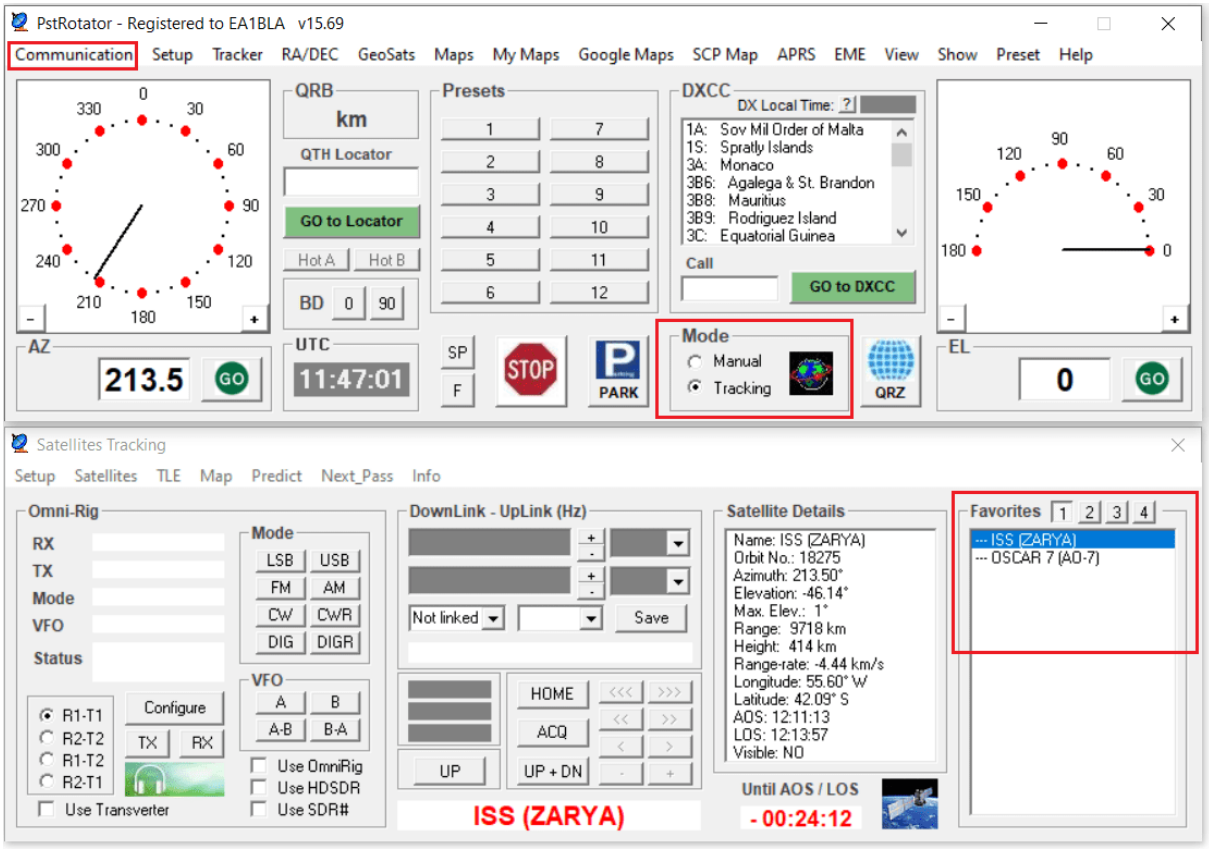


Figura 14.6 – Transmisión por puerto serie

Cabe destacar que este software ofrece la posibilidad de seguir otros objetos espaciales. En la pestaña Tracker de la pestaña principal del programa se encuentran las diversas opciones. Para este proyecto es necesario seleccionar la opción Satellites.

14.2. Decodificación de los datos

Para poder decodificar los datos con el microcontrolador, será necesario conocer el formato de la trama que se recibe por puerto serie. Para ello, mediante el programa de software libre Free Serial Analyzer se puede visualizar el puerto serie:

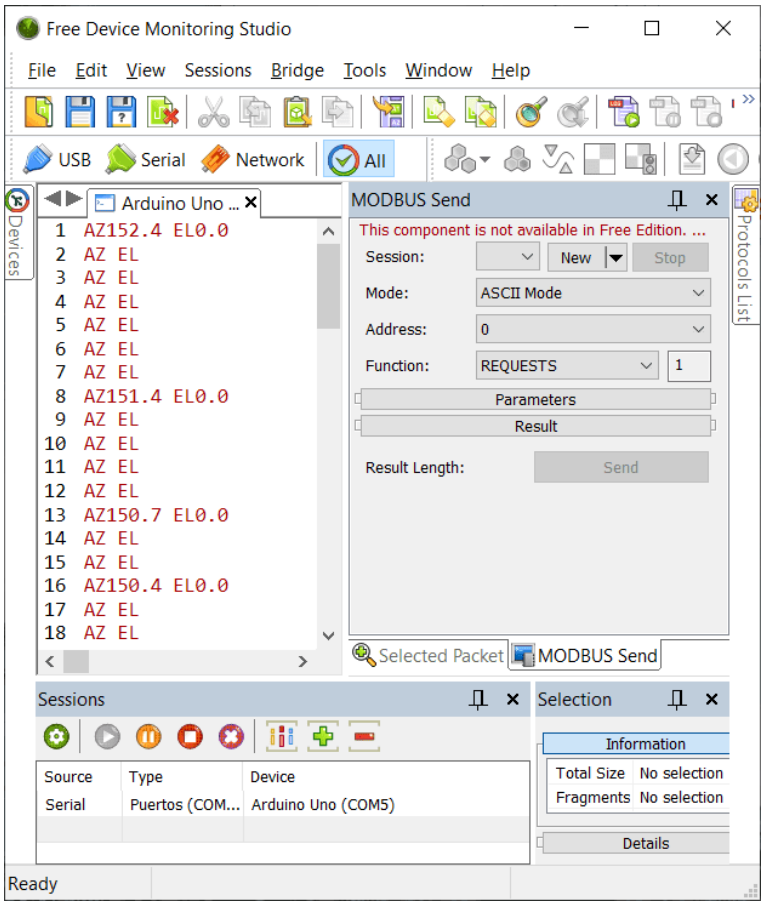


Figura 14.7 – Monitor serie en código ASCII

Como se observa en la figura 14.7, los dos ángulos van precedidos de un identificador de dos letras , AZ y EL, y separados por un espacio entre ellos. Si se observa el contenido del puerto serie en hexadecimal en lugar de en código ASCII, puede apreciarse que el elemento que separa dos mensajes Easycom es un retorno de carro, que se corresponde con el valor 0x0D en hexadecimal.

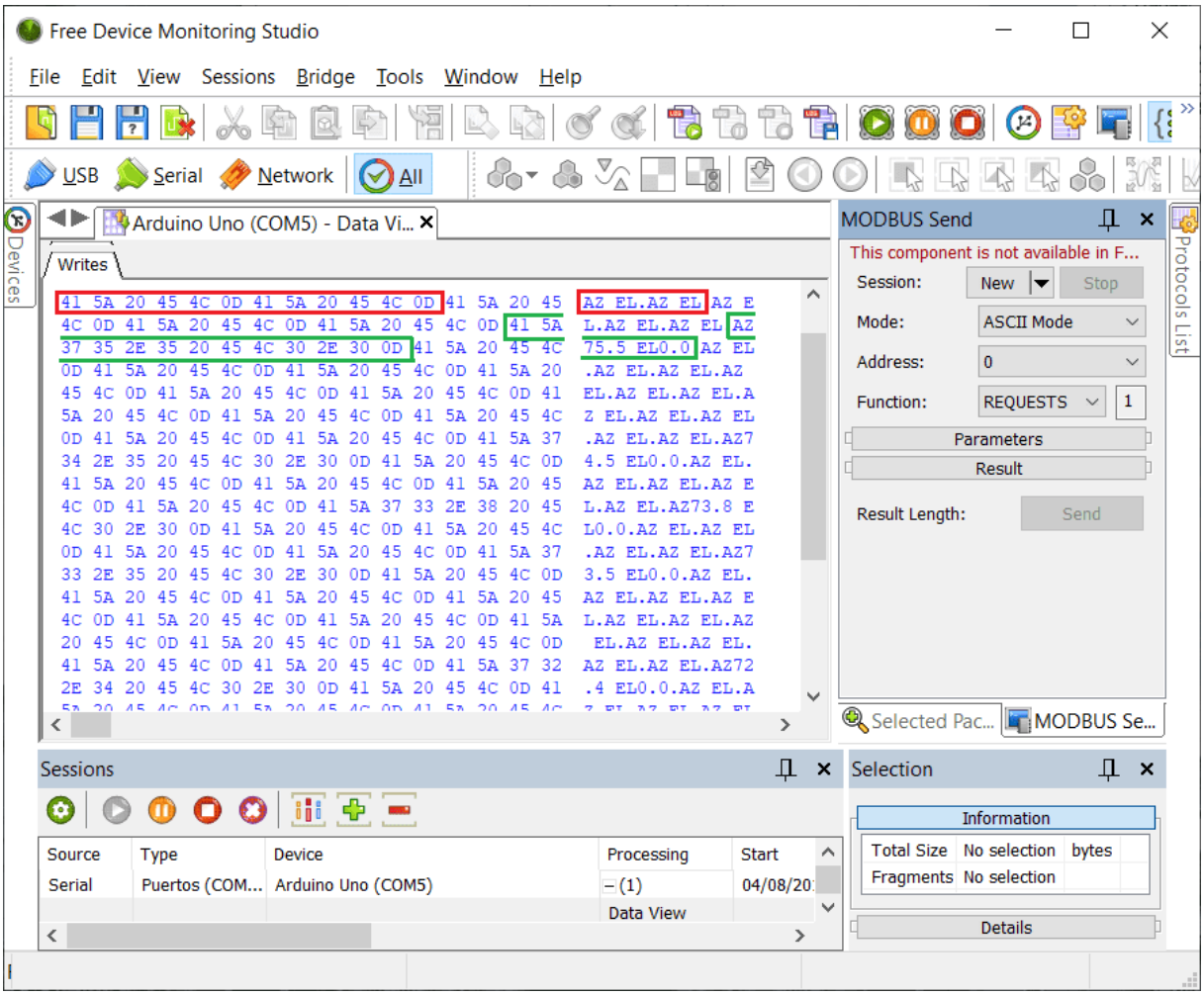


Figura 14.8 – Monitor serie en hexadecimal

Para decodificar esta trama Easycom se utiliza el mismo método que para decodificar la trama en formato NMEA del receptor GPS, es decir, una máquina de estados que verifique con cada carácter recibido por puerto serie si este se corresponde con el formato Easycom. La representación de esta máquina de estados se ilustra en la figura 14.9.

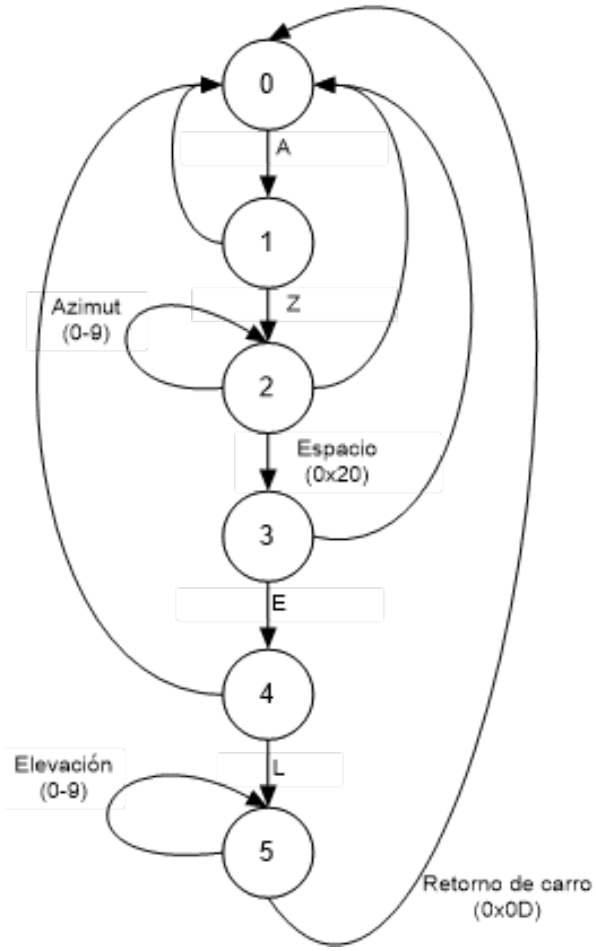


Figura 14.9 – Máquina de estados para el formato Easycom

El resultado se imprime por puerto serie para verificar el correcto funcionamiento:

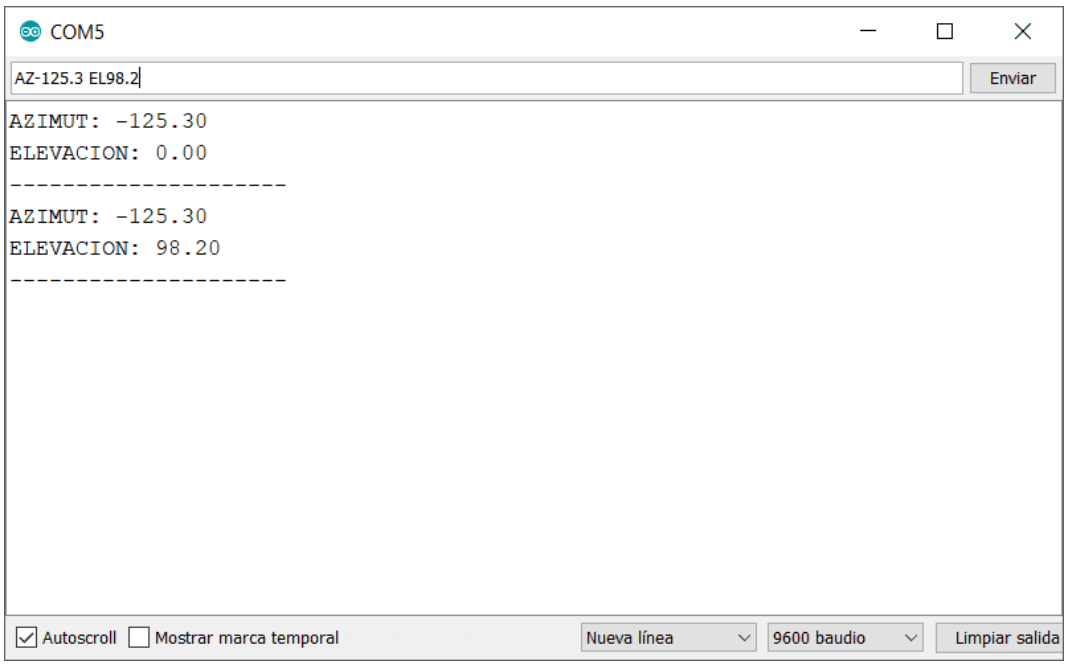


Figura 14.10 – Resultado de la decodificación Easycom

El código con el que se ha obtenido este resultado se encuentra en el programa principal (anexo 18) en las funciones referentes a la decodificación de la trama Easycom. En el se realizaron las modificaciones necesarias para poder visualizar en el monitor serie los resultados.

15 CÁLCULOS KEPLERIANOS DE APUNTAMIENTO. BASES TEÓRICAS

Tal como se ha comentado en el capítulo 7 de análisis de soluciones, el sistema es capaz de calcular la posición del satélite en coordenadas altazimutales en función del tiempo.

Para conseguir esto, se debe disponer de datos muy precisos de la órbita del satélite. Estos datos son actualizados periódicamente por NORAD (North American Aerospace Defense Command), y se publican en un formato normalizado denominado TLE (Two Line Elements). [28]

Estos conjuntos de datos incluyen los parámetros orbitales necesarios para realizar el cálculo de la posición y velocidad del satélite en un instante determinado.

Aunque en principio el cálculo pueda parecer trivial basándose en las ecuaciones de Kepler, determinar la posición del satélite no tan sencillo debido a la existencia de numerosas perturbaciones que alteran la trayectoria orbital. Para tener en cuenta estas perturbaciones, se han desarrollado unos algoritmos matemáticos, que en el argot aeroespacial reciben el nombre de propagadores. Estos algoritmos tienen efectos como los movimientos de la Tierra, atracción gravitatoria de la Luna y del Sol, rozamiento de la atmósfera, e incluso perturbaciones de espacio profundo.

El modelo de perturbaciones y propagador utilizado para este proyecto es el SGP4 (Simplified General Pertutbations Satellite Orbit Model 4). Este modelo fue desarrollado por la NASA y es especialmente útil para órbitas bajas (LEO), en las que la resistencia del rozamiento atmosférico es considerable. Con él, se consiguen precisiones en torno a 1km durante unos días. Más tarde, sería necesario actualizar los datos TLE.

15.1. Elementos orbitales

Para definir una órbita y la posición de un cuerpo sobre ella, es necesario conocer, basándose en la teoría de Johannes Kepler, además del tiempo, seis elementos orbitales.

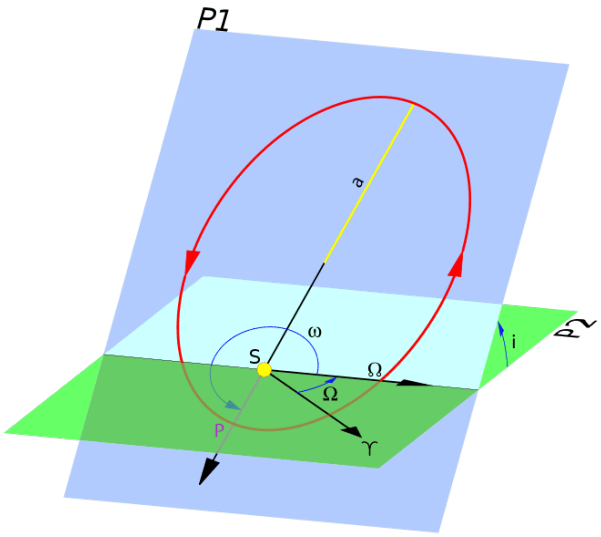


Figura 15.1 – *wikipedia*. Elementos orbitales [63]

Dos de ellos definen el tamaño y forma de la elipse:

- Excentricidad: parámetro que determina el grado de desviación respecto a una circunferencia.
- Semieje mayor: distancia desde el centro de la elipse hasta el periastro (punto más cercano de la órbita al cuerpo que genera la atracción gravitatoria de la misma).

Otros dos definen la orientación del plano orbital:

- Inclinación: ángulo vertical de la elipse con respecto al plano de referencia (en este caso, el plano ecuatorial), medido en el nodo ascendente. Este último coincide, en nuestro caso, con la dirección del equinoccio vernal, o punto Aries, que es un punto imaginario en el espacio sobre la línea de intersección del plano ecuatorial y el plano de la órbita de la Tierra alrededor del Sol, o eclíptica.
- Longitud del nodo ascendente: es el ángulo horizontal medido sobre el plano de referencia (ecuatorial para este caso) respecto al punto Aries.

Los últimos dos elementos son:

- Argumento del periastro: define la orientación de la elipse en el plano orbital, medido desde el nodo ascendente hacia el periastro.
- Anomalía media de la época: define la posición del cuerpo orbital sobre la elipse en un instante concreto. Este instante se denomina época. La anomalía define la posición utilizando una circunferencia auxiliar concéntrica con la elipse y de radio el semieje mayor. Tiene la ventaja de que el cuerpo sobre la circunferencia se mueve con velocidad constante. Se corresponde con el ángulo M en el dibujo.

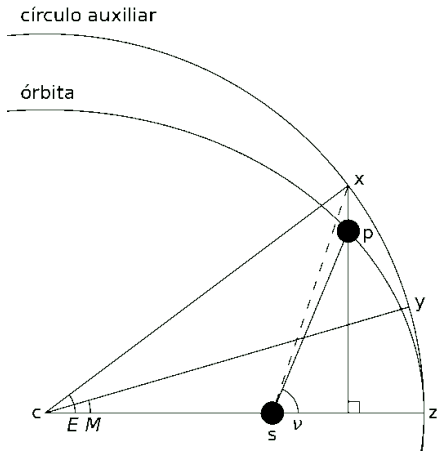


Figura 15.2 – *wikipedia*. Anomalía media [77]

15.2. Estructura del formato TLE

Los datos TLE se organizan en dos líneas de 69 caracteres cada una, precedidas de 11 caracteres con el nombre del satélite.

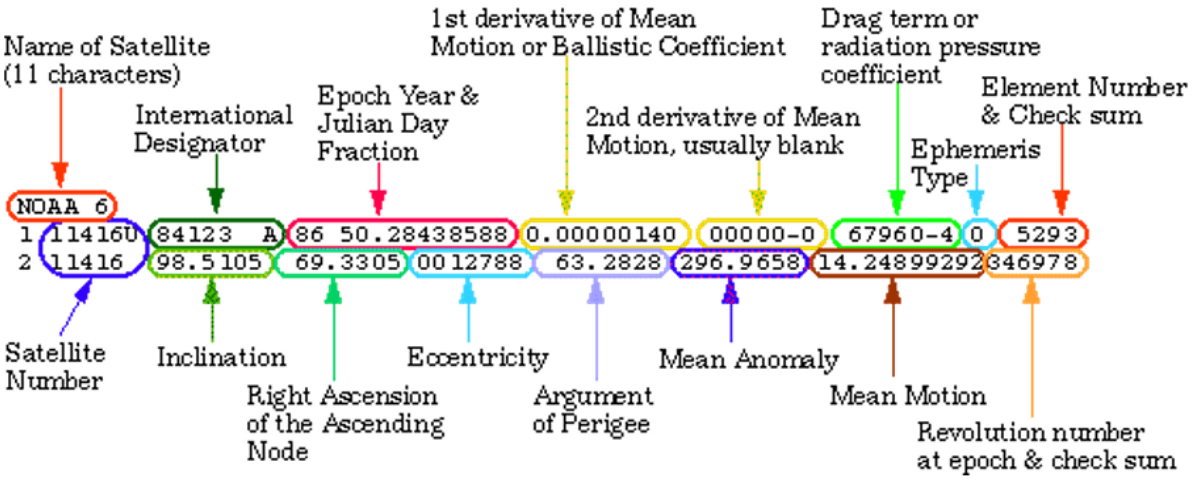


Figura 15.3 – *NASA*. TLE [62]

La primera línea contiene:

- Número del satélite más clasificación (5 + 1 caracteres)
- Designación internacional (10 caracteres)
- Instante de la época en fecha y fracción de día juliano (14 caracteres, 2 para el año y 12 para la fecha juliana). La época define el momento en el que se realizó la última actualización de los datos del satélite. Esta viene expresada en lo se denomina Día Juliano. Se entiende Día Juliano como el tiempo transcurrido desde el mediodía en Greenwich del 1 de enero de 4713 a.C. Consta de una parte entera (número de días) y una parte decimal (hora del día en UTC).

- Coeficiente balístico (11 caracteres): primera derivada del movimiento medio del satélite. Segunda derivada del movimiento medio (7 caracteres más 2 para la potencia de 10)
- Coeficiente de presión de radiación, BSTAR (7 caracteres más 2 para la potencia de 10)
- Tipo de efemérides
- Número de TLE más check sum: el número TLE es un número correlativo que se incrementa con cada actualización de datos.

En la segunda línea se encuentra:

- Número del satélite, continuación del dato homónimo de la primera línea.
- Inclinación
- Ascensión del nodo ascendente
- Excentricidad
- Argumento del perigeo
- Anomalía media
- Movimiento medio. Representa el número de vueltas por día.
- Número de revoluciones y check sum. El número de revoluciones representa el número de órbitas completadas en la época de estos datos.

15.3. Cálculo de las coordenadas altazimutales

Es conveniente comprender los sistemas de coordenadas involucrados en este proceso.

15.3.1. Coordenadas geocéntricas (ECI)

El sistema de coordenadas que se utiliza para realizar todos los cálculos de posición del satélite es el sistema denominado ECI, cuyo origen se sitúa en el centro de la Tierra. El eje X apunta en la dirección del punto Aries, el eje Z coincide con el eje de rotación terrestre y el eje Y se corresponde con un eje perpendicular a estos dos en un sistema dextrógiro. Se trata de un sistema de referencia inercial, ya que no tiene en cuenta la aceleración debida al giro en su movimiento de traslación con respecto al Sol. Es decir, sus ejes están fijos respecto del espacio profundo.

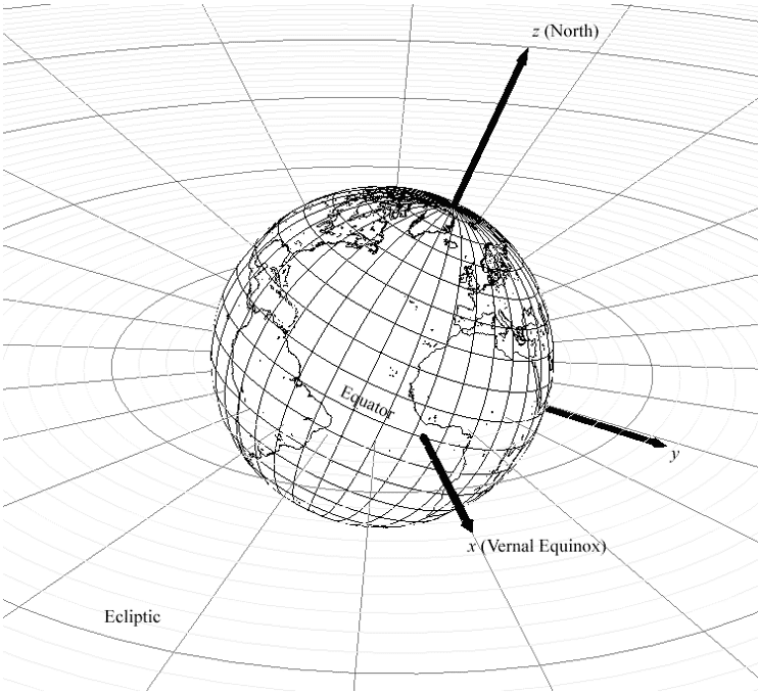


Figura 15.4 – *celestrak*. Sistema de coordenadas ECI [52]

En particular, el algoritmo orbital SGP4 utiliza el sistema de coordenadas denominado TEME (True Equator Mean Equinox), que utiliza una aproximación necesaria ya que el equinoccio vernal varía con los movimientos de precesión y nutación (este último no se tiene en cuenta en dicho algoritmo).

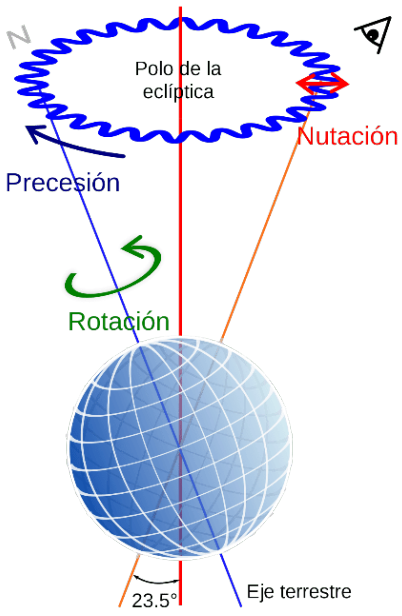


Figura 15.5 – *wikipedia*. Movimientos de precesión y nutación [81]

15.3.2. Coordenadas topocéntricas

Las coordenadas geocéntricas descritas en el apartado anterior son útiles para determinar la posición del satélite. Sin embargo, se necesita un sistema de coordenadas centrado en la posición de observación que nos permita obtener la dirección de apuntamiento del satélite desde nuestra ubicación.

El sistema de coordenadas final que se pretende obtener es un sistema de coordenadas polar en el que la posición del satélite viene determinada por la distancia al observador y los ángulos de elevación y azimut.

Dado que la posición del satélite y del observador vienen determinadas en el sistema ECI, es necesario realizar un proceso de transformación de sistema de coordenadas hasta alcanzar el resultado final. Para ello se utiliza un sistema cartesiano de coordenadas topocéntricas intermedio denominado SEZ (Sur Este Zenit).

Por otro lado, para determinar la posición del lugar de observación sobre la superficie de la Tierra, se utiliza el sistema habitual de coordenadas polares, altitud, longitud y latitud.

Por último, conviene precisar, en relación con esto último, la diferencia que existe entre coordenadas geocéntricas y coordenadas geodésicas. Estas últimas, que son las que figuran en los mapas, tienen en cuenta el achatamiento de la Tierra, no así las primeras.

15.4. Procedimiento de cálculo para el apuntamiento

El proceso de cálculo consiste básicamente en:

- Determinar el vector de posición del satélite en el instante de observación
- Determinar el vector de posición del lugar de observación
- Calcular la diferencia de ambos vectores
- Transformar este vector diferencia a los ángulos de elevación y azimut

Todos estos vectores son función del tiempo, pero hay que precisar qué referencia de tiempo se va a utilizar.

Todas las medidas de tiempo que se realizan en la vida cotidiana están referidas a lo que se denomina día solar medio, que es el tiempo transcurrido desde el mediodía de un día hasta el mediodía del día siguiente, entendiendo por mediodía el instante en el que la línea que une el centro de la Tierra con el centro del Sol pasa por nuestro meridiano local. El tiempo en este sistema de referencia es lo que se conoce como UT1 (Universal Time 1). Además, existe también el UTC (Universal Time Coordinated), que inserta un segundo en determinados momentos con el fin de evitar la imprecisión entre las unidades de medida empleadas (horas minutos y segundos) y el día solar medio real, ya que estas no coinciden. Aunque UT1 y UTC no son realmente las mismas, el error cometido se encuentra dentro de la incertidumbre teórica del procedimiento SGP4 (David Vallado e.a, *Revisiting Spacetrack Report 3*, página 6 [6]).

Además del día solar, existe otra unidad de medida: el día sideral. La definición de este coincide con la de día solar medio, pero cambiando el Sol por una estrella muy lejana.

Puede apreciarse en la figura que ambos períodos no coinciden, ya que el ángulo recorrido en un día solar es θ º mayor que recorrido en el día sideral.

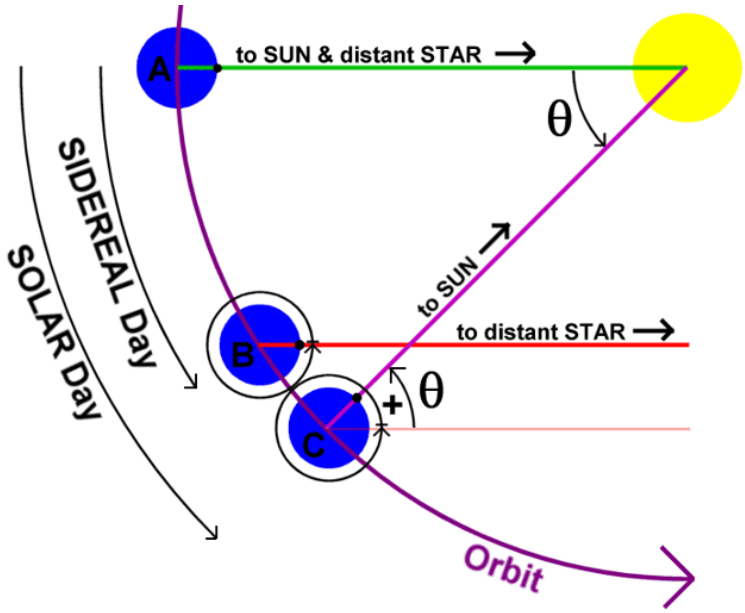


Figura 15.6 – *celestialnorth*. Día sideral y día solar medio [69]

Por otro lado, además de estar de acuerdo en las unidades de medida, es necesario tener una referencia temporal común para todos los procesos, y para eso se utiliza el día juliano (JD), definido anteriormente. De esta manera, tanto el instante de observación como la época de los datos TLE, la época del satélite y otras marcas temporales utilizadas se expresan como JD. La parte decimal de estas medidas temporales se corresponde con la hora UTC, mientras que la parte entera representa el número de días transcurridos. El día juliano se calcula mediante el algoritmo basado en el indicado en la página 606 de *Explanatory Supplement to the Astronomical Almanac* [4] (adaptado por Michael F. Henry para la aplicación Orbit Tools [51], de libre acceso para fines no comerciales):

$$JD = 367{,}0 \cdot anho - int((7 \cdot (anho + int(\tfrac{(mes+9)}{12{,}0}))) \cdot 0{,}25) + int(275 \cdot \tfrac{mes}{9{,}0}) + dia + 1721013{,}5 + \frac{(\frac{(\frac{seg}{60{,}0}+minuto)}{60{,}0}+hora)}{24{,}0} \tag{15.1}$$

entendiendo por int la parte entera de la operación.

Para determinar la posición y velocidad del satélite en el instante de la observación, debemos indicar al algoritmo propagador SGP4 los datos del satélite decodificados del conjunto TLE, así como el instante de observación en JD. Con estos datos, el algoritmo calcula los vectores de posición (x_s, y_s, z_s) y velocidad (Vx_s, Vy_s, Vz_s) en coordenadas ECI.

A continuación, se calcula el vector de posición del lugar de observación, también en coor-

denadas ECI. Inicialmente se supone que la Tierra es completamente esférica, suposición que será corregida más adelante.

La siguiente figura representa un corte transversal de la Tierra por un meridiano, donde se observan la coordenada z y la distancia R de la posición del observador al eje terrestre, en función del radio de la Tierra Re y del ángulo de latitud.

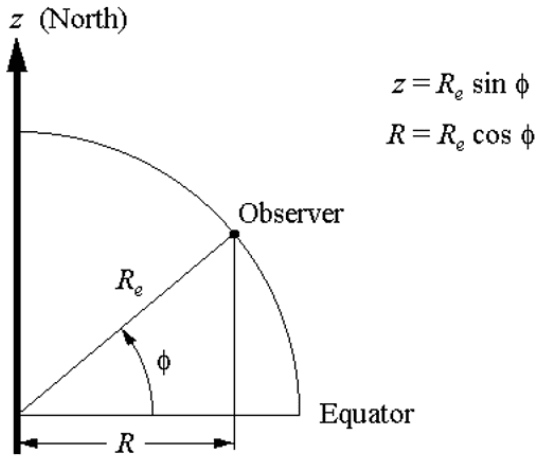


Figura 15.7 – *celestrak*. Coordenada z y distancia R [52]

Las coordenadas x e y no son tan inmediatas porque dependen del tiempo. Por tanto, es necesario conocer el ángulo del lugar de observación con respecto al punto Aries. Para ello se calcula la posición del meridiano de Greenwich a las 00:00:00 horas del día de la observación.

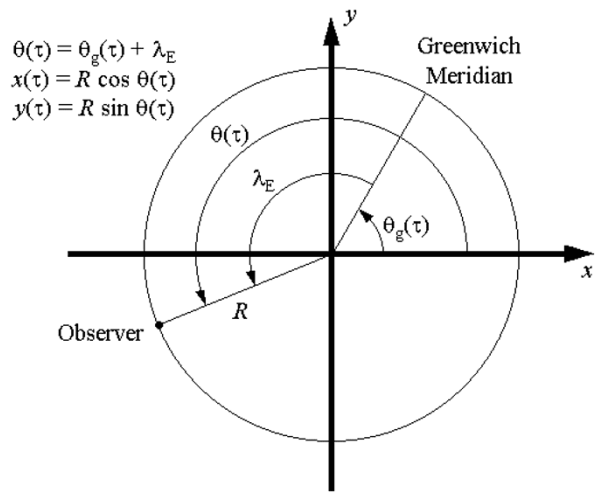


Figura 15.8 – *celestrak*. Coordenadas x e y [52]

Este cálculo se realiza con la siguiente fórmula (página 50 de *Explanatory Supplement to the Astronomical Almanac* [4]) (en UT1):

$$\theta_g(0h) = 24110s{,}54841 + 8640184s{,}812866 \cdot Tu + 0s{,}093104 \cdot Tu^2 - 6{,}210 - 6Tu^3 \qquad (15.2)$$

donde $T_u = \frac{d_u}{36525}$, siendo d_u el número de días de UT1 transcurridos desde el 1 de enero

del 2000 a las 12:00:00 UT1. A este valor hay que sumarle el ángulo recorrido por el meridiano de Greenwich hasta la hora de observación. Este ángulo viene dado por el producto de la velocidad de rotación de la Tierra por la hora UT1 (parte decimal de JD), (θ_g):

$$\theta_g(\tau) = \theta_g(0h) + \omega_e \cdot \Delta_\tau \tag{15.3}$$

Además, al valor anterior se le suma la longitud de la posición del observador (λ_E).

$$\theta = \theta_g + \lambda_E \tag{15.4}$$

Por lo tanto, las coordenadas ECI del vector de posición del observador son:

$$x_o = R \cdot cos(\phi) \tag{15.5}$$
$$y_o = R \cdot sin(\phi) \tag{15.6}$$
$$z_o = Re \cdot sin(\phi) \tag{15.7}$$

donde

$$R = Re \cdot cos(\phi) \tag{15.8}$$

Con esto ya se puede calcular el vector de posición del satélite respecto a la posición del observador (en coordenadas ECI), calculado como la diferencia entre la posición del satélite y la posición del observador $r = (x_s - x_o, y_s - y_o, z_s - z_o)$.

El último paso consiste en realizar las transformaciones de ejes del sistema de coordenadas geocéntrico al topocéntrico. Primero se transforma el sistema ECI en SEZ, como se describe en la siguiente figura:

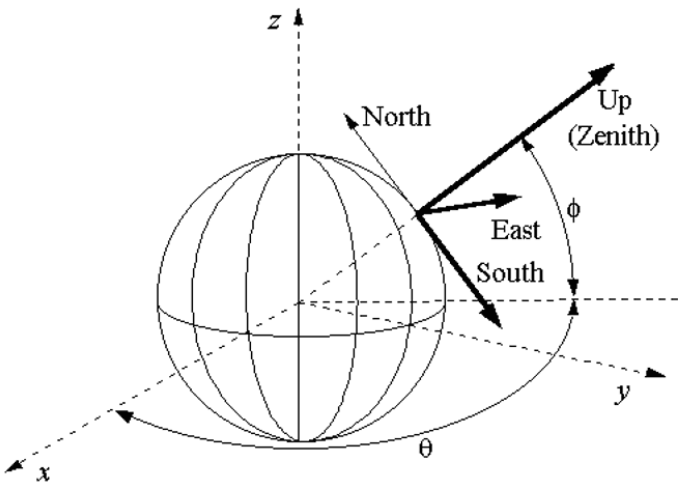
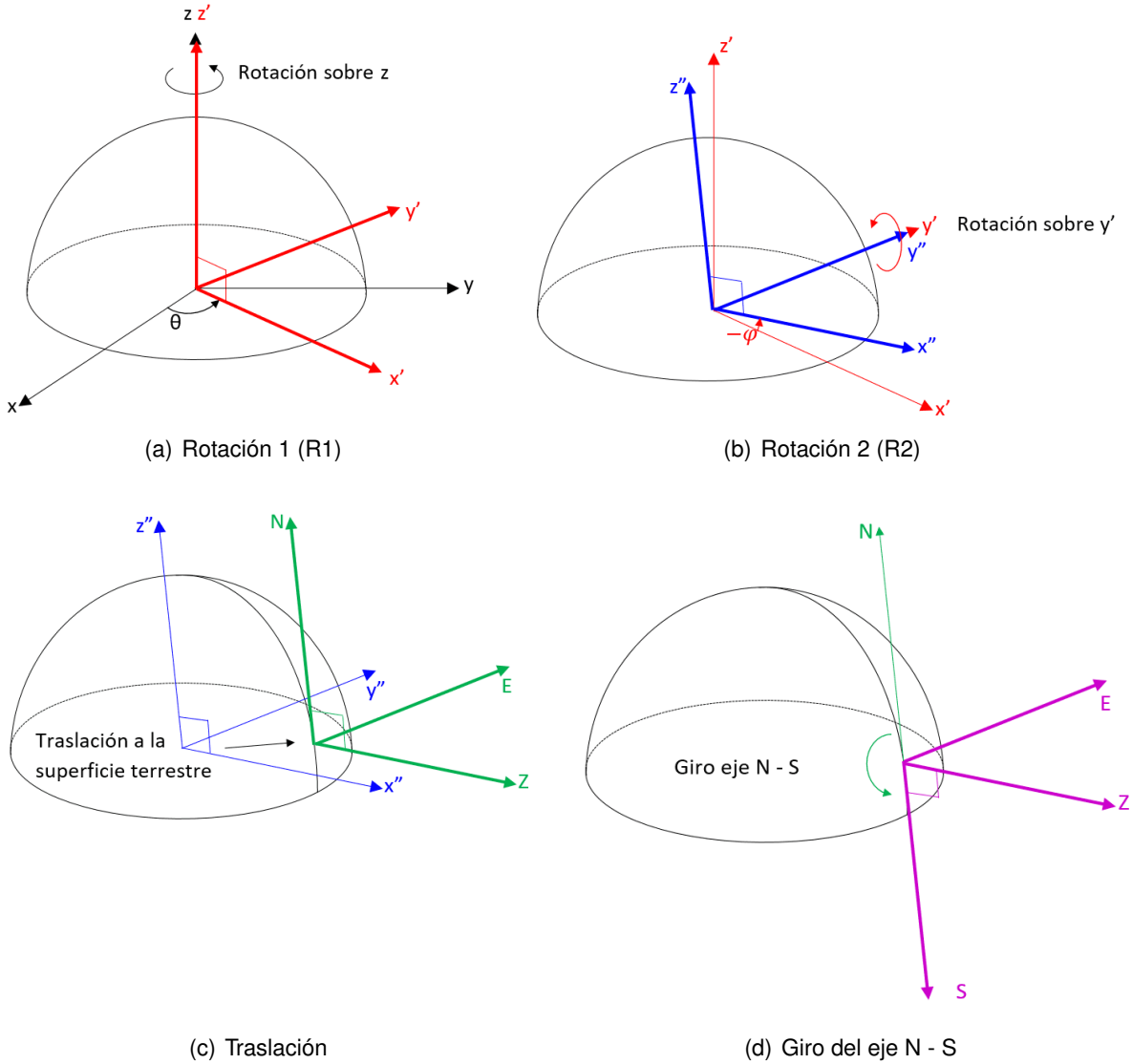


Figura 15.9 – *celestrak*. Coordenadas SEZ [53]

1. Rotación de θ º en torno al eje z

2. Rotación de ϕ º en torno al nuevo eje y

3. Inversión del eje z original para que apunte al Sur en lugar del Norte, y conformar así el sistema SEZ



(a) Rotación 1 (R1)

(b) Rotación 2 (R2)

(c) Traslación

(d) Giro del eje N - S

Figura 15.10 – Transformaciones de coordenadas

Las matrices de rotación quedan:

$$R1 = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R2 = \begin{pmatrix} cos(-\phi) & 0 & -sin(-\phi) \\ 0 & 1 & 0 \\ sin(-\phi) & 0 & cos(-\phi) \end{pmatrix}$$

$$R = R2 \cdot R1 = \begin{pmatrix} cos(\phi)cos(\theta) & cos(\phi)sin(\theta) & sin(\phi) \\ -sin(\theta) & cos(\theta) & 0 \\ -sin(\phi)cos(\theta) & -sin(\phi)sin(\theta) & cos(\phi) \end{pmatrix} \tag{15.9}$$

Así, para obtener las coordenadas del vector diferencia en el sistema SEZ, se realiza primero la siguiente ecuación matricial, que nos da las coordenadas en el sistema NEZ (Norte Este Zénit):

$$\begin{aligned} r_{NEZ} &= R \cdot r_{ECI} \\ \begin{pmatrix} r_N \\ r_E \\ r_Z \end{pmatrix} &= R \cdot \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \end{aligned} \tag{15.10}$$

siendo r el vector diferencia y R la matriz de rotación final.

Ahora se invierte el eje Norte (N), obteniendo así el sistema de coordenadas SEZ:

$$\begin{pmatrix} r_S \\ r_E \\ r_Z \end{pmatrix} = \begin{pmatrix} -r_N \\ r_E \\ r_Z \end{pmatrix} \tag{15.11}$$

Hay que tener en cuenta que el movimiento de traslación de los ejes no resulta necesario ya que queda implícito en el vector diferencia entre la posición del satélite y la del observador (ambas referidas al centro de la Tierra).

Transformado el vector al sistema SEZ, queda simplemente calcular las coordenadas altazimutales.

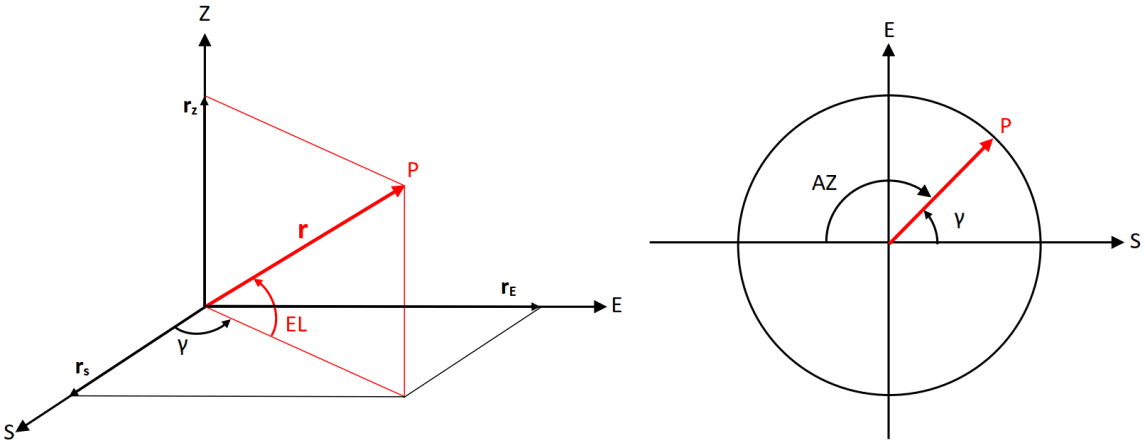


Figura 15.11 – Ángulos de azimut (AZ) y elevación (EL)

De la imagen 15.11 se pueden obtener las coordenadas altazimutales del punto P, que representa la posición del satélite:

$$r = \sqrt{r_S^2 + r_E^2 + r_Z^2} \tag{15.12}$$

$$EL = sin^{-1}(\frac{r_Z}{r}) \tag{15.13}$$

$$\gamma = tan^{-1}(\frac{r_E}{r_S}) \tag{15.14}$$

$$AZ = \pi - \gamma \tag{15.15}$$

Debe observarse que el ángulo γ está medido con respecto al Sur, mientras que los ángulos de azimut se miden con respecto al Norte, por lo que:

$$AZ = tan^{-1}(-\frac{r_E}{r_S}) \tag{15.16}$$

Por último, queda por realizar la corrección por el achatamiento de la Tierra.

En el proceso el proceso anterior se ha mencionado que los cálculos están simplificados al considerar la Tierra como una esfera perfecta. Sin embargo, esto no es así, sino que presenta un achatamiento en la dirección del eje de rotación, que origina dos sistemas de coordenadas distintos para el posicionamiento de un punto sobre la superficie terrestre: sistema geocéntrico y sistema geodésico, ambos en coordenadas polares, que es el método utilizado para definir la posición de un punto sobre la Tierra, mediante los ángulos de latitud y longitud.

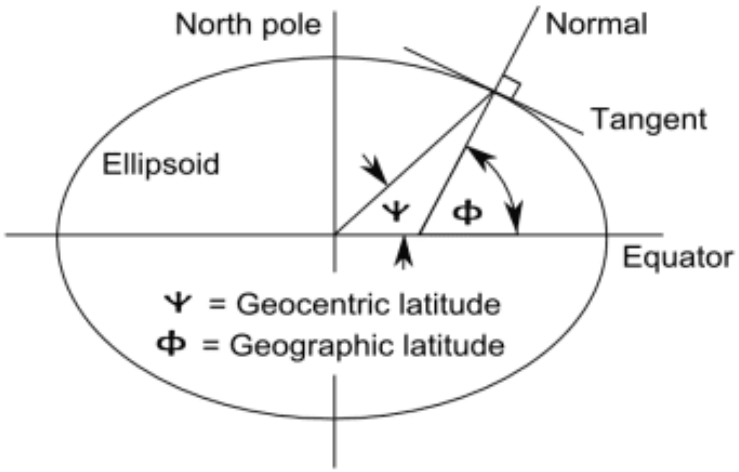


Figura 15.12 – *johndcook*. Coordenadas geocéntricas y geodésicas [35]

En un sistema de coordenadas geocéntrico, el plano horizontal es perpendicular a la línea que une la posición del observador con el centro de la Tierra, y esta línea es lo que se toma como vertical del lugar.

Por el contrario, dado que la Tierra no es esférica, la anterior afirmación no es correcta, de manera que la vertical es una línea perpendicular al plano horizontal, siendo este este el plano tangente a la superficie terrestre en dicho punto, como puede verse en la figura 15.12.

Como consecuencia, el ángulo de latitud, que es el ángulo formado entre la vertical y el plano del ecuador, es diferente en ambos sistemas.

El error cometido por considerar la Tierra esférica puede corregirse con el siguiente procedimiento (página 126 de *Explanatory Supplement to the Astronomical Almanac* [4]), que considera la Tierra como un esferoide de referencia (una elipse de revolución que queda definida por su radio (a) y achatamiento (f), dónde es el radio de la Tierra en km y $f = \frac{1}{298,257}$).

Se definen dos variables auxiliares c y s, que son función de la latitud (ϕ) y longitud (θ) geodésicas:

$$c = \frac{1}{\sqrt{(1 + f \cdot sin^2(\phi) \cdot (f - 2))}} \tag{15.17}$$

$$s = (1 - f)^2 \cdot c \tag{15.18}$$

Y con ellas obtenemos las coordenadas (x, y, z) del punto de observación en coordenadas ECI.

$$x_o = (a \cdot c + h) \cdot cos(\phi) \cdot cos(\theta) \tag{15.19}$$

$$y_o = (a \cdot c + h) \cdot cos(\phi) \cdot sin(\theta) \tag{15.20}$$

$$z_o = (a \cdot s + h) \cdot sin(\phi) \tag{15.21}$$

16 CÁLCULOS KEPLERIANOS DE APUNTAMIENTO. ACLARACIONES DEL SOFTWARE

Para realizar los cálculos de apuntamiento del sistema de forma autónoma, se ha elaborado un software que implementa los procesos teóricos descritos en 15, utilizando las funciones de David Vallado para el propagador SGP4 [6]. Estas son complemento de su libro *Fundamentals of astrodynamics and applications* [5].

La programación de este cálculo autónomo incorpora software de diversas fuentes, especialmente el de Michael F. Henry, de Orbit Tools [51], que realiza el proceso completo de apuntamiento, y el del mencionado David Vallado. Este último no está orientado al apuntamiento, sino a la realización de comprobaciones de operatividad y precisión del propagador.

Una vez analizado el funcionamiento de este software, se procedió a la programación de un código que se adaptara a las necesidades de este proyecto, incluyendo también funciones (de código libre) te terceros (D. Vallado), concretamente las que realizan la propagación con el algoritmo SGP4. Estas requieren unos conocimientos de teoría y tecnología aeroespacial que superan el alcance de este proyecto. En algunos casos, como se comentará más adelante, ha sido necesario realizar ciertas modificaciones al código original para adaptarlo a las necesidades de este sistema.

La elaboración del software para el cálculo autónomo de apuntamiento se realizó en dos etapas:

- En la primera, se optó por compilar y ejecutar el programa sobre el propio PC, debido a las numerosas pruebas a realizar, que requerían una agilidad que no sería posible si se compilara sobre Arduino, dada la considerable lentitud del proceso de compilación y grabación debido al tamaño del programa.

  Es en esta fase en la que se elaboró el código, y se realizaron las verificaciones de precisión, obteniendo unos resultados altamente satisfactorios.

- En la segunda fase, se realizaron los cambios necesarios para adaptar el código anterior a la plataforma Arduino.

  Durante esta fase se encontró un inesperado problema de precisión que hacía que el programa fuera completamente inservible, dando unos resultados completamente erróneos. Por este motivo, fue necesario realizar una nueva serie de modificaciones para obtener unos resultados aceptables como se observa en las figuras 16.1 y 16.2.

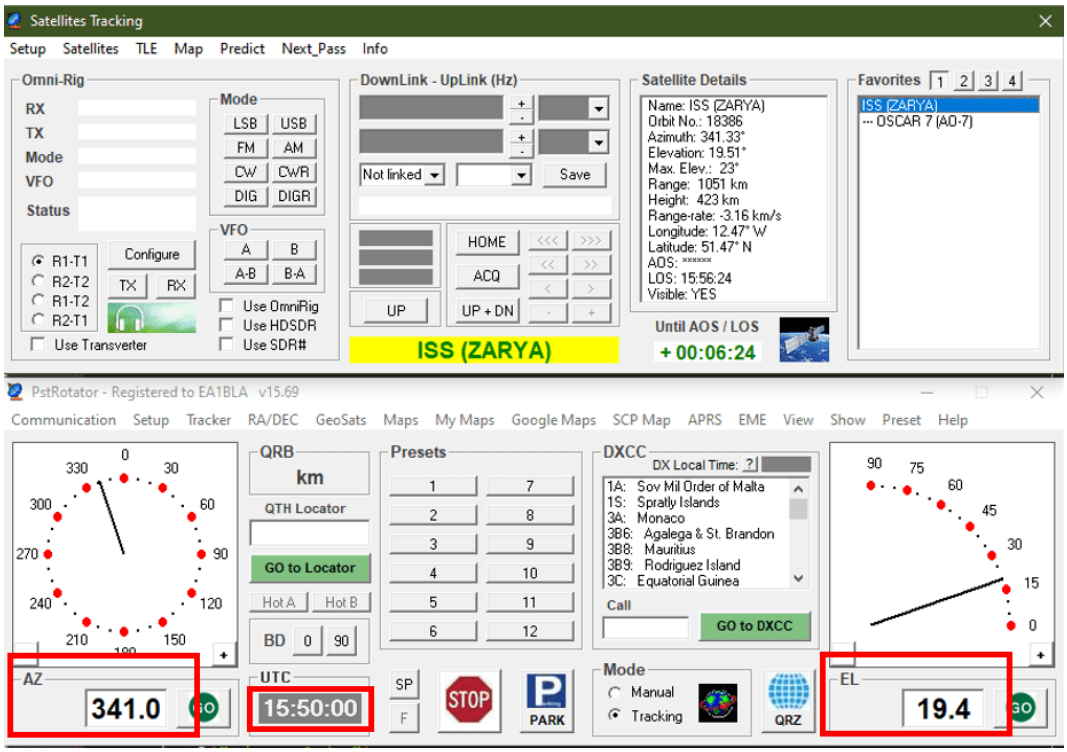


Figura 16.1 – Resultado en Pst Rotator

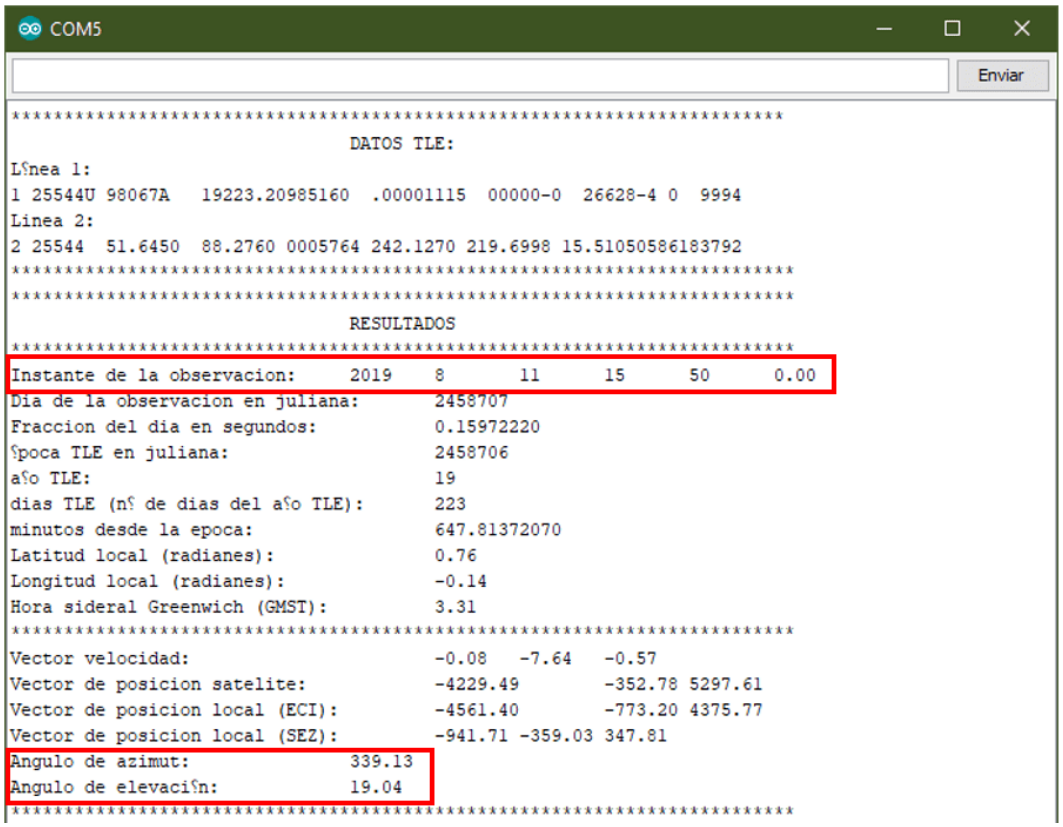


Figura 16.2 – Resultado en Arduino

La falta de precisión se debía al hecho de que muchas de las variables empleadas por

las funciones del propagador SGP4 y otras funciones son de coma flotante con doble precisión (tipo double), de 8 bytes. Estas no causaron problemas en la compilación en PC, sin embargo, tal como se documenta en la web de Arduino [13], en el modelo utilizado (MEGA 2560), las variables double son reducidas a tipo tipo float, de 4 bytes.

Aunque a primera vista, esta precisión pueda parecer suficiente, dependiendo del tamaño del número representado, el número de cifras decimales se ve recortado considerablemente, haciendo el cálculo totalmente incorrecto. Esto sucedía, por ejemplo, con el instante de observación, que en el código original viene expresado en días, por lo que los segundos ocupan de la quinta posición decimal en adelante.

Al ejecutar el programa en Arduino y convertir estas cantidades a tipo float, quedaban recortadas a 2 cifras decimales, que corresponden a las horas, por lo que puede comprenderse que el sistema resultaba inservible para el seguimiento.

Para corregir este problema hubo que sustituir varias de las funciones originales, programar otras nuevas e introducir modificaciones en el software original del propagador SGP4, que se detalla a continuación. Básicamente, las modificaciones introducidas consisten en separar la parte entera de las variables que representan el instante de la observación (días) de la parte decimal (horas, minutos y segundos), para permitir que las variables de tipo float conserven todas las cifras significativas necesarias.

16.1. Descripción del código

Con el objetivo de ofrecer más claridad en la organización de las funciones, se ha mantenido la organización del código empleado en la comprobación sobre el compilador del PC, y se han separado las funciones referentes al cálculo autónomo del programa principal de basado en menús.

Así, las funciones empleadas para el cálculo autónomo están definidas en el fichero '.cpp' *sgp4*, y sus cabeceras, declaradas en '.h' *sgp4*.

En lo referente al programa principal (.ino), en la función *setup()*, que se ejecuta al iniciar el sistema, los datos TLE guardados en la memoria EEPROM son cargados en dos vectores de caracteres llamados *linea1[]* y *linea2[]*.

Cuando se selecciona el modo de funcionamiento de cálculo autónomo en el menú del setup, los datos de hora y localización son actualizados periódicamente con la función *leeGPS()*, llamada desde la función *automatico()*, que representa el modo automático de funcionamiento. Esos datos se pasan como argumentos a la función *calc_kepler()*, siendo esta es la que inicia el proceso de cálculo para la obtención de los ángulos de azimut y elevación.

La función *tleArv()* interpreta los datos TLE, los acondiciona y los guarda en una estructura, definida como variable global, llamada *datosSat*, del tipo *elsetrec*. Esta estructura está declarada en el fichero *sgp4.h*, y es original del software del propagador SGP4 (las funciones de este software se refieren a ella como *satrec*). Incluye todos los datos del satélite requeridos para la aplicación del algoritmo propagador.

Como se ha indicado en el anexo 15, es necesario referir el tiempo a un origen común, lo que se conoce como día juliano. Esto es lo que hace la función *diajul()*. Sin embargo, los datos TLE indican la época (instante de actualización de dichos datos) en formato de año (de dos dígitos) y número de días del año, con parte decimal que indica la hora. Por ejemplo, 19223,523410 indicaría que nos encontramos en el día 223,523410 del año 2019.

Para poder determinar el día juliano de la época hay que transformar ese formato TLE en año, mes, día, hora, minutos y segundos. Esto se consigue con la función *days2mdhms()* (original del software de David Vallado).

La función que implementa el algoritmo SGP4 es *sgp4()*, pero previamente hay que realizar una inicialización de los parámetros, para poder aplicar dicho algoritmo. Esto lo hace la función *sgp4init()*, que toma los datos cargados en la estructura *datosSat* (*satrec* en esta función) procedentes de los TLE, y realiza las operaciones necesarias para obtener todos los datos requeridos por el SGP4. Para ello, *sgp4init()* requiere las funciones:

- *initl()*: es otra función de inicialización . Calcula algunos valores adicionales de la época del satélite y del ángulo theta de posición del meridiano de Greenwich en ese momento, así como la corrección conocida como mecanismo de Kozai que determina la perturbación de la órbita por la atracción gravitatoria de otro cuerpo.

- *getgravconst()*: el software original del propagador tiene un propósito de verificación, y permite la utilización de diversos modos de operación así como varios juegos de constantes gravitatorias, necesarias para los cálculos de perturbación de la órbita. Estas están referidas a distintos marcos de referencia geodésicos, denominados wgs (World Geodetic System), concretamente el wgs-72, wgs-72 de baja precisión y wgs-84.

  Esta función toma los valores adecuados, dependiendo del marco de referencia elegido. En el caso concreto de este proytecto, se utiliza el valor por defecto wgs-72, el usado como referencia en el Spacetrack Report #3 [6](David Vallado e.a. ,página 15).

- *dpper()*, *dscom()* y *dsinit()*: determinan los elementos de espacio profundo que pueden influir en la alteración de la órbita. Sin embargo, estas funciones sólo se aplican para órbitas que no entran en la categoría de órbitas bajas (se consideran órbitas bajas aquellas que tienen un período inferior a 225 minutos). Este procedimiento proporciona contribuciones en el espacio profundo a elementos medios para perturbar el tercer cuerpo. Estos efectos se han promediado en una revolución del sol y la luna. Para los efectos de resonancia de la tierra, los efectos se han promediado sin revoluciones del satélite (movimiento medio).

Además, en relación con la contribución de los elementos de espacio profundo, *sgp4()* llama a la función *dspace()*, que promedia los efectos de estos elementos sobre una revolución del Sol y la Luna.

En cuanto al modo de operación, el software de David Vallado ofrece dos posibilidades: la estándar AFSPC (Air Force SPace Command) (a) y la mejorada (i). Se ha optado por la opción estándar, que ofrece una suficiente precisión para los cálculos realizados.

El resultado final de la función *sgp4()* es el vector de posición del satélite en el momento de la observación (*rSat[]*), en coordenadas ECI.

Una vez aplicado el propagador, se calcula el vector de posición del punto de observación. Para ello usamos la función *theta()*, que proporciona el ángulo theta (en radianes) entre el punto de observación y el equinoccio vernal (punto aries) en el momento de la observación. A continuación aplicamos la corrección por achatamiento de la Tierra, mediante la función *geoide()*, que finalmente nos proporciona el vector de posición local (*rLoc[]*), también en coordenadas ECI.

La diferencia entre estos dos vectores (*rSat[]* y *rLoc[]*) proporciona las componentes cartesianas del vector de posición del satélite desde el lugar de observación (vector *dif[]*), también en coordenadas ECI.

El siguiente paso es la aplicación de la matriz de transformación descrita en el anexo 15, para transformar el vector diferencia al sistema de coordenadas SEZ (vector *rSez[]*).

Por último, se pasa de coordenadas rectangulares a polares, para determinar los ángulos de azimut y elevación.

En la segunda fase, comentada anteriormente, para solventar los problemas con las variables de tipo double al implementar el programa en Arduino, además de adaptar las funciones propias y crear otras (*restaxSep()*, que realiza la diferencia de números expresados con la parte entera y la decimal por separado), se realizaron las siguientes modificaciones sobre el código del propagador SGP4:

- En la estructura *datosSat* (*satrec* en el código original):
  - *double jdstepoch* fue sustituida por *long int jdsatepoch* más *double jdsatepoch-fraccion*
  - *double epochdays* fue sustituida por *long int epochdays* más *double epochdays-fraccion*
- En la función *days2mdhms()*:
  - El segundo argumento, *double days* fue sustituido por *long int days* más *double days-fraccion*
  - *dayofyr* se igualó a *days*
  - Se modificó la conversión a horas, minutos y segundos
- En la función *sgp4init()*:
  - Se sustituyó *const double epoch* (4º argumento) por *const long int epoch* más *const double epoch-fraccion*
  - Se incluyó una nueva variable *double época = epoch + epoch-fraccion*
  - Llamamos a *dscom* con epoca en lugar de con *epoch*
- En la función *initl()*:

- Se sustituyó el 4º argumento, *double epoch* por *const long int epoch* más *const double epoch-fraccion*
- Se modificaron los cálculos de las variables locales *ts70* y *ds70*

17 FLUJOGRAMAS DEL CÓDIGO DE PROGRAMACIÓN

17.1. Flujograma principal. Sistema de menús

Se ha mencionado en apartados anteriores que todo el funcionamiento del sistema está gestionado por el programa cargado sobre la placa del microcontrolador de Arduino.

Este programa está basado en un sistema de menús que permiten al usuario acceder a los distintos modos de funcionamiento y configuración.

El sistema de menús involucra, para cada menú, una variable matricial tipo char en la que se almacena el texto del menú correspondiente y un contador, que se incrementa o decrementa con cada pulsación de las teclas para desplazarse por el menú. Este contador sirve como índice de la matriz a la hora de presentar el texto del menú en la pantalla. Se necesita también una variable (nivel), que indicará qué opción de cada menú ha seleccionado.

Por ejemplo, para el menú principal tendríamos:

- Variable de nivel: nivel_menu
- Índice de la matriz: x
- Menus[]: matriz tipo char

Mientras nivel-menu = 0, permanecemos en el entorno de selección de los modos de funcionamiento que ofrece el menú principal, incrementando y decrementando el índice de la matriz [x] con cada pulsación de las teclas de selección.

Al pulsar la tecla central (intro), se accede a ese modo de funcionamiento dado que se actualiza el valor del nivel del menú, nivel_menu = x + 1.

De esta froma, mientras no se seleccione ningún modo, nivel-menu = 0.

A continuación se muestra el diagrama de flujo principal del programa, así como los flujogramas de las funciones relevantes que intervienen en el funcionamiento del sistema.

Estos diagramas de flujo ilustran de una manera visual el funcionamiento del software, y sirven de guía, junto con las anotaciones en los comentarios del mismo, para comprender el código.

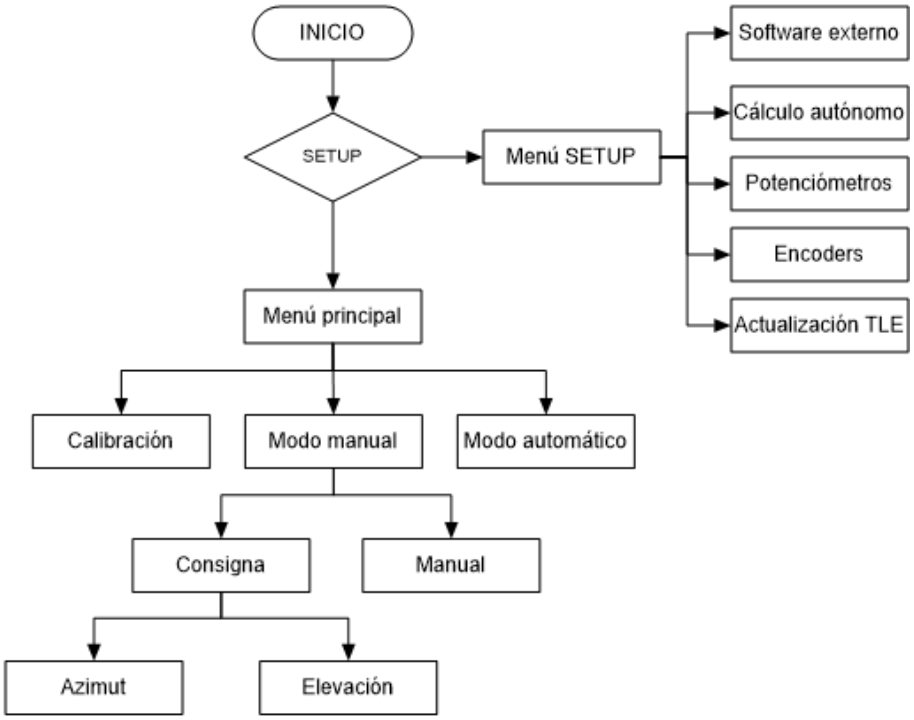


Figura 17.1 – Flujograma principal

Los cometidos de cada uno de los distintos menús se han detallado anterioridad en la sección 8.2, y en el manual de usuario del anexo 19 se especifica el modo de utilización del sistema de menús.

17.2. Flujograma del modo de seguimiento automático

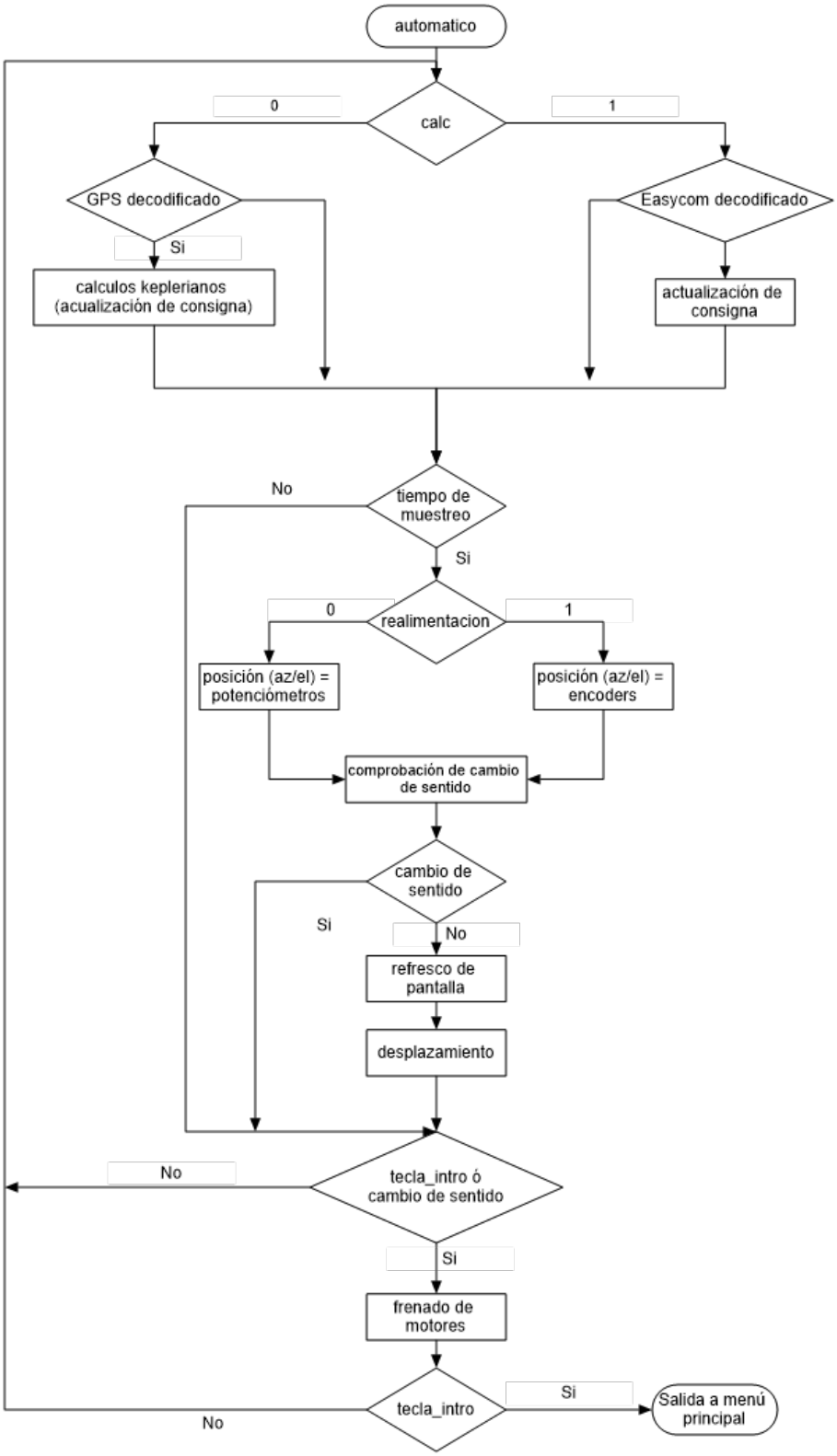


Figura 17.2 – Modo automático

Ya se ha explicado en la sección 8.2 que este modo de funcionamiento es el que permite apuntar en todo momento a la posición actual del satélite.

La variable *calc* al inicio del flujograma determina el modo de obtención de los ángulos de apuntamiento, seleccionado en el setup.

A continuación, cuando se cumple el tiempo de muestreo, se adquiere la posición de la antena, ya sea a través de encoders o a través de potenciómetros, según se haya establecido en el setup.

Adquirida la posición se comprueba que no se haya producido ningún cambio de sentido. Ya se ha explicado en la sección 8.2 que un cambio de sentido sucede, por ejemplo, si la antena se pasa de la posición de consigna, o bien si hay un cambio de consigna que obligue a los motores a girar en sentido contrario al que actualmente se encuentra girando. Este último caso puede darse, por ejemplo, si en el software de cálculo externo se cambia el satélite que se desea seguir sin antes haber detenido los motores, acción altamente recomendable si se desea cambiar de satélite.

En caso de que no se haya detectado cambio de sentido, se calcula la señal PWM para cada motor.

En caso de que sí se haya detectado cambio de sentido o se haya pulsado la tecla central de la botonera, se frenan los motores. Si el motivo de la frenada fue por un cambio de sentido, se vuelve al inicio de la función. Por otro lado, si se ha pulsado la tecla central, además de frenar los motores, el programa regresa al menú principal.

17.3. Control de posición y velocidad. Flujograma

Para realizar controles de posición y velocidad se suelen emplear reguladores PID. Sin embargo, para el caso concreto de este proyecto, no será así.

En primer lugar, la parte derivativa del regulador queda descartada, ya que conferiría al sistema un carácter más brusco, cuando la intención es justo la contraria, debido a que, como ya se ha mencionado, se trata de un sistema de naturaleza lenta.

Por otro lado, la parte integral de un regulador reduce el error en el estado estacionario. En este proyecto, una vez alcanzada la posición deseada por los motores, estos se enclavarán con un freno del propio motor. Por lo tanto, no será necesario corregir el error en régimen permanente.

Queda, por tanto, un regulador de tipo proporcional. Un regulador proporcional genera una señal de control proporcional al error. Así, para este proyecto, la señal de control se corresponde con la señal PWM de cada motor, y el error es la diferencia entre la posición deseada y la posición actual de la antena. De esta forma, cuanto mayor sea el error, mayor será la señal de control generada.

Teniendo en cuenta que los cambios bruscos de potencia son perjudiciales para los motores eléctricos y para la propia estructura del sistema, especialmente si se trata de motores de alta potencia, como sería el caso de un radiotelescopio o de antenas de grandes dimensiones, es necesario evitar cambios bruscos de potencia en el motor, que dañarían la estructura del mismo. Por tanto, a este regulador proporcional se le añade una restricción tanto en la aceleración como en la deceleración del mismo.

Así, el sistema acelerará siempre con una rampa de aceleración constante, calculando en todo momento la parte proporcional. En el momento en el que la parte proporcional calcule una velocidad menor que la calculada por la rampa de aceleración, la velocidad PWM será entonces la parte proporcional. Si esta parte proporcional decelera en exceso, también se limita el cambió de velocidad.

Si en cualquier momento cambia la consigna de posición, y el error es suficientemente grande como para superar la velocidad calculada por la deceleración, se acelera el motor con la rampa de aceleración que se mencionaba anteriormente.

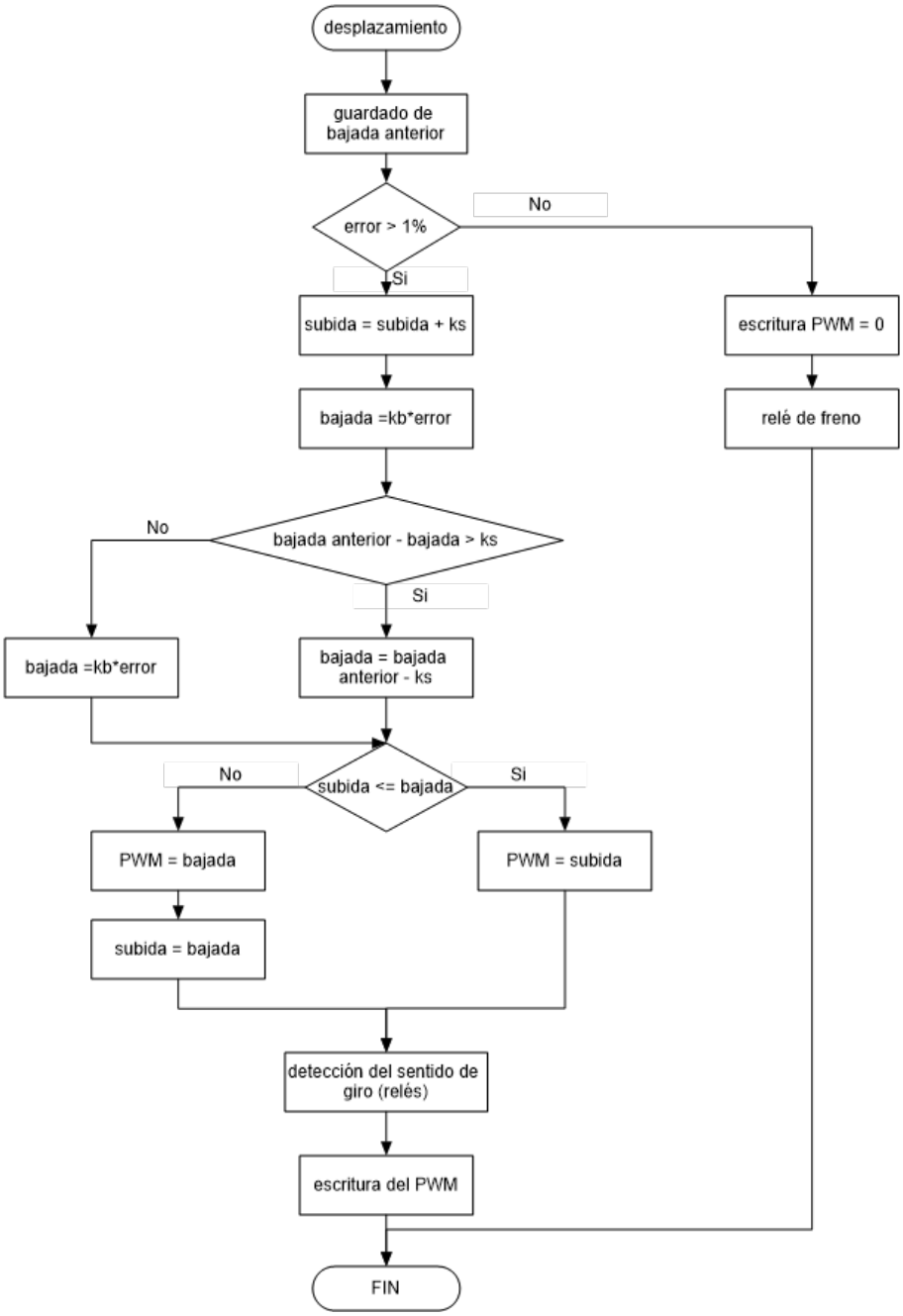


Figura 17.3 – Control de posicionamiento

17.4. Flujograma del posicionamiento por consigna

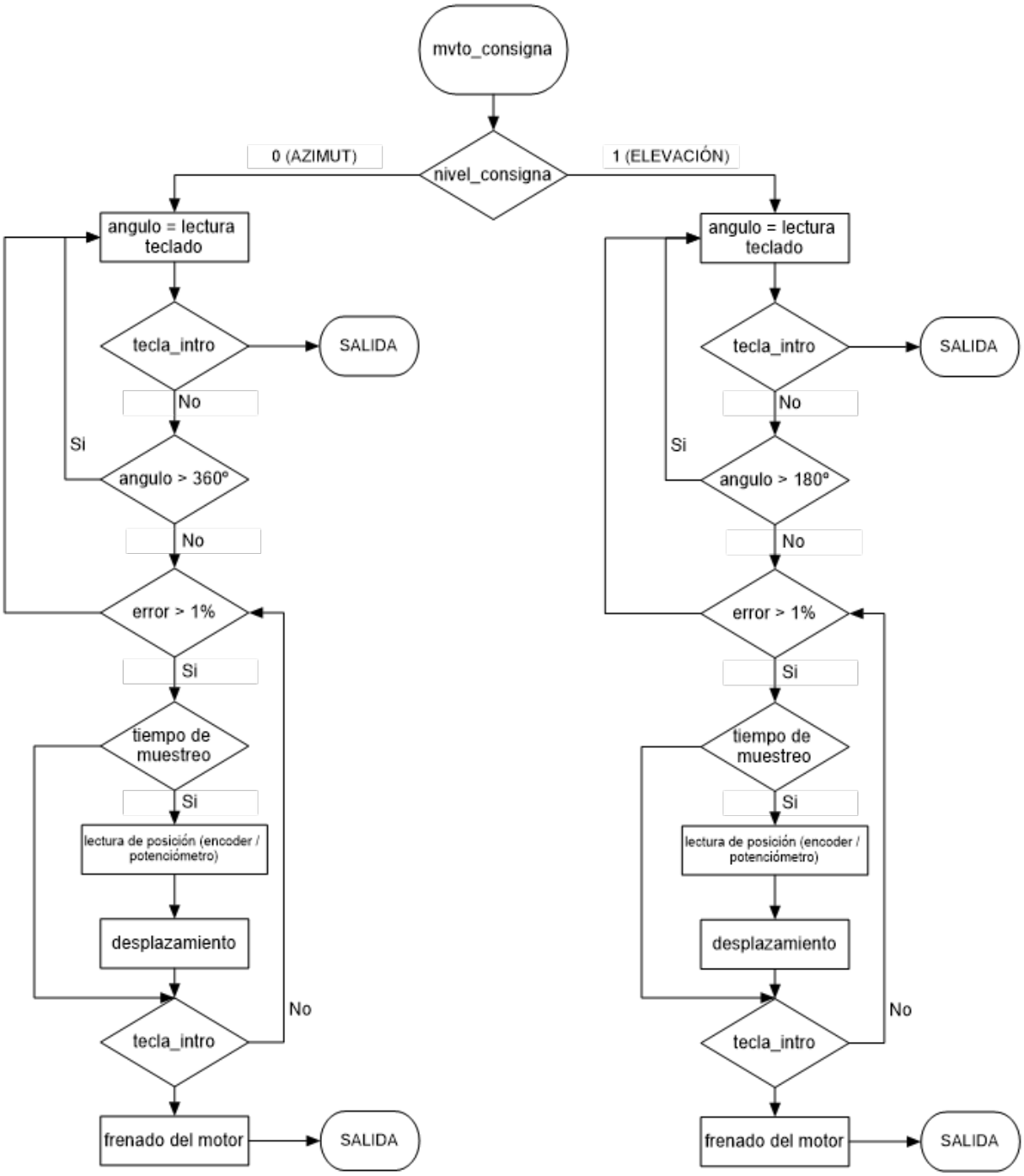


Figura 17.4 – Posicionamiento de motores por consigna

Esta función es la que permite al usuario colocar un determinado motor en la posición deseada introduciendo la consigna por teclado.

La variable *nivel_consigna* al inicio del flujograma representa qué motor es el que se ha de posicionar.

La primera acción será obtener la lectura del teclado. Si durante dicha lectura el usuario

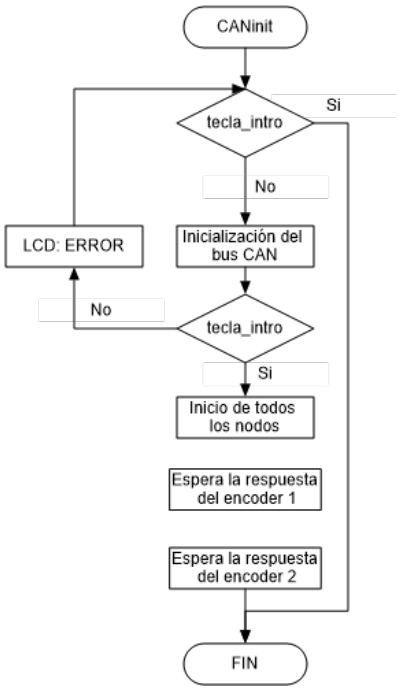
presiona la tecla central, el programa regresa el menú anterior (dónde se selecciona qué motor se desea desplazar). Si la consigna introducida no es válida, se solicitará una nueva consigna.

Una vez obtenida la consigna, se comprueba si el motor se encuentra o no en esa posición. Si la posición es suficientemente próxima a la consigna, no se moverá el motor y se solicitará una nueva consigna. De lo contrario, comenzará el movimiento del motor (controlado por la función desplazamiento, descrita anteriormente), cumpliendo con el tiempo de muestreo.

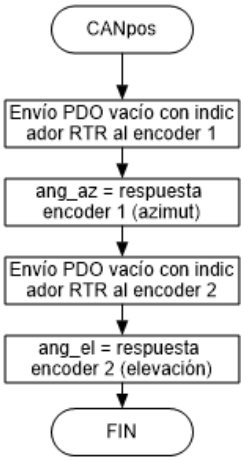
Si durante el movimiento del motor se presiona la tecla central de la botonera, se detendrán los motores y se regresará al menú anterior.

17.5. Flujograma de comunicación con los encoder

En el anexo 12 se realiza una descripción detallada del diálogo que se debe establecer entre los encoder y el microcontrolador. En primer lugar se inicializa el bus y los encoders mediante la función *CANinit()*, desde el setup, y posteriormente, cunado sea necesaria la lectura de la posición, se utiliza *CANpos()*. Este diálogo queda reflejado de manera más visual en los siguientes flujogramas.



(a) Inicialización del bus y de los encoders



(b) Comunicación y recepción de la posición de los encoders

Figura 17.5 – Comunicaciones CANopen

17.6. Cálculos keplerianos

El primer paso se realiza en el setup, donde se guardan los datos TLE. Posteriormente, en la función *automatico()* se llama a la función *calc_kepler()*, que devolverá los ángulos AZ y EL.

En los siguientes flujogramas se muestran funciones y procesos previamente explicadas en los anexos 16 y 15.

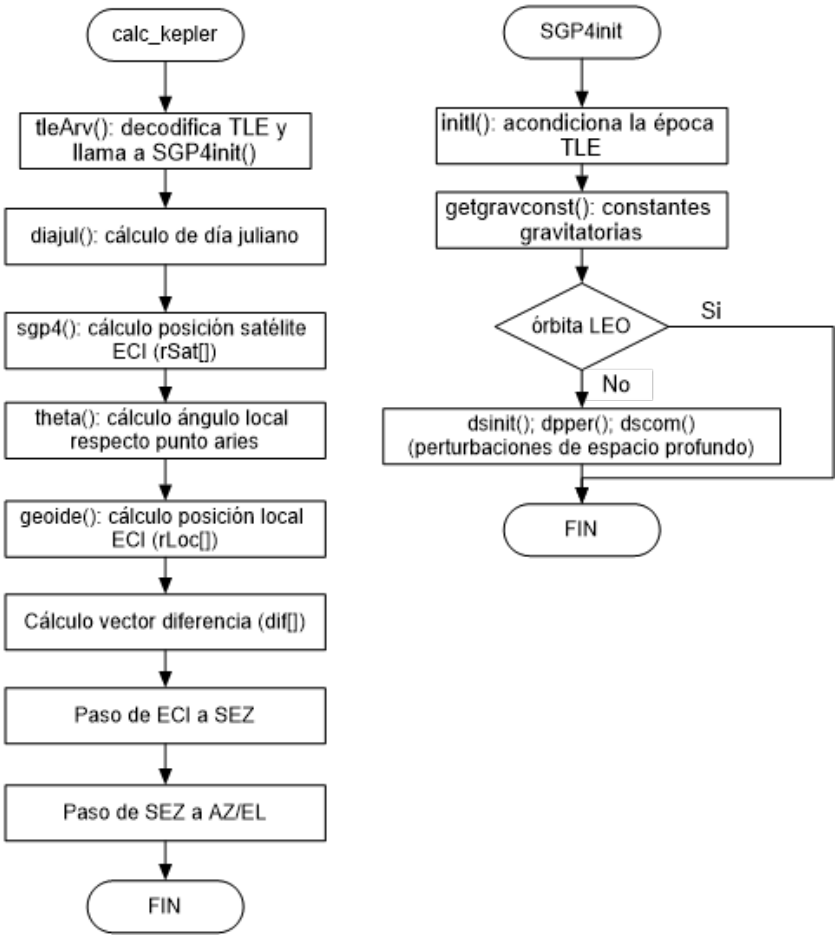


Figura 17.6 – Cálculos keplerianos

18 CÓDIGOS DE PROGRAMACIÓN

18.1. Código de comunicación simple con encoder

```
#include <SPI.h>
#include <mcp_can.h>


const int SPI_CS_PIN = 10;

MCP_CAN CAN(SPI_CS_PIN)

void setup()
{
    unsigned char stmp[2] = {1, 0};
    unsigned long canId;
    unsigned char len;
    unsigned char buf[8];

    Serial.begin(115200);

//---------------------------------------------------------------------
//                  Inicializacion del bus a 20 kbaudios
//---------------------------------------------------------------------

    while (CAN_OK != CAN.begin(CAN_20KBPS))
    {
        Serial.println("Error en la inicializacion del CAN BUS");
        Serial.println("Inicializa el bus de nuevo");
        delay(100);
    }
    Serial.println("Inicializacion del CAN BUS: OK!");
    Serial.println("En espera del cambio del voltaje OFF -> ON");

    while(CAN_MSGAVAIL != CAN.checkReceive());


//---------------------------------------------------------------------
//              Respuesta en el paso de ON-->OFF
//---------------------------------------------------------------------
    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        Serial.println("--------------------Setup----------------------");
        while(CAN_MSGAVAIL == CAN.checkReceive())
        {

            while((canId = CAN.getCanId())!= 0x701)
            {
                CAN.readMsgBuf(&len, buf);
            }
            Serial.println("COB-ID \t Databyte 0 \t Databyte 1" );
            Serial.print("0x");
            Serial.print(canId, HEX);
            Serial.print("    \t");
            for(int i = 0; i<len; i++)
            {
                Serial.print(buf[i], HEX);
                Serial.print("\t\t");
            }
            Serial.println();
            Serial.println("-----------------------------------------");
            Serial.flush();

            delay(50);
        }
    }

//---------------------------------------------------------------------
//                          START ALL NODES
//---------------------------------------------------------------------

    stmp[1] = 0x00;
    stmp[0] = 0x01;
    CAN.sendMsgBuf(0x00, CAN_STDID, 2, stmp);
    delay(50);

    while(CAN_MSGAVAIL != CAN.checkReceive());
```

```
74  //-------------------------------------------------------------------
75  //                          Respuesta PDO1
76  //-------------------------------------------------------------------
77      if(CAN_MSGAVAIL == CAN.checkReceive())
78      {
79          while(CAN_MSGAVAIL == CAN.checkReceive())
80          {
81
82              while((canId = CAN.getCanId())!= 0x181)
83              {
84                  CAN.readMsgBuf(&len, buf);
85              }
86              Serial.println("COB-ID \t Databyte 0 \t Databyte 1" );
87              Serial.print("0x");
88              Serial.print(canId, HEX);
89              Serial.print("    \t");
90              for(int i = 0; i<len; i++)
91              {
92                  Serial.print(buf[i], HEX);
93                  Serial.print("\t\t");
94              }
95              Serial.println();
96              Serial.flush();
97
98              delay(50);
99          }
100     }
101     Serial.println("--------------------Fin Setup-------------------");
102
103 }
104
105
106
107 void loop()
108 {
109     unsigned char len = 0;
110     unsigned char buf[8];
111     float valor=9000,valor_anterior;
112     unsigned long canId = CAN.getCanId();
113
114
115     while(1)
116     {
117         CAN.sendMsgBuf(0x80, 0, 0,0);   // envio del objeto SYNC
118         delay(50);
119
120
121 //-------------------------------------------------------------------
122 //                          Respuesta PDO2
123 //-------------------------------------------------------------------
124         if(CAN_MSGAVAIL == CAN.checkReceive())
125         {
126
127             while(CAN_MSGAVAIL == CAN.checkReceive())
128             {
129
130                 while((canId = CAN.getCanId())!= 0x281)
131                 {
132                     CAN.readMsgBuf(&len, buf);
133                 }
134
135
136                 valor_anterior=valor;
137                 valor=(buf[0]+256*buf[1])*(360.0/4095.0);
138                 // El buffer son 8 bytes, pero el TBN
139                 // me envia la posicion solo en 2, lo
140                 // dice el manual de usuario en pagina
141                 // 8/22 y la propia hoja de caracteristicas.
142                 if(valor_anterior!=valor)
143                 {
144
145
146                     Serial.println("COB-ID \t Databyte 0 \t Databyte 1" );
```

```
                    Serial.print("0x");
                    Serial.print(canId, HEX);
                    Serial.print("    \t");
                    for(int i = 0; i<len; i++)
                    {
                        Serial.print(buf[i], HEX);
                        Serial.print("\t\t");
                    }
                    Serial.println();
                    Serial.print("Posicion:");
                    Serial.println(valor);
                    Serial.println("----------------------------------------");

                }

                delay(100);
            }
        }
    }


}


```

18.2. Código de cambio de dirección del encoder

```
#include <SPI.h>
#include <mcp_can.h>


//const int SPI_CS_PIN = 9;
const int SPI_CS_PIN = 10;

MCP_CAN CAN(SPI_CS_PIN);                                    // Set CS pin

void setup()
{
    Serial.begin(115200);
/*-------------------------------------------------------
                Inicializacion del bus
-------------------------------------------------------*/

    while (CAN_OK != CAN.begin(CAN_20KBPS))              // init can bus : baudrate
    = 20k
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println(" Init CAN BUS Shield again");
        delay(100);
    }
    Serial.println("CAN BUS Shield init ok!");
}



bool ini=1; // Variable con valor 1 uncamente en la primera iteración del bucle
void loop()
{
    byte msg2[2];
    byte msg8[8];
    unsigned char len = 0;
    unsigned char buf[8];


    while(CAN_MSGAVAIL != CAN.checkReceive());  // Espera hasta que se conecte el
    encoder

    CAN.readMsgBuf(&len, buf);

    unsigned long canId = CAN.getCanId();

    Serial.println("--------------------Boot-up del nodo--------------------");
    if(ini == 1)
    {
        Serial.println("-----------------------ID inicial-----------------------");
    }else{
        Serial.println("-----------------------ID nueva-----------------------");
    }
    Serial.print("COB-ID: 0x");
    Serial.println(canId, HEX);
    Serial.print("Databytes: ");


    for(int i = 0; i<len; i++)
    {
        Serial.print(buf[i], HEX);
        Serial.print("\t");
    }
    Serial.println();

    if (ini==1)
    // Cambio de direccion a través de LSS se realiza únicamente en el arranque
    {

/*-------------------------------------------------------
                Start all nodes
-------------------------------------------------------*/
        msg2[0] = 0x01;
        msg2[1] = 0x00;
        CAN.sendMsgBuf(0x0, CAN_STDID, 2, msg2);    //Start all nodes
```

```
        delay(10);
/*-------------------------------------------------------
                RESPUESTA DE LA 0x181
-------------------------------------------------------*/

        CAN.readMsgBuf(&len, buf);

        canId = CAN.getCanId();

        Serial.println("-----------------------------------------------------------");
        Serial.print("COB-ID: 0x");
        Serial.println(canId, HEX);
        Serial.print("Databytes: ");

        for(int i = 0; i<len; i++)
        {
            Serial.print(buf[i], HEX);
            Serial.print("\t");
        }
        Serial.println();
/*-------------------------------------------------------
                Stop all nodes
-------------------------------------------------------*/


        msg2[0] = 0x02;
        msg2[1] = 0x00;
        CAN.sendMsgBuf(0x0, CAN_STDID, 2, msg2);    //Stop all nodes
        delay(10);

/*-------------------------------------------------------
                Manufacturer name
-------------------------------------------------------*/

        //msg8 = {0x40,0x0D,0X01,0X00,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
        //msg8 = {64,13,1,0,0,0,0,0}; // Mensaje en decimal
        msg8[0] = 0x40;
        msg8[1] = 0x0D;
        msg8[2] = 0X01;
        msg8[3] = 0X00;
        msg8[4] = 0X00;
        msg8[5] = 0X00;
        msg8[6] = 0X00;
        msg8[7] = 0X00;
        CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Manufacturer name
        delay(10);
/*-------------------------------------------------------
                Product number
-------------------------------------------------------*/

        //msg8 = {0x41,0x00,0X60,0X00,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
        //msg8 = {65,0,96,0,0,0,0,0};   // Mensaje en decimal
        msg8[0] = 0x41;
        msg8[1] = 0x00;
        msg8[2] = 0X60;
        msg8[3] = 0X00;
        msg8[4] = 0X00;
        msg8[5] = 0X00;
        msg8[6] = 0X00;
        msg8[7] = 0X00;
        CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Product number
        delay(10);
/*-------------------------------------------------------
                Revision number
-------------------------------------------------------*/

        //msg8 = {0x42,0x03,0X00,0X01,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
        //msg8 = {66,3,0,1,0,0,0,0}; // Mensaje en decimal
        msg8[0] = 0x42;
        msg8[1] = 0x03;
        msg8[2] = 0X00;
        msg8[3] = 0X01;
        msg8[4] = 0X00;
```

```
145         msg8[5] = 0X00;
146         msg8[6] = 0X00;
147         msg8[7] = 0X00;
148         CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Revision number
149         delay(10);
150 /*-------------------------------------------------------
151                 S/N:459764=0X0703F4 (serial number)
152 -------------------------------------------------------*/
153
154         //msg8 = {0x43,0x66,0XBE,0X02,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
155         //msg8 = {67,244,3,7,0,0,0,0};  // Mensaje en decimal
156         msg8[0] = 0x43;
157         msg8[1] = 0xF4;
158         msg8[2] = 0X03;
159         msg8[3] = 0X07;
160         msg8[4] = 0X00;
161         msg8[5] = 0X00;
162         msg8[6] = 0X00;
163         msg8[7] = 0X00;
164         CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //S/N:459764=0X0703F4 (serial
            number)
165         delay(10);
166 /*-------------------------------------------------------
167                 SUCCES MESSAGE FROM SENSOR
168 -------------------------------------------------------*/
169
170         CAN.readMsgBuf(&len, buf);
171
172         canId = CAN.getCanId();
173
174         Serial.println("-------------------------------------------------------");
175         Serial.print("COB-ID: 0x");
176         Serial.println(canId, HEX);
177         Serial.print("Databytes: ");
178
179         for(int i = 0; i<len; i++)
180         {
181             Serial.print(buf[i], HEX);
182             Serial.print("\t");
183         }
184         Serial.println();
185         delay(10);
186 /*-------------------------------------------------------------
187                 Node adress 2 programming
188 -------------------------------------------------------------*/
189
190         //msg8 = {0X11,0x02,0X00,0X00,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
191         //msg8 = {17,2,0,0,0,0,0,0}; // Mensaje en decimal
192         msg8[0] = 0X11;
193         msg8[1] = 0x01;
194         msg8[2] = 0X00;
195         msg8[3] = 0X00;
196         msg8[4] = 0X00;
197         msg8[5] = 0X00;
198         msg8[6] = 0X00;
199         msg8[7] = 0X00;
200         CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Node adress 2 programming
201         delay(10);
202
203 /*-------------------------------------------------------------
204                 SUCCES MESSAGE FROM SENSOR
205 -------------------------------------------------------------*/
206         CAN.readMsgBuf(&len, buf);
207
208         canId = CAN.getCanId();
209
210         Serial.println("-------------------------------------------------------");
211         Serial.print("COB-ID: 0x");
212         Serial.println(canId, HEX);
213         Serial.print("Databytes: ");;
214
215         for(int i = 0; i<len; i++)
216         {
```

```
            Serial.print(buf[i], HEX);
            Serial.print("\t");
        }
        Serial.println();
        delay(10);
/*-------------------------------------------------------
                Zero-voltage protected saving
-------------------------------------------------------*/

        //msg8 = {0X17,0x00,0X00,0X00,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
        //msg8 = {23,0,0,0,0,0,0,0};    // Mensaje en decimal
        msg8[0] = 0X17;
        msg8[1] = 0x00;
        msg8[2] = 0X00;
        msg8[3] = 0X00;
        msg8[4] = 0X00;
        msg8[5] = 0X00;
        msg8[6] = 0X00;
        msg8[7] = 0X00;
        CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Zero-voltage protected saving
        delay(10);
/*-------------------------------------------------------
                SUCCES MESSAGE FROM SENSOR
-------------------------------------------------------*/

        CAN.readMsgBuf(&len, buf);

        canId = CAN.getCanId();

        Serial.println("-----------------------------------------------------------");
        Serial.print("COB-ID: 0x");
        Serial.println(canId, HEX);
        Serial.print("Databytes: ");

        for(int i = 0; i<len; i++)
        {
            Serial.print(buf[i], HEX);
            Serial.print("\t");
        }
        Serial.println();
        delay(10);
/*-------------------------------------------------------
                Sensor reseted to LSS-Operating-Mode
-------------------------------------------------------*/

        //msg8 = {0X04,0x00,0X00,0X00,0X00,0X00,0X00,0X00}; // Mensaje en hexadecimal
        //msg8 = {4,0,0,0,0,0,0,0}; // Mensaje en decimal
        msg8[0] = 0X04;
        msg8[1] = 0x00;
        msg8[2] = 0X00;
        msg8[3] = 0X00;
        msg8[4] = 0X00;
        msg8[5] = 0X00;
        msg8[6] = 0X00;
        msg8[7] = 0X00;
        CAN.sendMsgBuf(0x7E5, CAN_STDID, 8, msg8);  //Sensor reseted to
        LSS-Operating-Mode
        delay(10);
/*-------------------------------------------------------
                BOOT-UP NODE WITH NEW NODE ADRESS
-------------------------------------------------------*/

        CAN.readMsgBuf(&len, buf);

        canId = CAN.getCanId();

        Serial.println("-----------------------------------------------------------");
        Serial.print("COB-ID: 0x");
        Serial.println(canId, HEX);
        Serial.print("Databytes: ");

        for(int i = 0; i<len; i++)
        {
```

```
289                 Serial.print(buf[i], HEX);
290                 Serial.print("\t");
291             }
292             Serial.println();
293             delay(10);
294     //-----------------------------------------------------------------
295         }
296         ini=0;
297     }
298
299
300
```

18.3. Código axuiliar para GPS

```ino
#include <SoftwareSerial.h>
/*const int RX = 10;
const int TX = 11;*/
/*const int RX = 3;
const int TX = 4;*/
const int TX = 62; //A8
const int RX = 63; //A9
SoftwareSerial gps(RX, TX);
void setup()
{
   Serial.begin(115200);
   //gps.begin(115200);
   gps.begin(9600);
}
void loop()
{
    if (gps.available())
    {
        char data;
        data = gps.read();
        Serial.print(data);
    }
}
```

18.4. Fichero principal .ino

```
#include <Arduino.h>
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Wire.h>
//#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <mcp_can.h>
#include "sgp4Completo.h"
#include <SoftwareSerial.h>
#include <string.h>
#include <EEPROM.h>


/*const int RX = 0;
const int TX = 1;*/ //Para arduino UNO

const int TX = 62; //A8
const int RX = 63; //A9

SoftwareSerial gps(RX, TX);


// CANopen
#define SPI_CS_PIN 10

MCP_CAN CAN(SPI_CS_PIN);            // pin CS del CAN shield

// Botones
#define izquierda 43
#define intro 39
#define derecha 37
#define arriba 45
#define abajo 41


// Menus
#define numeroMenus 3
//#define numeroNiveis 1
#define num_menus_setup 5

// Constantes de control de velocidad
#define ks 5        // Aceleración
#define kb 0.75     // Constante proporcional
#define vmin 1      // PWM minimo
#define vmax 255    // PWM maximo

// Pines PWM
#define AZ_PWM_pin 7
#define EL_PWM_pin 8

// Pines relés
#define AZcw 3
#define AZbrk 14
#define AZccw 15
#define ELcw 6
#define ELbrk 5
#define ELccw 4

// Variable para activar o no los relés de freno (1 activos, 0 desactivados)
#define freno 0


// Pines lectura analógica
#define AZ_input A0
#define EL_input A1



// Depuración
#define debug_kepler false
#define debug_kepler_loop false     //debug keplerianos en loop()
#define debug_CAN false
#define debug_GPS false
```

```
74  #define debug_easycom false
75  #define debug_auto false
76  #define debug_dsplz false           //debug de f() desplazamiento
77
78
79
80
81  /*-----------------------------------------------------------------------------
82                                  Modos de cálculo de
                                    posición
83  -----------------------------------------------------------------------------
84  *   calc:
85  *           1 => EXTERNO
86  *           0 => AUTÓNOMO
87  *       *DEFAULT => AUTÓNOMO
88  -----------------------------------------------------------------------------*/
89  bool calc = 0;
90
91  /*-----------------------------------------------------------------------------
92                                  Modos de realimentación
93  -----------------------------------------------------------------------------
94  *   REALIMENTACION:
95  *           0 => POTENCIOMETRO
96  *           1 => ENCODER TWK
97  *       *DEFAULT => POTENCIOMETRO
98  -----------------------------------------------------------------------------*/
99  char realimentacion = 0;
100
101 /*-----------------------------------------------------------------------------
102                                     BOTONES
103 -----------------------------------------------------------------------------*/
104 bool tecla_intro = 1, tecla_derecha = 1, tecla_izquierda = 1,
105      tecla_arriba = 1, tecla_abajo = 1;
106
107 //lectura a nivel bajo
108
109 //Tengo que declararlas globales para poder usarlas en las funciones
110
111 /*-----------------------------------------------------------------------------
112                                     MENÚS
113 -----------------------------------------------------------------------------
114 * El sistema de menús involucra, para cada menú,una variable matricial tipo char
115 * en la que se almacena el texto del menú correspondiente y un contador, que se
116 * incrementa o decrementa con cada pulsación de las teclas para desplazarse por
117 * el menú. Este contador sirve como índice de la matriz a la hora de presentar
118 * el texto del menú en la pantalla.
119 *
120 * Se necesita también una variable, nivel, que indicará qué opción de cada menú
121 * se ha seleccionado (seleccionar la opción deseada pulsando la tecla central).
122 *
123 ******************************************************************************
124 *
125 * Ej: menú principal
126 *
127 *               Variable de nivel => nivel_menu
128 *             Índice de la matriz => x
129 *
130 *   Mientras nivel_menu = 0, permanecemos en el entorno de selección de los
131 *   modos de funcionamiento que ofrece el menú principal, incrementando y
132 *   decrementando el índice de la matriz (x) con cada pulsación de las teclas de
133 *   selección. Al pulsar la tecla central (intro), se accede a ese modo de
134 *   funcionamiento al actualizar el valor del nivel del menú, nivel_menu = x + 1
135 *   De esta froma, mientras no se seleccione ningún modo, nivel_menu = 0.
136 *   Si se selecciona el modo de funcionamiento que corresponde con el índice 0
137 *   de la matriz, el nivel del menú será 1.
138 *
139 -----------------------------------------------------------------------------*/
140 /*-----------------------------------------------------------------------------
141                                     MENU SETUP
142 -----------------------------------------------------------------------------*/
143
144 char menu_setup[num_menus_setup][19] = {
145   "1-Software externo",
```

```
    "2-Potenciometro   ",
    "3-Encoder TWK     ",
    "4-Calculo autonomo",
    "5-Actualizar TLE  "};


char nivel_setup = 0;           // Para el nivel de SETUP
char z = 0;


/*-----------------------------------------------------------------------------
                                MENU PRINCIPAL
-----------------------------------------------------------------------------*/

char Menus[numeroMenus][16] = {
  "1-Calibracion",
  "2-Manual     ",
  "3-Automatico "};
char nivel_menu = 0;            // Para el nivel de menu PRINCIPAL
char x = 0;
/*-----------------------------------------------------------------------------
                                MENU MANUAL
-----------------------------------------------------------------------------*/

char submenu_manual[2][16] = {
  "2.1-Consigna   ",
  "2.2-Manual     "};

char nivel_submenu = 0;         //Para el nivel de submenus de MANUAL
char y = 0;

/*-----------------------------------------------------------------------------
               MENU MANUAL CONSIGNA (AZIMUT o ELEVACION)
-----------------------------------------------------------------------------*/

char submenu_consigna[2][16] = {
  "2.1.1-Azimut   ",
  "2.1.2-Elevacion"};

char nivel_consigna = 0;        //Para el nivel de submenus de CONSIGNA
char p = 0;


/*-----------------------------------------------------------------------------
                            Variables de tiempo
-----------------------------------------------------------------------------*/

long t0 = 0,t1 = 0;

/*-----------------------------------------------------------------------------
                                Pantalla
-----------------------------------------------------------------------------*/
LiquidCrystal lcd(23, 25, 27, 29, 31, 33, 35); //(RS,RW, E, D4, D5, D6, D7)

/*-----------------------------------------------------------------------------
                            Teclado matricial
-----------------------------------------------------------------------------*/


/*const byte filasCont = 4;
const byte colCont = 4;*/
#define filasCont 4
#define colCont 4

char teclas[filasCont][colCont] = {
   { '1','2','3','A' },
   { '4','5','6','B' },
   { '7','8','9','C' },
   { '.','0','#','D' }
};

const byte filPin[filasCont] = { 47, 49, 51, 53 };
const byte colPin[colCont] = { 46, 48, 50, 52 };
```

```
219
220  Keypad keypad=Keypad(makeKeymap(teclas),filPin,colPin,filasCont,colCont);
221
222
223  /*-----------------------------------------------------------------------------
224                          Variables máquina de estados
225  -----------------------------------------------------------------------------*/
226
227  /*****************************************************************************
228   Estas variables deben ser globales ya que la maquina de estados se ejectua en
229   bucle, se ejecuta cada vez  que recibe un caracter por puerto serie, pero debe
230   mantener el valor de la ejecucion anterior
231  *****************************************************************************/
232  int s_easycom = 0;
233  bool dat_valido=0;  // Variable utilizada para que la
234                      // trama recibida no viene vacía,
235                      // es decir AZ EL no se decodifica
236
237  int cel = 0;        //contador elevacion
238  int caz = 0;        //contador azimut
239  char azimut[6] = "000000";
240  char elevacion[6] = "000000";
241
242
243  /*-----------------------------------------------------------------------------
244                       Variables de lectura GPS
245  -----------------------------------------------------------------------------*/
246  int s_GPS = 0;  // estado de la maquina de estados
247  int ch = 0;     // contador hora
248  int clat = 0;   // contador latitud
249  int clong = 0;  // contador longitud
250  int cfec = 0;   // contador fecha
251  char hora[10] = "000000.00";
252  char fecha[7] = "0000000";
253  char latitud[11] = "0000.0000N";
254  char longitud[12] = "00000.0000W";
255  char validez;
256
257
258  /*-----------------------------------------------------------------------------
259                          Variables control velocidad
260  -----------------------------------------------------------------------------*/
261
262  float subida_az = 0, bajada_az = 0, subida_el = 0, bajada_el = 0;
263
264  /*-----------------------------------------------------------------------------
265                              DATOS TLE (provisionales)
266  -----------------------------------------------------------------------------*/
267  char linea1[70] =
268  "1 25544U 98067A   19244.80978439  .00023162  00000-0  40601-3 0  9996",
269  linea2[70] =
270  "2 25544  51.6453 341.1922 0008118 354.9234 154.2883 15.50431209187148";
271
272  /*-----------------------------------------------------------------------------
273                                       SETUP
274  -----------------------------------------------------------------------------*/
275
276  bool bios = 0;
277  /*-----------------------------------------------------------------------------
278  *   BIOS:
279  *           1 => permanece en menú del setup
280  *           0 => sale del menú del setup (o no entra)
281  *       *DEFAULT= 0
282  -----------------------------------------------------------------------------*/
283
284  void setup()
285  {
286
287      Serial.begin(9600);
288
289      gps.begin(9600);
290
291      // Botonera
```

```
292         pinMode(intro,INPUT);
293         pinMode(derecha,INPUT);
294         pinMode(izquierda,INPUT);
295         pinMode(arriba,INPUT);
296         pinMode(abajo,INPUT);
297 
298         // Conrol de posición-velocidad
299         pinMode(A0 , INPUT);
300         pinMode(A1 , INPUT);
301         pinMode(8 , OUTPUT);
302         pinMode(7 , OUTPUT);
303 
304         // Relés
305         pinMode(AZcw , OUTPUT);
306         pinMode(AZccw , OUTPUT);
307         pinMode(AZbrk , OUTPUT);
308         pinMode(ELcw , OUTPUT);
309         pinMode(ELccw , OUTPUT);
310         pinMode(ELbrk , OUTPUT);
311 
312         //(control del transistor a nivel bajo)
313         // Freno AZ activado
314         digitalWrite(AZcw,HIGH);
315         digitalWrite(AZccw,HIGH);
316         digitalWrite(AZbrk,HIGH); // NC
317         // Freno EL activado
318         digitalWrite(ELcw,HIGH);
319         digitalWrite(ELccw,HIGH);
320         digitalWrite(ELbrk,HIGH); // NC
321 
322         // Lectura de los datos TLE grabados en la EEPROM
323         int i, j;
324 
325         for(i = 0; i < 69; i++)
326         {
327             linea1[i] = EEPROM.read(i);
328         }
329         linea1[i++] = '\0';
330 
331         for(j = 0; i < 139; i++, j++)
332         {
333             linea2[j] = EEPROM.read(i);
334         }
335         linea2[j] = '\0';
336 
337         /*lcd.init();
338         lcd.backlight();*/
339         lcd.begin(20, 4);
340 
341         lcd.setCursor(3,0);
342         lcd.print("Seguimiento de");
343         lcd.setCursor(6,1);
344         lcd.print("satelites");
345         lcd.setCursor(9,2);
346         lcd.print("UDC");
347 
348         delay(3000);
349 
350         // Comprobación de entrada en Setup
351         tecla_izquierda=digitalRead(izquierda);
352         tecla_derecha=digitalRead(derecha);
353         if(!tecla_izquierda||!tecla_derecha)
354         {
355             delay(300);
356             if(!tecla_derecha&&!tecla_izquierda)
357             {
358                 t0=millis();
359                 while(!tecla_derecha&&!tecla_izquierda)
360                 {
361                     tecla_izquierda=digitalRead(izquierda);
362                     tecla_derecha=digitalRead(derecha);
363                     t1=millis();
364 
```

```
                // Si permanecen pulsados los dos botones más de dos segundos
                // se accede al setup

                if((t1-t0)>2000)
                {
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("SETUP");
                    delay(500);
                }
            }
            if((t1-t0)>2000)
            {
                delay(1000);
                lcd.setCursor(0,0);
                lcd.print("SETUP: CONFIGURACION");
                lcd.setCursor(0,1);
                lcd.print(menu_setup[z]);

                bios=1;
                // Mientras bios = 1, se permanece en el setup. En cuanto se
                // seleccione la configuración deseada --> bios = 0 y se sale
                // del setup

                while(bios)
                {
                    while(nivel_setup == 0 && bios)
                    {
                        sel_setup();
                    }
                    while(nivel_setup == 1 && bios)
                    {
                        CANinit();
                    }
                    while(nivel_setup == 2 && bios)
                    {
                        actualizaTLE();
                    }
                }
            }
        }


    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print(Menus[x]);
    lcd.setCursor(0,0);
    lcd.print("Menu principal");
}

/*-------------------------------------------------------------------------------
                                        LOOP
-------------------------------------------------------------------------------
*
*
*
* Esta función llama a las funciones necesarias según el nivel del menú
* principal
*
*
*
*  entradas:
*           nivel_menu
*
*  salidas:
*
*
*  llamadas a otras funciones:
*           sel_menu
*           calibracion
*           manual
*           automatico
```

```
438   *
439   *
440   *-------------------------------------------------------------------------*/
441   void loop()
442   {
443       while(debug_kepler_loop)
444       {
445           int anho, mes, dia, horas, minut, lat_grad, lon_grad;
446           float lat_min, lon_min,
447                   az, el;
448           double seg;
449           if(leeGPS(anho, mes, dia, horas, minut, seg, lat_grad, lon_grad,
450               lat_min, lon_min))
451           {
452               // La función leer GPS devuelve 1 sólo caundo acaba de actualizarse
453               // Entonces en ese momento calculo los keplerianos, mientras tanto
454               // no
455
456               // Cálculos
457
458               calc_kepler
459               ( anho,  mes,  dia,  horas,  minut,  seg,
460                lat_grad,  lon_grad,  lat_min,  lon_min,
461               az, el);
462           }
463       }
464
465
466       int PWM_az = 0, PWM_el = 0;
467       float ang_az = 0, ang_el = 0, az_old = 0, el_old = 0,
468             pos_az_old = 0, pos_el_old = 0;
469       // ang_az y ang_el = consigna posicion
470       // az_old y el_old = consigna posicion anterior
471       // pos_az_old y pos_el_old = posición anterior
472
473       bool init_apunt_auto = 0;
474
475       /*-------------------------------------------------------------------------
476       * Esta variable bool se envía a la función automatico, donde se iguala a 1
477       * en el momento de obtener la primera consigna de posición (por calculo
478       * externo o autónomo). De esta manera, sólo se iniciará el movimiento de los
479       * motores cuando esta variable indique que se ha obtenido la primera
480       * consigna.
481       * Al salir del modo automático se resetea esta variable.
482       -------------------------------------------------------------------------*/
483
484
485       while(nivel_menu==0)
486       {
487           sel_menu();
488       }
489       while(nivel_menu==1)
490       {
491           calibracion();
492       }
493       while(nivel_menu==2)
494       {
495           manual();
496       }
497       while(nivel_menu==3)
498       {
499           automatico(
500           init_apunt_auto,
501           PWM_az, PWM_el,
502           ang_az, ang_el,
503           az_old, el_old,
504           pos_az_old, pos_el_old);
505       }
506
507   }
508
509
510   /*-------------------------------------------------------------------------
```

```
511                 SELECCIÓN DE LA CONFIGURACIÓN (SETUP)
512 ------------------------------------------------------------------------------
513 *
514 *                       función sel_menu
515 *
516 * Esta función permite desplazarse por el SETUP
517 *
518 *
519 *  entradas:
520 *           lectura hardware de los botones
521 *
522 *  salidas:
523 *           bios
524 *           realimentacion
525 *           calc
526 *
527 *  llamadas a otras funciones:
528 *
529 *
530 *------------------------------------------------------------------------------*/
531 void sel_setup()
532 {
533     tecla_intro = digitalRead(intro);
534     tecla_derecha = digitalRead(derecha);
535     tecla_izquierda = digitalRead(izquierda);
536     if(!tecla_derecha)
537     {
538         delay(300);
539         if(++z > num_menus_setup-1)
540         {
541           z=0;
542         }
543         lcd.clear();
544         lcd.setCursor(0,0);
545         lcd.print("SETUP: CONFIGURACION");
546         lcd.setCursor(0,1);
547         lcd.print(menu_setup[z]);
548     }
549     if(!tecla_izquierda)
550     {
551         delay(300);
552         if(--z < 0)
553         {
554             z = num_menus_setup-1;
555         }
556         lcd.clear();
557         lcd.setCursor(0,0);
558         lcd.print("SETUP: CONFIGURACION");
559         lcd.setCursor(0,1);
560         lcd.print(menu_setup[z]);
561     }
562     if(!tecla_intro)
563     {
564         delay(300);
565
566         t0 = millis();
567         while(!tecla_intro)
568         {
569             t1 = millis();
570             if((t1-t0)>1000)
571             {
572                 lcd.clear();
573                 lcd.setCursor(0,0);
574                 lcd.print("SETUP: CONFIGURACION");
575                 lcd.setCursor(0,1);
576                 lcd.print("Configuracion");
577                 lcd.setCursor(0,2);
578                 lcd.print("realizada");
579                 delay(1500);
580             }
581             tecla_intro = digitalRead(intro);
582         }
583
```

```
584             t1 = millis();
585 
586             if((t1-t0)>1000)
587             {
588                 bios = 0;
589                 // En caso de que la pulsación sea sostenida salgo del setup con la
590                 // última configuración
591             }else
592             {
593             // Si la pulsación es corta, se selecciona el valor de realimentación
594             // que corresponda
595 
596             // En este menu no voy a querer acceder a otras funciones salvo en el
597             // caso de introducir constente de velocidad o cambiar a encoder, asique
598             // no voy a cambiar el nivel del menu salvo en ese caso. Por tanto, el
599             // switch case sera con el indice del menu,y no con el nivel
600                 switch(z)
601                 {
602                     case 0:
603                         calc = 1;
604 
605                         lcd.clear();
606                         lcd.setCursor(0,0);
607                         lcd.print("SETUP: CONFIGURACION");
608                         nivel_setup = 0;
609                         lcd.setCursor(0,1);
610                         lcd.print("Seleccion: ");
611                         lcd.setCursor(0,2);
612                         lcd.print("Software externo");
613                         //-------------------------
614                         delay(1500);
615                         lcd.clear();
616                         lcd.setCursor(0,0);
617                         lcd.print("SETUP: CONFIGURACION");
618                         lcd.setCursor(0,1);
619                         lcd.print(menu_setup[z]);
620                         break;
621                     case 1:
622                         realimentacion = 0;
623                         lcd.clear();
624                         lcd.setCursor(0,0);
625                         lcd.print("SETUP: CONFIGURACION");
626                         lcd.setCursor(0,1);
627                         lcd.print("Seleccion:");
628                         lcd.setCursor(0,2);
629                         lcd.print("Potenciometro");
630                         //-------------------------
631                         // El pin 11 pertenece al pinout del CANshield, como output
632                         // el bus no funciona
633                         pinMode(11 , OUTPUT);
634                         //-------------------------
635                         delay(1500);
636                         lcd.clear();
637                         lcd.setCursor(0,0);
638                         lcd.print("SETUP: CONFIGURACION");
639                         lcd.setCursor(0,1);
640                         lcd.print(menu_setup[z]);
641                         break;
642                     case 2:
643                         realimentacion = 1;
644                         //-------------------------
645                         pinMode(11 , INPUT);
646                         // Así el bus puede funionar de nuevo
647                         //-------------------------
648                         nivel_setup = 1;
649                         lcd.clear();
650                         lcd.setCursor(0,0);
651                         lcd.print("SETUP: CONFIGURACION");
652                         lcd.setCursor(0,1);
653                         lcd.print("Seleccion:");
654                         lcd.setCursor(0,2);
655                         lcd.print("Encoder TBN 12b");
656                         lcd.setCursor(0,3);
```

```
                    lcd.print("CANbus inicializando");
                    delay(1500);

                    break;
                case 3:
                    calc = 0;
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("SETUP: CONFIGURACION");
                    lcd.setCursor(0,1);
                    lcd.print("Seleccion: ");
                    lcd.setCursor(0,2);
                    lcd.print("Calculo autonomo");
                    delay(1500);
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("SETUP: CONFIGURACION");
                    lcd.setCursor(0,1);
                    lcd.print(menu_setup[z]);
                    break;
                case 4:
                    nivel_setup = 2;
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("SETUP: CONFIGURACION");
                    lcd.setCursor(0,1);
                    lcd.print("Actualizar TLE");
                    lcd.setCursor(0,2);
                    lcd.print("Abre monitor serie");
                    lcd.setCursor(0,3);
                    lcd.print("y vuelve a este menu");
                    break;

                default:
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Intro para salir");
                    break;
            }
        }
    }
}

/*-----------------------------------------------------------------------------
                SELECCION DEL MODO DE FUNCIONAMIENTO PRINCIPAL
-----------------------------------------------------------------------------
*
*                       función sel_menu
*
* Esta función permite desplazarse por el menú principal
*
*
*  entradas:
*           lectura hardware de botones
*
*  salidas:
*           nivel_menu
*
*  llamadas a otras funciones:
*
*
*----------------------------------------------------------------------------*/
void sel_menu()
{

    tecla_intro = digitalRead(intro);
    tecla_derecha = digitalRead(derecha);
    tecla_izquierda = digitalRead(izquierda);


    if(!tecla_derecha)
    {
```

```
730             delay(300);
731             if(++x > numeroMenus-1)
732             {
733                 x = 0;
734             }
735             lcd.setCursor(0,1);
736             lcd.print(Menus[x]);
737         }
738         if(!tecla_izquierda)
739         {
740             delay(300);
741             if(--x<0)
742             {
743                 x = numeroMenus-1;
744             }
745             lcd.setCursor(0,1);
746             lcd.print(Menus[x]);
747         }
748         if(!tecla_intro)
749         {
750             delay(300);
751             nivel_menu = x+1;
752             //porque el indice empieza en cero pero los menus son del 1 al 5
753             switch(x)
754             {
755                 case 0:
756                     lcd.clear();
757                     lcd.setCursor(0,0);
758                     lcd.print("Calibracion");
759                     break;
760                 case 1:
761                     x=0;
762                     lcd.clear();
763                     lcd.setCursor(0,0);
764                     lcd.print("Modo manual");
765                     lcd.setCursor(0,1);
766                     lcd.print(submenu_manual[x]);
767                     break;
768                 case 2:
769                     lcd.clear();
770                     lcd.setCursor(0,0);
771                     lcd.print("Modo automatico");
772 
773                     break;
774                 /*case 3:
775                     lcd.clear();
776                     lcd.setCursor(0,0);
777                     lcd.print("Seleccion de");
778                     lcd.setCursor(0,1);
779                     lcd.print("satelite");
780                     break;
781                 case 4:
782                     lcd.clear();
783                     lcd.setCursor(0,0);
784                     lcd.print("Calculos");
785                     lcd.setCursor(0,1);
786                     lcd.print("keplerianos");
787                     break;*/
788                 default:
789                     lcd.clear();
790                     lcd.setCursor(0,0);
791                     lcd.print("Intro para sair");
792                     break;
793             }
794         }
795     }
796 
797     /*-------------------------------------------------------------------------------
798                     SELECCION DEL MODO DE FUNCIONAMIENTO MANUAL
799     -------------------------------------------------------------------------------
800     *
801     *                           función manual
802     *
```

```
803   * Esta función llama a las funciones necesarias según el nivel del submenú
804   * manual
805   *
806   *
807   *
808   *  entradas:
809   *           lectura hardware de botones
810   *           nivel_menu
811   *           nivel_submenu
812   *
813   *  salidas:
814   *
815   *
816   *  llamadas a otras funciones:
817   *           sel_submenu_manual
818   *           consigna
819   *           flechas
820   *
821   *-------------------------------------------------------------------------*/
822   void manual()
823   {
824       while(nivel_submenu == 0 && nivel_menu == 2)
825       {
826         sel_submenu_manual();             // Selección den el submenú manual
827                                           // (movimiento manual o por consigna)
828       }
829       while(nivel_submenu == 1 && nivel_menu == 2)
830       {
831         consigna();                       // Entra en un submenú para seleccionar
832                                           // el motor que se desea colocar
833       }
834       while(nivel_submenu == 2 && nivel_menu == 2)
835       {
836         flechas();                        // Movimiento de los motores de manera
837                                           // completamente manual
838       }
839   }
840
841   /*-------------------------------------------------------------------------
842                   SELECCION DEL MODO DE FUNCIONAMIENTO MANUAL
843   -------------------------------------------------------------------------
844   *
845   *                       función sel_submenu_manual
846   *
847   * Esta función permite desplazarse por el submenú del menú manual (consgina o
848   * sirectamente manual)
849   *
850   *
851   *  entradas:
852   *           lectura hardware de botones
853   *
854   *  salidas:
855   *           nivel_menu
856   *           nivel_submenu
857   *
858   *  llamadas a otras funciones:
859   *
860   *
861   *-------------------------------------------------------------------------*/
862   void sel_submenu_manual()
863   {
864       tecla_intro=digitalRead(intro);
865       tecla_derecha=digitalRead(derecha);
866       tecla_izquierda=digitalRead(izquierda);
867       if(!tecla_derecha)
868       {
869           delay(300);
870           if(++y>2-1)
871           {
872             y=0;
873           }
874           lcd.setCursor(0,1);
875           lcd.print(submenu_manual[y]);
```

```
        }
        if(!tecla_izquierda)
        {
            delay(300);
            if(--y<0)
            {
              //y=numeroMenus-1;
              y=2-1;
            }
            lcd.setCursor(0,1);
            lcd.print(submenu_manual[y]);
        }
        if(!tecla_intro)
        {
            delay(300);

            t0=millis();
            while(!tecla_intro)
            {
                t1=millis();
                if((t1-t0)>1000)
                {

                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Menu principal");
                    lcd.setCursor(0,1);
                    lcd.print(Menus[x]);
                    delay(300);
                }
                tecla_intro=digitalRead(intro);
            }

            t1=millis();

            if((t1-t0)>1000)
            {
                nivel_menu=0;
                nivel_submenu=0;
                // Salida del menú manual y entrada en menú principal
            }else
            {
                nivel_submenu=y+1;
                switch(y)
                {
                    case 0:
                        lcd.clear();
                        lcd.setCursor(0,0);
                        lcd.print("Modo manual");
                        lcd.setCursor(0,1);
                        lcd.print("Selecciona el angulo");
                        lcd.setCursor(0,2);
                        lcd.print(submenu_consigna[p]);
                        //lcd2.clear();
                        //delay(1500);
                        break;
                    case 1:
                        lcd.clear();
                        lcd.setCursor(0,0);
                        lcd.print("Modo manual");
                        //lcd.clear();
                        lcd.setCursor(0,1);
                        lcd.print("Mantener pulsado");
                        lcd.setCursor(0,2);
                        lcd.print("para mover");
                        delay(1500);
                        lcd.setCursor(0,1);
                        lcd.print("                ");
                        lcd.setCursor(0,2);
                        lcd.print("          ");
                        break;
                    default:
                        lcd.clear();
```

```
                    lcd.setCursor(0,0);
                    lcd.print("Intro para sair");
                    break;
                }
            }
        }
    }

/*-------------------------------------------------------------------------------
                    MOVIMIENTO MANUAL POR BOTONES
-------------------------------------------------------------------------------
*
*                       función flechas
*
* Esta función permite mover los motores de manea completamente manual mientras
* se mantiene presionada la tecla correspondiente
*
*
*  entradas:
*           lectura hardware de los botones
*
*
*  salidas:
*           nivel_submenu
*           accionamiento sobre motores
*
*  llamadas a otras funciones:
*           actuación sobre motores
*
*
*------------------------------------------------------------------------------*/
void flechas()
{
    float pos_az, pos_el;

    tecla_intro = digitalRead(intro);
    tecla_derecha = digitalRead(derecha);
    tecla_izquierda = digitalRead(izquierda);
    tecla_arriba = digitalRead(arriba);
    tecla_abajo = digitalRead(abajo);


    while(!tecla_derecha)
    {
        t1 = millis();
        if((t1-t0) > 250)                       // Tiempo de muestreo
        {
            t0=t1;
            if(realimentacion == 0)
            {
                // Lectura analógica
                pos_az = analogRead(A0)*(360.0/1024.0);         // Posición en °
                pos_el = analogRead(A1)*(180.0/1024.0);         // Posición en °
            }else if(realimentacion == 1)
            {
                // Lectura encoders
                CANpos(pos_az, pos_el);
            }
            lcd.setCursor(0,1);
            lcd.print("AZ: ");
            lcd.setCursor(4,1);
            lcd.print("         ");
            lcd.setCursor(4,1);
            lcd.print(pos_az);
            lcd.setCursor(0,2);
            lcd.print("EL: ");
            lcd.setCursor(4,2);
            lcd.print("         ");
            lcd.setCursor(4,2);
            lcd.print(pos_el);
        }
        tecla_derecha = digitalRead(derecha);
        digitalWrite(AZcw,LOW);
```

```
            digitalWrite(AZccw,HIGH);
            if(freno)
            {
                digitalWrite(AZbrk,LOW);
            }

            analogWrite(AZ_PWM_pin, vmin);
        }

        while(!tecla_izquierda)
        {
            t1 = millis();
            if((t1-t0) > 250)                           // Tiempo de muestreo
            {
                t0=t1;
                if(realimentacion == 0)
                {
                    // Lectura analógica
                    pos_az = analogRead(A0)*(360.0/1024.0);         // Posición en °
                    pos_el = analogRead(A1)*(180.0/1024.0);         // Posición en °
                }else if(realimentacion == 1)
                {
                    // Lectura encoders
                    CANpos(pos_az, pos_el);
                }
                lcd.setCursor(0,1);
                lcd.print("AZ: ");
                lcd.setCursor(4,1);
                lcd.print("         ");
                lcd.setCursor(4,1);
                lcd.print(pos_az);
                lcd.setCursor(0,2);
                lcd.print("EL: ");
                lcd.setCursor(4,2);
                lcd.print("         ");
                lcd.setCursor(4,2);
                lcd.print(pos_el);
            }
            tecla_izquierda = digitalRead(izquierda);
            digitalWrite(AZcw,HIGH);
            digitalWrite(AZccw,LOW);
            if(freno)
            {
                digitalWrite(AZbrk,LOW);
            }
            analogWrite(AZ_PWM_pin, vmin);
        }

        while(!tecla_arriba)
        {
            t1 = millis();
            if((t1-t0) > 250)                           // Tiempo de muestreo
            {
                t0=t1;
                if(realimentacion == 0)
                {
                    // Lectura analógica
                    pos_az = analogRead(A0)*(360.0/1024.0);         // Posición en °
                    pos_el = analogRead(A1)*(180.0/1024.0);         // Posición en °
                }else if(realimentacion == 1)
                {
                    // Lectura encoders
                    CANpos(pos_az, pos_el);
                }
                lcd.setCursor(0,1);
                lcd.print("AZ: ");
                lcd.setCursor(4,1);
                lcd.print("         ");
                lcd.setCursor(4,1);
                lcd.print(pos_az);
                lcd.setCursor(0,2);
                lcd.print("EL: ");
                lcd.setCursor(4,2);
```

```
            lcd.print("         ");
            lcd.setCursor(4,2);
            lcd.print(pos_el);
        }
        tecla_arriba = digitalRead(arriba);
        digitalWrite(ELcw,LOW);
        digitalWrite(ELccw,HIGH);
        if(freno)
        {
            digitalWrite(ELbrk,LOW);
        }

        analogWrite(EL_PWM_pin, vmin);
    }


    while(!tecla_abajo)
    {
        t1 = millis();
        if((t1-t0) > 250)                          // Tiempo de muestreo
        {
            t0=t1;
            if(realimentacion == 0)
            {
                // Lectura analógica
                pos_az = analogRead(A0)*(360.0/1024.0);          // Posición en °
                pos_el = analogRead(A1)*(180.0/1024.0);          // Posición en °
            }else if(realimentacion == 1)
            {
                // Lectura encoders
                CANpos(pos_az, pos_el);
            }
            lcd.setCursor(0,1);
            lcd.print("AZ: ");
            lcd.setCursor(4,1);
            lcd.print("         ");
            lcd.setCursor(4,1);
            lcd.print(pos_az);
            lcd.setCursor(0,2);
            lcd.print("EL: ");
            lcd.setCursor(4,2);
            lcd.print("         ");
            lcd.setCursor(4,2);
            lcd.print(pos_el);
        }
        tecla_abajo = digitalRead(abajo);
        digitalWrite(ELcw,HIGH);
        digitalWrite(ELccw,LOW);
        if(freno)
        {
            digitalWrite(ELbrk,LOW);
        }

        analogWrite(EL_PWM_pin, vmin);
    }

    analogWrite(AZ_PWM_pin, 0);
    analogWrite(EL_PWM_pin, 0);
    digitalWrite(AZcw,HIGH);
    digitalWrite(AZccw,HIGH);
    digitalWrite(AZbrk,HIGH);
    digitalWrite(ELcw,HIGH);
    digitalWrite(ELccw,HIGH);
    digitalWrite(ELbrk,HIGH);


    if(!tecla_intro)
    {
        delay(300);
        nivel_submenu=0;
        lcd.clear();
        lcd.setCursor(0,1);
        lcd.print(submenu_manual[y]);
```

```
1168            //lcd2.clear();
1169            lcd.setCursor(0,0);
1170            lcd.print("Menu manual");
1171        }
1172    }
1173
1174    /*-------------------------------------------------------------------------------
1175                        INTRODUCCIÓN DE LA CONSIGNA MANUALMENTE
1176    -------------------------------------------------------------------------------
1177    *
1178    *                           función consigna
1179    *
1180    * Esta función permite introducir la consigna manualmente con  el teclado y
1181    * desplazar el motor a dicha consigna después de haber seleccionado
1182    * qué motor se desea mover
1183    *
1184    *
1185    *  entradas:
1186    *           nivel_submenu
1187    *           nivel_consigna
1188    *
1189    *
1190    *  salidas:
1191    *
1192    *           nivel_consigna
1193    *
1194    *  llamadas a otras funciones:
1195    *           sel_submenu_consigna
1196    *           mvto_consigna
1197    *
1198    *-------------------------------------------------------------------------------*/
1199    void consigna()
1200    {
1201        while(nivel_consigna == 0 && nivel_submenu == 1)
1202        {
1203            sel_submenu_consigna();
1204        }
1205
1206        // nivel_consigna = 1 -> AZIMUT
1207        // nivel_consigna = 2 -> ELEVACION
1208        while((nivel_consigna == 1 || nivel_consigna == 2) && nivel_submenu == 1)
1209        {
1210            mvto_consigna();
1211        }
1212
1213    }
1214
1215    /*-------------------------------------------------------------------------------
1216
1217    -------------------------------------------------------------------------------
1218    *
1219    *                           función mvto_consigna
1220    *
1221    * Esta función
1222    *
1223    *
1224    *  entradas:
1225    *           nivel_submenu
1226                nivel_consigna
1227                realimentación
1228    *
1229    *  salidas:
1230    *           subida_el
1231    *           bajada_el
1232    *           subida_az
1233    *           bajada_az
1234    *           nivel_consigna
1235    *
1236    *  llamadas a otras funciones:
1237    *
1238    *           CANpos
1239    *           desplazamiento
1240    *
```

```
*-----------------------------------------------------------------------------*/
void mvto_consigna()
{
    float pos_el, pos_az;
    int PWM;

    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("CONSIGNA:");

    // nivel_consigna = 1 -> AZIMUT
    // nivel_consigna = 2 -> ELEVACION

    if(nivel_consigna == 1)
    {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("AZIMUT");
        float angulo = teclado();
        tecla_intro = digitalRead(intro);
        if(!tecla_intro)
        {
            delay(300);
            nivel_consigna = 0;
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Selecciona el angulo");
            lcd.setCursor(0,2);
            lcd.print(submenu_consigna[p]);
        }else if (angulo > 360.0)
        {
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Angulo > 360");
            delay(1000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Consigna: ");
            lcd.setCursor(0,2);
            lcd.print("Azimut: ");
        }else
        {

            if(realimentacion == 0)
            {
                pos_az = analogRead(A0)*(360.0/1024.0);// Posición en °
            }else if(realimentacion == 1)
            {
                CANpos(pos_az, pos_el);
            }
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Consigna: ");
            lcd.setCursor(0,2);
            lcd.print("Azimut: ");

            // Si hay un error mayor de 1% se mueven los motores
            while((abs(angulo - pos_az)*100.0/360.0 > 1) && nivel_consigna == 1)
            {
                t1 = millis();
                if((t1-t0) > 250)
                {
                    t0=t1;
                    if(realimentacion == 0)
                    {
                        pos_az = analogRead(A0)*(360.0/1024.0);
                        // Posición en °
```

```
                    }else if(realimentacion == 1)
                    {
                        CANpos(pos_az, pos_el);
                    }

                    desplazamiento(angulo, pos_az, subida_az, bajada_az,
                        AZ_PWM_pin, AZcw, AZccw, AZbrk, 360.0, PWM);


                    lcd.setCursor(10,1);
                    lcd.print("         ");
                    lcd.setCursor(10,1);
                    lcd.print(angulo);
                    lcd.setCursor(8,2);
                    lcd.print("         ");
                    lcd.setCursor(8,2);
                    lcd.print(pos_az);
                }

                tecla_intro = digitalRead(intro);
                if(!tecla_intro)
                {
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Modo manual");
                    lcd.setCursor(0,1);
                    lcd.print("Frenando motor AZ");

                    while(PWM != 0)
                    {
                        t1 = millis();
                        if((t1-t0) > 250)
                        {
                            t0 = t1;
                            PWM-= ks*255/100;

                            if(PWM <= 0)
                                PWM = 0;

                            analogWrite(AZ_PWM_pin,PWM);
                            // Frenada del motor antes de salir

                        }
                    }
                    // Freno AZ activado
                    digitalWrite(AZcw,HIGH);
                    digitalWrite(AZccw,HIGH);
                    digitalWrite(AZbrk,HIGH); // NC
                    // Reset de variables de control de velocidad
                    subida_az = bajada_az = 0;
                    delay(300);
                    nivel_consigna = 0;
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Modo manual");
                    lcd.setCursor(0,1);
                    lcd.print("Selecciona el angulo");
                    lcd.setCursor(0,2);
                    lcd.print(submenu_consigna[p]);
                }
            }
            subida_az = bajada_az = 0; //Reseteo porque son globales
        }
    }
    if(nivel_consigna == 2)
    {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("ELEVACION");
        float angulo = teclado();
        tecla_intro = digitalRead(intro);
        if(!tecla_intro)
        {
```

```
            delay(300);
            nivel_consigna = 0;
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Selecciona el angulo");
            lcd.setCursor(0,2);
            lcd.print(submenu_consigna[p]);
        }else if (angulo > 180.0)
        {
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Angulo > 180");
            delay(1000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Consigna: ");
            lcd.setCursor(0,2);
            lcd.print("Elevacion: ");
        }else
        {

            if(realimentacion == 0)
            {
                pos_el = analogRead(A1)*(180.0/1024.0);// Posición en °
            }else if(realimentacion == 1)
            {
                CANpos(pos_az, pos_el);
            }

            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Modo manual");
            lcd.setCursor(0,1);
            lcd.print("Consigna: ");
            lcd.setCursor(0,2);
            lcd.print("Elevacion: ");

            // Si hay un error mayor de 1% se mueven los motores
            while((abs(angulo - pos_el)*100/180 > 1) && nivel_consigna == 2)
            {
                t1 = millis();
                if((t1-t0) > 250)
                {
                    t0=t1;
                    if(realimentacion == 0)
                    {
                        pos_el = analogRead(A1)*(180.0/1024.0);
                        // Posición en °
                    }else if(realimentacion == 1)
                    {
                        CANpos(pos_az, pos_el);
                    }

                    desplazamiento(angulo, pos_el, subida_el, bajada_el,
                        EL_PWM_pin, ELcw, ELccw, ELbrk, 180.0, PWM);

                    lcd.setCursor(10,1);
                    lcd.print("         ");
                    lcd.setCursor(10,1);
                    lcd.print(angulo);
                    lcd.setCursor(11,2);
                    lcd.print("         ");
                    lcd.setCursor(11,2);
                    lcd.print(pos_el);
                }

                tecla_intro = digitalRead(intro);
                if(!tecla_intro)
                {
```

```
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Modo manual");
                    lcd.setCursor(0,1);
                    lcd.print("Frenando motor EL");
                    while(PWM != 0)
                    {
                        t1 = millis();
                        if((t1-t0) > 250)
                        {
                            t0 = t1;
                            PWM-= ks*255/100;

                            if(PWM <= 0)
                                PWM = 0;

                            analogWrite(EL_PWM_pin,PWM);
                            // Frenada del motor antes de salir

                        }
                    }
                    // Freno AZ activado
                    digitalWrite(ELcw,HIGH);
                    digitalWrite(ELccw,HIGH);
                    digitalWrite(ELbrk,HIGH); // NC
                    // Reseteo de variables de control de velocidad

                    subida_el = bajada_el = 0;
                    delay(300);
                    nivel_consigna = 0;
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Modo manual");
                    lcd.setCursor(0,1);
                    lcd.print("Selecciona el angulo");
                    lcd.setCursor(0,2);
                    lcd.print(submenu_consigna[p]);
                }
            }
            subida_el = bajada_el = 0; //Reset porque son globales
        }
    }
}

/*-----------------------------------------------------------------------------
                    SELECCIÓN DEL ÁNGULO PARA CONSIGNA
-----------------------------------------------------------------------------
*
*                       función sel_submenu_consigna
*
* Esta función permite esplazarnos por el menú en donde seleccionamos el angulo
* azimut o elevación (para introducir posteriormente la consigna manualmente con
* el teclado)
*
*
*  entradas:
*           lectura hardware de los botones
*
*  salidas:
*           nivel_submenu
*           nivel_consigna
*
*  llamadas a otras funciones:
*
*
*----------------------------------------------------------------------------*/
void sel_submenu_consigna()
{
    tecla_intro = digitalRead(intro);
    tecla_derecha = digitalRead(derecha);
    tecla_izquierda = digitalRead(izquierda);
    if(!tecla_derecha)
    {
```

```
            delay(300);
            if(++p > 2-1)
            {
                p = 0;
            }
            lcd.setCursor(0,2);
            lcd.print(submenu_consigna[p]);
        }
        if(!tecla_izquierda)
        {
            delay(300);
            if(--p<0)
            {
                //y=numeroMenus-1;
                p = 2-1;
            }
            lcd.setCursor(0,2);
            lcd.print(submenu_consigna[p]);
        }
        if(!tecla_intro)
        {
            delay(300);

            t0 = millis();
            while(!tecla_intro)
            {
                t1 = millis();
                if((t1-t0)>1000)
                {
                    nivel_submenu = 0;
                    nivel_consigna = 0;
                    lcd.clear();
                    lcd.setCursor(0,0);
                    lcd.print("Menu manual");
                    lcd.setCursor(0,1);
                    lcd.print(submenu_manual[x]);
                    delay(300);
                }
                tecla_intro = digitalRead(intro);
            }

            t1 = millis();

            if((t1-t0)>1000)
            {
                nivel_consigna = 0;
                nivel_submenu = 0;
                // Salida del menu de consigna y regreso al submenu manual
            }else
            {
                nivel_consigna = p+1;
                // porque el indice empieza en cero pero los menus son del 1 al 5
                switch(p)
                {
                    case 0:
                        lcd.clear();
                        lcd.setCursor(0,0);
                        lcd.print("Modo manual");
                        lcd.setCursor(0,1);
                        lcd.print("CONSIGNA AZIMUT");
                        break;
                    case 1:
                        lcd.clear();
                        lcd.setCursor(0,0);
                        lcd.print("Modo manual");
                        lcd.setCursor(0,1);
                        lcd.print("CONSIGNA ELEVACION");
                        break;
                    default:
                        lcd.clear();
                        lcd.setCursor(0,0);
                        lcd.print("Intro para sair");
                        break;
```

```
            }
        }
    }
}


/*-------------------------------------------------------------------------------
                            Calibracion
-------------------------------------------------------------------------------
*
*                               función calibracion
*
*
*  entradas:
*           nivel_menu
*
*  salidas:
*
*  llamadas a otras funciones:
*           calibrar
*
*------------------------------------------------------------------------------*/

void calibracion()
{
    calibrar();

    nivel_menu=0;
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print(Menus[x]);
    lcd.setCursor(0,0);
    lcd.print("Menu principal");
}

/*-------------------------------------------------------------------------------
        CALIBRACION DE OFFSET Y GANANCIA DEL POTENCIOMETRO ANALOGICO
-------------------------------------------------------------------------------
*
*                               función calibracion
*
* Esta función permite ajustar el offset y ganancia de ambos motores
*
*
*  entradas:
*           lectura hardware de botones
*           lectura analógica de offset y ganancia
*
*  salidas:
*           actuación sobre motores
*
*
*  llamadas a otras funciones:
*
*
*------------------------------------------------------------------------------*/
void calibrar()
{
    float offset,ganancia;

//                              AJUSTE AZIMUT
//-------------------------------------------------------------------------------
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Calibracion");
    lcd.setCursor(0,1);
    lcd.print("AZIMUT");
    lcd.setCursor(0,2);
    lcd.print("Ajuste el offset 0V");
    lcd.setCursor(0,3);
    lcd.print("Offset: ");

    while (digitalRead(intro)) //Mientras no se pulse la tecla central
```

```
1679        {
1680
1681            tecla_derecha = digitalRead(derecha);
1682            tecla_izquierda = digitalRead(izquierda);
1683
1684            while(!tecla_derecha)
1685            {
1686
1687                tecla_derecha = digitalRead(derecha);
1688                digitalWrite(AZcw,LOW);
1689                digitalWrite(AZccw,HIGH);
1690                if(freno)
1691                {
1692                    digitalWrite(AZbrk,LOW);
1693                }
1694
1695                analogWrite(AZ_PWM_pin, vmin);
1696            }
1697
1698            while(!tecla_izquierda)
1699            {
1700
1701                tecla_izquierda = digitalRead(izquierda);
1702                digitalWrite(AZcw,HIGH);
1703                digitalWrite(AZccw,LOW);
1704                if(freno)
1705                {
1706                    digitalWrite(AZbrk,LOW);
1707                }
1708                analogWrite(AZ_PWM_pin, vmin);
1709            }
1710
1711            analogWrite(AZ_PWM_pin, 0);
1712            analogWrite(EL_PWM_pin, 0);
1713            digitalWrite(AZcw,HIGH);
1714            digitalWrite(AZccw,HIGH);
1715            digitalWrite(AZbrk,HIGH);
1716            digitalWrite(ELcw,HIGH);
1717            digitalWrite(ELccw,HIGH);
1718            digitalWrite(ELbrk,HIGH);
1719
1720            offset = analogRead(A0)*(5.0/1023.0);;
1721            t1 = millis();
1722            if((t1-t0)>250)
1723            {
1724                t0 = t1;
1725                lcd.setCursor(8,3);
1726                lcd.print(offset);
1727            }
1728        }
1729        delay(300);
1730
1731        lcd.clear();
1732        lcd.setCursor(0,0);
1733        lcd.print("Calibracion");
1734        lcd.setCursor(0,1);
1735        lcd.print("AZIMUT");
1736        lcd.setCursor(0,2);
1737        lcd.print("Ajuste ganancia 5V");
1738        lcd.setCursor(0,3);
1739        lcd.print("Ganancia: ");
1740
1741        while (digitalRead(intro)) //Mientras no se pulse la tecla central
1742        {
1743            tecla_derecha = digitalRead(derecha);
1744            tecla_izquierda = digitalRead(izquierda);
1745
1746            while(!tecla_derecha)
1747            {
1748
1749                tecla_derecha = digitalRead(derecha);
1750                digitalWrite(AZcw,LOW);
1751                digitalWrite(AZccw,HIGH);
```

```
            if(freno)
            {
                digitalWrite(AZbrk,LOW);
            }

            analogWrite(AZ_PWM_pin, vmin);
        }

        while(!tecla_izquierda)
        {

            tecla_izquierda = digitalRead(izquierda);
            digitalWrite(AZcw,HIGH);
            digitalWrite(AZccw,LOW);
            if(freno)
            {
                digitalWrite(AZbrk,LOW);
            }
            analogWrite(AZ_PWM_pin, vmin);
        }

        analogWrite(AZ_PWM_pin, 0);
        analogWrite(EL_PWM_pin, 0);
        digitalWrite(AZcw,HIGH);
        digitalWrite(AZccw,HIGH);
        digitalWrite(AZbrk,HIGH);
        digitalWrite(ELcw,HIGH);
        digitalWrite(ELccw,HIGH);
        digitalWrite(ELbrk,HIGH);

        ganancia = analogRead(A0)*(5.0/1023.0);;
        t1 = millis();
        if((t1-t0)>250)
        {
            t0=t1;
            lcd.setCursor(10,3);
            lcd.print(ganancia);
        }
    }
    delay(300);


//                          AJUSTE ELEVACIÓN
//-------------------------------------------------------------------------
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Calibracion");
    lcd.setCursor(0,1);
    lcd.print("ELEVACION");
    lcd.setCursor(0,2);
    lcd.print("Ajuste offset 0V");
    lcd.setCursor(0,3);
    lcd.print("Offset: ");

    while (digitalRead(intro)) //Mientras no se pulse la tecla central
    {
        tecla_arriba = digitalRead(arriba);
        tecla_abajo = digitalRead(abajo);

        while(!tecla_arriba)
        {

            tecla_arriba = digitalRead(arriba);
            digitalWrite(ELcw,LOW);
            digitalWrite(ELccw,HIGH);
            if(freno)
            {
                digitalWrite(ELbrk,LOW);
            }

            analogWrite(EL_PWM_pin, vmin);
        }

```

```
1825            while(!tecla_abajo)
1826            {
1827
1828                tecla_abajo = digitalRead(abajo);
1829                digitalWrite(ELcw,HIGH);
1830                digitalWrite(ELccw,LOW);
1831                if(freno)
1832                {
1833                    digitalWrite(ELbrk,LOW);
1834                }
1835                analogWrite(EL_PWM_pin, vmin);
1836            }
1837
1838            analogWrite(AZ_PWM_pin, 0);
1839            analogWrite(EL_PWM_pin, 0);
1840            digitalWrite(AZcw,HIGH);
1841            digitalWrite(AZccw,HIGH);
1842            digitalWrite(AZbrk,HIGH);
1843            digitalWrite(ELcw,HIGH);
1844            digitalWrite(ELccw,HIGH);
1845            digitalWrite(ELbrk,HIGH);
1846
1847            offset = analogRead(A1)*(5.0/1023.0);;
1848            t1 = millis();
1849            if((t1-t0)>250)
1850            {
1851                t0=t1;
1852                lcd.setCursor(8,3);
1853                lcd.print(offset);
1854            }
1855        }
1856        delay(300);
1857
1858        lcd.clear();
1859        lcd.setCursor(0,0);
1860        lcd.print("Calibracion");
1861        lcd.setCursor(0,1);
1862        lcd.print("ELEVACION");
1863        lcd.setCursor(0,2);
1864        lcd.print("Ajuste ganancia 5V");
1865        lcd.setCursor(0,3);
1866        lcd.print("Ganancia: ");
1867
1868        while (digitalRead(intro)) //Mientras no se pulse la tecla central
1869        {
1870            tecla_arriba = digitalRead(arriba);
1871            tecla_abajo = digitalRead(abajo);
1872
1873            while(!tecla_arriba)
1874            {
1875
1876                tecla_arriba = digitalRead(arriba);
1877                digitalWrite(ELcw,LOW);
1878                digitalWrite(ELccw,HIGH);
1879                if(freno)
1880                {
1881                    digitalWrite(ELbrk,LOW);
1882                }
1883
1884                analogWrite(EL_PWM_pin, vmin);
1885            }
1886
1887            while(!tecla_abajo)
1888            {
1889
1890                tecla_abajo = digitalRead(abajo);
1891                digitalWrite(ELcw,HIGH);
1892                digitalWrite(ELccw,LOW);
1893                if(freno)
1894                {
1895                    digitalWrite(ELbrk,LOW);
1896                }
1897                analogWrite(EL_PWM_pin, vmin);
```

```
1898            }
1899
1900            analogWrite(AZ_PWM_pin, 0);
1901            analogWrite(EL_PWM_pin, 0);
1902            digitalWrite(AZcw,HIGH);
1903            digitalWrite(AZccw,HIGH);
1904            digitalWrite(AZbrk,HIGH);
1905            digitalWrite(ELcw,HIGH);
1906            digitalWrite(ELccw,HIGH);
1907            digitalWrite(ELbrk,HIGH);
1908
1909            ganancia=analogRead(A1)*(5.0/1023.0);;
1910            t1 = millis();
1911            if((t1-t0)>250)
1912            {
1913                t0 = t1;
1914                lcd.setCursor(10,3);
1915                lcd.print(ganancia);
1916            }
1917        }
1918        delay(300);
1919    }
1920
1921    /*-------------------------------------------------------------------------------
1922                                AUTOMÁTICO
1923    -------------------------------------------------------------------------------
1924    *
1925    *                                función automatico
1926    *
1927    * Esta función gestiona el funcionamiento automático del sistema. En función de
1928    * la consigna establecida por el software externo o el cálculo autónomo, se
1929    * calcula una señal PWM para cada motor de la antena en función de la posición
1930    * actual, la posición de consigna y la velocidad
1931    *
1932    *
1933    *  entradas:
1934    *           ang_az
1935    *           ang_el
1936    *           nivel_menu
1937    *           calc --> indica si se utiliza software autónomo o externo
1938    *                   0 = externo
1939    *                   1 = autónomo
1940    *           realimentación --> indica el tipo de realimentación
1941    *                   0 = potencióemtros
1942    *                   1 = encoders
1943    *           init_apunt_auto --> explicado en loop
1944    *
1945    *  salidas:
1946    *           ang_az --> consigna az
1947    *           ang_el --> consigna el
1948    *           subida --> control velocidad
1949    *           bajada --> control velocidad
1950    *           az_old --> consigna az anterior
1951    *           pos_az_old --> posición az anterior
1952    *           el_old --> consigna el anterior
1953    *           pos_el_old --> posición el anterior
1954    *           PWM --> PWM
1955    *           nivel_menu -->
1956    *
1957    *  llamadas a otras funciones:
1958    *           deco_easycomm
1959    *           desplazamiento
1960    *           cadena2float
1961    *           leeGPS
1962    *           calc_kepler
1963    *           CANpos
1964    *
1965    *-------------------------------------------------------------------------------*/
1966    void  automatico
1967    (bool &init_apunt_auto, int &PWM_az, int &PWM_el,
1968    float &ang_az, float &ang_el, float &az_old, float &el_old, float &pos_az_old,
1969    float &pos_el_old)
1970    {
```

```
1971        float pos_az, pos_el;
1972
1973        bool cambio_sentido = 0;
1974
1975        // Variables de lectura GPS para cálculos keplerianos
1976
1977        int anho, mes, dia, horas, minut, lat_grad, long_grad;
1978        float lat_min, long_min;
1979        double seg;
1980
1981
1982        if(debug_auto)
1983        {
1984            pinMode(A3, INPUT);
1985
1986            ang_az = analogRead(A3)*(360.0/1024.0);
1987            ang_el = analogRead(A3)*(180.0/1024.0);
1988        }
1989        //-------------------------------------
1990
1991        // Lectura serie de la posición del satélite
1992        if(calc == 1)
1993        {
1994            if (Serial.available())
1995            {
1996
1997                if (deco_easycomm(Serial.read()))
1998                {
1999                    char i;
2000
2001                    az_old = ang_az,
2002                    el_old = ang_el;
2003
2004
2005                    ang_az = cadena2float(azimut);
2006                    ang_el = cadena2float(elevacion);
2007
2008                    for (i=2;i<8;i++) azimut[i]='0';
2009                    for (i=2;i<8;i++) elevacion[i]='0';
2010
2011                    if(!init_apunt_auto)
2012                    {
2013                        init_apunt_auto = 1;
2014                        // se recibe la primera consigna y se desactivan los frenos
2015                        // Freno AZ desactivado
2016                        digitalWrite(AZcw,HIGH);
2017                        digitalWrite(AZccw,HIGH);
2018                        if(freno)
2019                        {
2020                            digitalWrite(AZbrk,LOW);    // NC
2021                        }
2022
2023                        // Freno EL desactivado
2024                        digitalWrite(ELcw,HIGH);
2025                        digitalWrite(ELccw,HIGH);
2026                        if(freno)
2027                        {
2028                            digitalWrite(ELbrk,LOW);    // NC
2029                        }
2030                        // Al activar el rele freno (LOW), se desactiva el freno
2031                    }
2032                }
2033            }
2034        }else
2035        {
2036
2037            // Cálculo autónomo (keplerianos)
2038
2039            if(leeGPS(anho, mes, dia, horas, minut, seg, lat_grad, long_grad,
2040                lat_min, long_min))
2041            {
2042                // La función leeGPS devuelve '1' sólo caundo acaba de actualizarse
2043                // En ese momento calculo los keplerianos, mientras tanto no
```

```
2044                az_old = ang_az,
2045                el_old = ang_el;
2046
2047                // Calculos keplerianos
2048
2049                calc_kepler
2050                ( anho, mes, dia, horas, minut, seg,
2051                 lat_grad, long_grad, lat_min, long_min,
2052                 ang_az, ang_el);
2053
2054                if(!init_apunt_auto)
2055                {
2056                    init_apunt_auto = 1;        // se recibe la primera consigna
2057                }
2058
2059            }
2060        }
2061
2062        if(ang_az < 0)
2063            ang_az = 0;
2064        if(ang_el < 0)
2065            ang_el = 0;
2066
2067        if(ang_az > 360)
2068            ang_az = 360;
2069        if(ang_el > 180)
2070            ang_el = 180;
2071
2072        t1 = millis();
2073        if((t1-t0) > 250)                       // Tiempo de muestreo
2074        {
2075            t0=t1;
2076
2077
2078            //-------------------------------
2079            if(debug_auto)
2080            {
2081                // Potenciómetro para simular consigna
2082                ang_az = analogRead(A3)*(360.0/1024.0);
2083                ang_el = analogRead(A3)*(180.0/1024.0);
2084            }
2085
2086            //------------------------------ lectura de posición
2087
2088
2089            if(realimentacion == 0)
2090            {
2091                // Lectura analógica
2092                pos_az = analogRead(A0)*(360.0/1024.0);             // Posición en °
2093                pos_el = analogRead(A1)*(180.0/1024.0);             // Posición en °
2094            }else if(realimentacion == 1)
2095            {
2096                // Lectura encoders
2097                CANpos(pos_az, pos_el);
2098            }
2099
2100
2101            // Sólo se calcula el PWM si no se detecta cambio de sentido
2102            // Si la posición se encuetra entre la consigna nueva y la vieja hay un
2103            // cambio de sentido
2104            // Si la consigna se encuentra entre la posición actual y la vieja
2105            // también hay un cambio de sentido
2106
2107
2108    //-------------------------------------------------------------------------
2109    //------------------detección de cambio de sentido-------------------------
2110    //-------------------------------------------------------------------------
2111            //Cambio de sentido AZ (cambio de sonsigna || cambio de posición)
2112            if(((ang_az - pos_az)*(az_old - pos_az) < 0) || ((pos_az - ang_az)*
2113            (pos_az_old - ang_az) < 0))
2114            {
2115                //Si el error es > 1%
2116                if(abs(ang_az - pos_az)*100.0/360.0 > 1)
```

```
            {
                cambio_sentido = 1;
            }
        }else if(((ang_el - pos_el)*(el_old - pos_el) < 0) || ((pos_el - ang_el)
            *(pos_el_old - ang_el) < 0))
        //Cambio de sentido EL (cambio de sonsigna || cambio de posición)
        {
            //Si el error es > 1%
            if(abs(ang_el - pos_el)*100.0/180.0 > 1)
            {
                cambio_sentido = 1;
            }
        }else
        {
            cambio_sentido = 0;
        }


        // Se calcula el PWM si no se detecta cambio de sentido y si se ha
        // obtenido la primera consigna
        if(cambio_sentido == 0 && init_apunt_auto == 1)
        {

            lcd.setCursor(0,0);
            lcd.print("Modo automatico");
            // Consigna
            lcd.setCursor(0,1);
            lcd.print("CONSIGNA");
            lcd.setCursor(8,1);         lcd.print("   ");
            lcd.setCursor(0,2);
            lcd.print("AZ:");
            lcd.setCursor(0,3);
            lcd.print("EL:");

            lcd.setCursor(3,2);         lcd.print("       ");
            lcd.setCursor(3,2);         lcd.print(ang_az);
            lcd.setCursor(3,3);         lcd.print("       ");
            lcd.setCursor(3,3);         lcd.print(ang_el);
            // Posición
            lcd.setCursor(11,1);
            lcd.print("POSICION");
            lcd.setCursor(11,2);
            lcd.print("AZ:");
            lcd.setCursor(19,2);
            lcd.print(" ");

            lcd.setCursor(11,3);
            lcd.print("EL:");

            lcd.setCursor(14,2);        lcd.print("      ");
            lcd.setCursor(14,2);        lcd.print(pos_az);
            lcd.setCursor(14,3);        lcd.print("      ");
            lcd.setCursor(14,3);        lcd.print(pos_el);

            desplazamiento( ang_az, pos_az, subida_az, bajada_az,
                AZ_PWM_pin, AZcw, AZccw, AZbrk, 360.0, PWM_az);
            desplazamiento( ang_el, pos_el, subida_el, bajada_el,
                EL_PWM_pin, ELcw, ELccw, ELbrk, 180.0, PWM_el);
        }else if(cambio_sentido == 1)
        {

            if(debug_auto)
            {
                Serial.println("Cambio de sentido = 1");
                Serial.print("az_old= ");
                Serial.println(az_old);
                Serial.print("el_old= ");
                Serial.println(el_old);
                Serial.print("------------------------");
            }
        }

        if(init_apunt_auto == 0)
```

```
2190            {
2191                lcd.setCursor(0,1);
2192                lcd.print("Esperando posicion  ");
2193                lcd.setCursor(0,2);
2194                lcd.print("de apuntamiento     ");
2195                lcd.setCursor(0,3);
2196                lcd.print("                    ");
2197                lcd.setCursor(0,4);
2198                lcd.print("                    ");
2199            }
2200
2201            // Guardado posición actual de los motores para que en el siguiente
2202            // muestreo sean la posición anterior
2203            pos_az_old = pos_az;
2204            pos_el_old = pos_el;
2205        }
2206
2207        // Letura de la tecla al final para no ejecutar el resto del código después
2208        // de pulsar
2209        tecla_intro = digitalRead(intro);
2210        if(!tecla_intro || cambio_sentido == 1)
2211        {
2212            if(init_apunt_auto == 1)
2213            {
2214
2215                lcd.clear();
2216                lcd.setCursor(0,0);
2217                lcd.print("Modo automatico");
2218                lcd.setCursor(0,1);
2219                lcd.print("Frenando motores");
2220                if(cambio_sentido)
2221                // Si el motivo de parar motores fue cambio de sentido
2222                {
2223                    // Al detectar un cambio de sentido, se resetea la variable
2224                    // init_apunt_auto para que solamente escriba en la LCD la
2225                    // primera iteración mientras se frenan los motores, evitando
2226                    // el parpadeo de la LCD
2227                    cambio_sentido = 0;
2228                    init_apunt_auto = 0;
2229                    lcd.setCursor(0,2);
2230                    lcd.print("Cambio de sentido");
2231                    lcd.setCursor(0,3);
2232                    lcd.print("detectado");
2233                }
2234            }
2235
2236            // No hace falta comprobar hacia que lado estaban girando (cw/ccw)
2237            // En la funcion desplazamiento() quedaron activados los relés adecuados
2238            while(PWM_az != 0 || PWM_el != 0)
2239            {
2240                t1 = millis();
2241                if((t1-t0) > 250)
2242                {
2243                    t0 = t1;
2244                    PWM_az -= ks*255/100;
2245                    PWM_el -= ks*255/100;
2246
2247                    if(PWM_az <= 0)
2248                    {
2249                        PWM_az = 0;
2250                        // Freno AZ activado
2251                        digitalWrite(AZcw,HIGH);
2252                        digitalWrite(AZccw,HIGH);
2253                        digitalWrite(AZbrk,HIGH); // NC
2254                    }
2255
2256                    if(PWM_el <= 0)
2257                    {
2258                        PWM_el = 0;
2259                        // Freno EL activado
2260                        digitalWrite(ELcw,HIGH);
2261                        digitalWrite(ELccw,HIGH);
2262                        digitalWrite(ELbrk,HIGH); // NC
```

```
                }


                analogWrite(AZ_PWM_pin,PWM_az);
                // Frenada del motor antes de salir
                analogWrite(EL_PWM_pin,PWM_el);
                // Frenada del motor antes de salir

            }
        }
        subida_az = bajada_az = subida_el = bajada_el = 0;
        analogWrite(AZ_PWM_pin,0);
                // Frenada del motor antes de salir
        analogWrite(EL_PWM_pin,0);
                // Frenada del motor antes de salir

        if(!tecla_intro) // Si el motivo de parar motores es pulsación intro
        {
            nivel_menu = 0;
            init_apunt_auto = 0;
            lcd.clear();
            lcd.setCursor(0,1);
            lcd.print(Menus[x]);
            lcd.setCursor(0,0);
            lcd.print("Menu principal");
            delay(300);

        }
    }

}

/*-----------------------------------------------------------------------------
                        CONTROL DE VELOCIDAD
-----------------------------------------------------------------------------
*
*                               función desplazamiento
*
* Esta función realiza el control de velocidad y posición del motor
*
*
*  entradas:
*           consigna
*           pos --> posición actual del motor
*           subida --> velocidad de la rampa aceleración de la muestra anterior
*           bajada --> velocidad de la rampa deceleración de la muestra anterior
*           PWMpin --> pin PWM para el motor
*           cw --> pin de sentido horario
*           ccw --> pin de sentido antihorario
*           brk --> pin de freno
*           fondo_escala --> 360° en AZ y 180° en EL
*
*  salidas:
*           subida --> velocidad de la rampa aceleración actualizada
*           bajada --> velocidad de la rampa deceleración actualizada
*           PWM --> velocidad PWM calculada
*
*  llamadas a otras funciones:
*
*
*-----------------------------------------------------------------------------*/
void desplazamiento(float consigna, float pos, float &subida, float &bajada,
const int PWMpin, const int cw, const int ccw, const int brk,
float fondo_escala, int &PWM )
{
    float error;
    float bajada_old;
    //float PWM;

    bajada_old = bajada;
//                                  Cálculos
//-----------------------------------------------------------------------------
```

```
2335        error = abs(consigna - pos)*100.0/fondo_escala; // Error en %
2336
2337    //      AZIMUT
2338    //-------------------------------------------------------------------------------
2339
2340        if(error > 1) // Margen de frenada (%)
2341        {
2342            //Aceleración y frenada
2343            // AZIMUT
2344            subida += ks;           // Rampa de aceleración constante
2345            if(subida > 100)
2346                subida = 100;
2347            bajada = kb*error;// Rampa deceleración (regulador proporcional)
2348            // Control de pendiente de la deceleracion
2349
2350            //-----------------------------------------------------------------
2351            if(debug_dsplz)
2352            {
2353                Serial.println("bajada inicial:");
2354                Serial.println(bajada);
2355                Serial.print("bajada_old= ");
2356                Serial.println(bajada_old);
2357
2358            }
2359            //-----------------------------------------------------------------
2360
2361            if(bajada_old - bajada > ks)
2362            {
2363                bajada = bajada_old - ks;
2364                if(debug_dsplz)
2365                    Serial.println("Frenando despacio");
2366            }
2367
2368            if(debug_dsplz)
2369            {
2370                Serial.println("bajada acondicionada:");
2371                Serial.println(bajada);
2372
2373                Serial.println("------------------------");
2374            }
2375
2376
2377
2378            if(bajada > 100)
2379                bajada = 100;
2380
2381            //Velocidad en rango 0 - 255
2382            // AZIMUT
2383            if(subida <= bajada)
2384                PWM = subida*(255.0/100.0);
2385            else if (subida > bajada)
2386            {
2387                subida = bajada;
2388                // se iguala la velocidad de la rampa de
2389                // aceleración a la de deceleración para que en caso de
2390                // tener que acelerar se empiece desde la velocidad anterior
2391                PWM = bajada*(255.0/100.0);
2392            }
2393            if(PWM > vmax)
2394                PWM = vmax;
2395            if(consigna >= pos)
2396            {
2397                // CW
2398                digitalWrite(cw,LOW);
2399                digitalWrite(ccw,HIGH);
2400                if(freno)
2401                {
2402                    digitalWrite(brk,LOW); // NC
2403                }
2404
2405
2406            }else
2407            {
```

```
2408                // CCW
2409                digitalWrite(cw,HIGH);
2410                digitalWrite(ccw,LOW);
2411                if(freno)
2412                {
2413                    digitalWrite(brk,LOW); // NC
2414                }
2415            }
2416        }else
2417        {
2418            subida = 0;
2419            bajada = 0;
2420            PWM = 0;
2421            digitalWrite(cw,HIGH);
2422            digitalWrite(ccw,HIGH);
2423            digitalWrite(brk,HIGH); // NC
2424        }
2425
2426        analogWrite(PWMpin, PWM);
2427    }
2428
2429    /*-----------------------------------------------------------------------------
2430                                LECTURA DEL TECLADO
2431    -----------------------------------------------------------------------------
2432    *
2433    *                                función teclado
2434    *
2435    * Esta función realiza la lectura de la consigna escrita en el teclado
2436    *
2437    *
2438    *  entradas:
2439    *            lectura hardware de teclado
2440    *
2441    *  salidas:
2442    *            consigna --> num
2443    *
2444    *  llamadas a otras funciones:
2445    *
2446    *-----------------------------------------------------------------------------*/
2447
2448    float teclado()
2449    {
2450        char i;
2451        bool polling_teclado = 0;
2452        // Permanece en un bucle mientras polling sea 0, y sale del bucle
2453        // cuando se pulse tecla aceptar '#', poniendo polling = 1
2454        float num;
2455        char consigna[6] = "000000";//inicializo a 0 el vector
2456        char cont_key = 0;
2457        lcd.setCursor(0,1);
2458        lcd.print("CONSIGNA:");
2459        // Salgo del bucle si ya acepte la consigna o pulse intro
2460        tecla_intro = digitalRead(intro);
2461        while(!polling_teclado && tecla_intro)
2462        {
2463            char key = keypad.getKey();
2464            if (key)
2465            {
2466                delay(200); // Antirebotes
2467                // máximo = centenas + . + 2 decimales -> necesito vector de 6
2468                // Se rellena el vector de derecha a izquierda -> despalzar la
2469                // cifra anterior de derecha a izquierda con cada pulsación
2470
2471                if(((key >= '0' && key <= '9') || key =='.')&& cont_key < 6)
2472                // tecla valida y no excede el numero de pulsaciones
2473                {
2474                    cont_key++;
2475                    for(i=0;i<6;i++)
2476                    {
2477                        consigna[i] = consigna[i+1];
2478                        // Desplazo a la izquierda cada vez que pulso un numero
2479                    }
2480                    consigna[5] = key; // La última pulsación se guarda directamente
```

```
2481
2482                lcd.setCursor(0,2);
2483                num = atof(consigna);
2484                lcd.print("                    ");
2485                lcd.setCursor(0,2);
2486                lcd.print(num);
2487
2488            }else if(key == '#' && cont_key > 0)
2489            {
2490                num = atof(consigna);
2491                if(num <= 360)
2492                {
2493                    lcd.setCursor(0,1);
2494                    lcd.print("CONSIGNA ACEPTADA");
2495                    lcd.setCursor(0,2);
2496                    lcd.print("                    ");
2497                    lcd.setCursor(0,2);
2498                    lcd.print(num);
2499                    delay(1000);
2500                    cont_key = 0;
2501                    polling_teclado = 1;
2502                }else
2503                {
2504                    lcd.setCursor(0,1);
2505                    lcd.print("CONSIGNA INCORRECTA");
2506                    lcd.setCursor(0,2);
2507                    //lcd.print("                    ");
2508                    lcd.print(num);
2509                    delay(1000);
2510                    cont_key = 0;
2511                    polling_teclado = 0;
2512                    lcd.setCursor(0,1);
2513                    lcd.print("CONSIGNA:           ");
2514                    char i;
2515                    for(i = 0; i < 6; i++)
2516                    {
2517                        consigna[i] = '0';
2518                    }
2519                    lcd.setCursor(0,2);
2520                    num = atof(consigna);
2521                    //lcd.print("                    ");
2522                    lcd.print(num);
2523
2524                }
2525            }else if(cont_key > 0 && key == 'C')
2526            {
2527                char i;
2528                for(i = 0; i < 6; i++)
2529                {
2530                    consigna[i] = '0';
2531                }
2532                lcd.setCursor(0,2);
2533                lcd.print("                    ");
2534                lcd.setCursor(0,2);
2535                num = atof(consigna);
2536                lcd.print(num);
2537                cont_key = 0;
2538            }else if(cont_key > 0 && key == 'D')
2539            {
2540                cont_key--;
2541                for(i=5;i>0;i--)
2542                {
2543                    consigna[i] = consigna[i-1];
2544                    //Desplazo a la derecha para borrar
2545                }
2546                lcd.setCursor(0,2);
2547                lcd.print("                    ");
2548                num = atof(consigna);
2549                lcd.setCursor(0,2);
2550                lcd.print(num);
2551
2552
2553
```

```
            }
            else
            {
                //SI PULSO OTRA LETRA, SIMPLEMENTE NO HACE NADA
            }


        }
        tecla_intro = digitalRead(intro);
    }
    if(!tecla_intro)
    {
        delay(300);
        nivel_consigna = 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Modo manual");
        lcd.setCursor(0,1);
        lcd.print("Selecciona el angulo");
        lcd.setCursor(0,2);
        lcd.print(submenu_consigna[p]);
    }else
    {
        return num;
    }

}

/*-----------------------------------------------------------------------------
                    INICIALIZACIÓN DEL CANopen
-----------------------------------------------------------------------------
*
*                               función CANinit
*
* Esta función inicializa el bus CAN y los encoders
*
*  entradas:
*
*
*  salidas:
*
*
*  llamadas a otras funciones:
*            funciones de la biblioteca de CAN
*            CAN.begin
*            CAN_MSGAVAIL
*            CAN.checkReceive
*            CAN.getCanId
*            CAN.readMsgBuf
*
*
*----------------------------------------------------------------------------*/

void CANinit()
{
    unsigned char len = 0;
    unsigned char buf[8];
    unsigned long canId = 0x00;
    byte msg8[8], stmp[2];
    char i;


    while ((CAN_OK != CAN.begin(CAN_20KBPS)) && nivel_setup == 2)
    // init can bus : baudrate = 20k
    {
        if(debug_CAN)
        {
            Serial.println("CAN BUS Shield init fail");
            Serial.println(" Init CAN BUS Shield again");
        }


        lcd.clear();
```

```
2627            lcd.setCursor(0,0);
2628            lcd.print("SETUP: CONFIGURACION");
2629            lcd.setCursor(0,1);
2630            lcd.print("ERROR EN INICIO CAN");
2631
2632            delay(100);
2633
2634            tecla_intro = digitalRead(intro);
2635            if(!tecla_intro)
2636            {
2637                delay(200);
2638                nivel_setup = 0;
2639                lcd.clear();
2640                lcd.setCursor(0,0);
2641                lcd.print("SETUP: CONFIGURACION");
2642                lcd.setCursor(0,1);
2643                lcd.print(menu_setup[z]);
2644            }
2645        }
2646
2647        if(nivel_setup == 2)
2648        {
2649            if(debug_CAN)
2650                Serial.println("CAN BUS Shield init ok!");
2651
2652            lcd.clear();
2653            lcd.setCursor(0,0);
2654            lcd.print("SETUP: CONFIGURACION");
2655            lcd.setCursor(0,1);
2656            lcd.print("Seleccion:");
2657            lcd.setCursor(0,2);
2658            lcd.print("Encoder TBN 12b");
2659            lcd.setCursor(0,3);
2660            lcd.print("CANbus OK");
2661            delay(1500);
2662
2663            for(i = 0; i<8; i++)
2664            {
2665                msg8[i]=0x00; // Para enviar en el pdo un mensaje vacio
2666            }
2667
2668            //while(CAN_MSGAVAIL != CAN.checkReceive());
2669
2670
2671
2672    //                  Respuesta en el paso de ON-->OFF
2673    //-------------------------------------------------------------------------------
2674    //                  Diría que si quito todo esto funcionaría igual
2675    //-------------------------------------------------------------------------------
2676            if(CAN_MSGAVAIL == CAN.checkReceive())
2677            {
2678                if(debug_CAN)
2679                {
2680                    Serial.println
2681                    ("-----------------------Setup----------------------");
2682                    Serial.println("OFF -> ON");
2683                    Serial.println
2684                    ("--------------------------------------------------");
2685                }
2686
2687                while(CAN_MSGAVAIL == CAN.checkReceive())
2688                {
2689                    while((canId = CAN.getCanId())!= 0x701)
2690                        // espera respuesta 1°encoder
2691                    {
2692                        CAN.readMsgBuf(&len, buf);
2693                    }
2694                    if(debug_CAN)
2695                    {
2696                        Serial.print("COB-ID: ");
2697                        Serial.print("0x");
2698                        Serial.print(canId, HEX);
2699                        Serial.print("\t");
```

```
                    Serial.print("Databytes: ");
                    for(int i = 0; i<len; i++)
                    {
                        Serial.print(buf[i], HEX);
                        Serial.print("\t");
                    }
                    Serial.println();
                    Serial.flush();
                }
                while((canId = CAN.getCanId())!= 0x702)
                    // espera respuesta 2°encoder
                {
                    CAN.readMsgBuf(&len, buf);
                }

                if(debug_CAN)
                {
                    Serial.print("COB-ID: ");
                    Serial.print("0x");
                    Serial.print(canId, HEX);
                    Serial.print("\t");
                    Serial.print("Databytes: ");


                    for(i = 0; i<len; i++)
                    {
                        Serial.print(buf[i], HEX);
                        Serial.print("\t");
                    }
                    Serial.println();
                    Serial.flush();
                }
                delay(50);
            }
        }


//                          START ALL NODES
//--------------------------------------------------------------------------------
        stmp[0] = 0x01;
        stmp[1] = 0x00;

        CAN.sendMsgBuf(0x00, CAN_STDID, 2, stmp);
        delay(50);

//                          PDO1 del boot-up
//--------------------------------------------------------------------------------

        while(CAN_MSGAVAIL == CAN.checkReceive())
        {
            if(debug_CAN)
            {
                Serial.println
                ("-----------------------------------------------");
                Serial.println("Start all nodes");
                Serial.println
                ("-----------------------------------------------");
            }

            while((canId = CAN.getCanId())!= 0x181)// espera respuesta 1°encoder
            {
                CAN.readMsgBuf(&len, buf);
            }

            if(debug_CAN)
            {
                Serial.print("COB-ID: ");
                Serial.print("0x");
                Serial.print(canId, HEX);
                Serial.print("\t");
                Serial.print("Databytes: ");
                for(int i = 0; i<len; i++)
```

```
                {
                    Serial.print(buf[i], HEX);
                    Serial.print("\t");
                }
                Serial.println();
                Serial.flush();
            }

            while((canId = CAN.getCanId())!= 0x182)// espera respuesta 2°encoder
            {
                CAN.readMsgBuf(&len, buf);
            }

            if(debug_CAN)
            {
                Serial.print("COB-ID: ");
                Serial.print("0x");
                Serial.print(canId, HEX);
                Serial.print("\t");
                Serial.print("Databytes: ");

                for(i = 0; i<len; i++)
                {
                    Serial.print(buf[i], HEX);
                    Serial.print("\t");
                }
                Serial.println();
                Serial.flush();
            }
            delay(50);
        }
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("SETUP: CONFIGURACION");
        lcd.setCursor(0,1);
        lcd.print(menu_setup[z]);


        nivel_setup = 0; //Regreso al SETUP
    }

}
/*-------------------------------------------------------------------------------
                    LECTURA DE POSICIÓN POR ENCODERS
-------------------------------------------------------------------------------
*
*                               función CANpos
*
* Esta función realiza la lectura de los encoder, actualizando la posición de
* antenas  (az y el)
*
*  entradas:
*
*
*  salidas:
*            az --> azimut
*            el --> elevación
*
*  llamadas a otras funciones:
*            funciones de la biblioteca de CAN
*            CAN.sendMsgBuf
*            CAN.getCanId
*            CAN.readMsgBuf
*
*-------------------------------------------------------------------------------*/
void CANpos(float &az, float &el)
{
    byte RTR = 0x01;
    unsigned char len = 0;
    unsigned char buf[8];
    unsigned long canId;            //canId = 32 bits --> long
    int i;

```

```
2846        if(debug_CAN)
2847        {
2848            Serial.println
2849            ("-----------------------------Loop-------------------------");
2850            Serial.println
2851            ("----------------------------------------------------------");
2852            Serial.println
2853            ("Nombre Nodo\tCOB-ID\t\tPosicion");
2854            Serial.println
2855            ("----------------------------------------------------------");
2856        }
2857
2858    //-------------------------------------------------------------------------------
2859    // ENVÍO EL PDO VACÍO CON RTR AL NODO 1
2860    //-------------------------------------------------------------------------------
2861
2862        CAN.sendMsgBuf(0x181, CAN_STDID, RTR, 0, 0, true);
2863        // CAN.sendMsgBuf(ID, extension, RTR, DLC, nada,bool wait_sent=true)
2864        // (línea 136 de mcp_can.h)
2865        delay(50);
2866
2867    //-------------------------------------------------------------------------------
2868    // RESPUESTA DEL ENCODER 1
2869    //-------------------------------------------------------------------------------
2870        while((canId = CAN.getCanId())!= 0x181)// espera respuesta 1°encoder
2871        {
2872            CAN.readMsgBuf(&len, buf);
2873        }
2874        CAN.readMsgBuf(&len, buf);
2875
2876        az = (buf[0]+256*buf[1])*(360.0/4095.0);
2877
2878        if(debug_CAN)
2879        {
2880            Serial.print("AZIMUT\t\t");
2881            Serial.print("0x");
2882            Serial.print(canId,HEX);
2883            Serial.print("\t\t");
2884            Serial.println(az);
2885            Serial.flush();
2886        }
2887
2888        delay(10);
2889
2890    //-------------------------------------------------------------------------------
2891    // ENVÍO EL PDO VACÍO CON RTR AL NODO 2
2892    //-------------------------------------------------------------------------------
2893
2894        CAN.sendMsgBuf(0x182, CAN_STDID, RTR, 0, 0, true);
2895        // CAN.sendMsgBuf(ID, extension, RTR, DLC, nada,bool wait_sent=true)
2896        // (línea 136 de mcp_can.h)
2897        delay(50);
2898
2899    //-------------------------------------------------------------------------------
2900    // RESPUESTA DEL ENCODER 2
2901    //-------------------------------------------------------------------------------
2902        while((canId = CAN.getCanId())!= 0x182)// espera respuesta 2°encoder
2903        {
2904            CAN.readMsgBuf(&len, buf);
2905        }
2906        CAN.readMsgBuf(&len, buf);
2907
2908        el = (buf[0]+256*buf[1])*(360.0/4095.0);
2909
2910
2911        if(debug_CAN)
2912        {
2913            Serial.print("\tELEVACION\t");
2914            Serial.print("0x");
2915            Serial.print(canId, HEX);
2916            Serial.print("\t\t");
2917            Serial.println(el);
2918            Serial.flush();
```

```
    }


    delay(10);
}


/*-------------------------------------------------------------------------------
            MAQUINA DE ESTADOS: DECODIFICADOR DE TRAMA EASYCOMM
-------------------------------------------------------------------------------
*
*                               función deco_easycomm
*
* Esta función decodifica la trama easycom recibida del software externo por
* puerto serie
*
*
*  entradas:
*           caracter del puerto serie --> c
*
*  salidas:
*           s_easycom --> estados del decodificador easycom
*           dat_valido --> validez de la lectura
*           cel --> contador elevacion
*           caz --> contador azimut
*           azimut[6]
*           elevacion[6]
*
*  llamadas a otras funciones:
*
*-------------------------------------------------------------------------------*/

boolean deco_easycomm(char c)
{
    boolean analizado = false;

    // dat_valido se usa para que cuando la trama venga vacía (AZ EL) no se
    // decodifique

    switch (c) {
    case 'A':
    case 'a':
        if (s_easycom == 0) {
            s_easycom++;
            dat_valido=0;
            cel = caz = 0;
        }
        break;
//-------------------------------------------------------------------------------
    case 'Z':
    case 'z':
        if (s_easycom == 1)
            s_easycom++;
        else
            s_easycom = 0;
        break;
//-------------------------------------------------------------------------------
    case 'e':
    case 'E':
        if (s_easycom == 3)
            s_easycom++;
        else
            s_easycom = 0;
        break;
//-------------------------------------------------------------------------------
    case 'l':
    case 'L':
        if (s_easycom == 4){
            s_easycom++;
        }else
        {
            s_easycom = 0;
```

```
2992            }
2993            break;
2994    //-------------------------------------------------------------------------
2995        case ' ': // Espacio
2996            if (s_easycom == 2)
2997            {
2998                if(dat_valido == 1)
2999                {
3000                    s_easycom++;
3001                }else
3002                {
3003                    s_easycom = 0;
3004                }
3005            }else
3006                s_easycom = 0;
3007          break;
3008        case 0x0D:    //CR (retorno de carro)
3009    //-------------------------------------------------------------------------
3010            if(s_easycom == 5){
3011                if(dat_valido == 1)
3012                {
3013                    analizado = true;
3014                    dat_valido = 0;
3015                }
3016                s_easycom == 0;
3017            }
3018            break;
3019    //-------------------------------------------------------------------------
3020        case '0':
3021        case '1':
3022        case '2':
3023        case '3':
3024        case '4':
3025        case '5':
3026        case '6':
3027        case '7':
3028        case '8':
3029        case '9':
3030        case ',':
3031        case '-': // valor negativo
3032        case '.':
3033            if (s_easycom == 2)
3034            {
3035                dat_valido=1;
3036                if (caz > 5)
3037                {
3038                    caz = 0;
3039                }else
3040                {
3041                    azimut[caz++] = c;
3042                }
3043            } else if (s_easycom == 5)
3044            {
3045                if (cel > 5)
3046                {
3047                    cel = 0;
3048                }else
3049                {
3050                    elevacion[cel++] = c;
3051                }
3052            }
3053            break;
3054    //-------------------------------------------------------------------------
3055        default:
3056            s_easycom = 0;
3057            break;
3058        }
3059        return analizado;
3060    }
3061
3062    /*-------------------------------------------------------------------------
3063                CONVERTIDOR DE CADENA DE CARACTERES A FLOAT
3064    -------------------------------------------------------------------------
```

```
*
*                                función cadena2float
*
* Esta función devuelve el float correspondiente a una cadena de 6 caracteres
* cuyo separador decimal es ',' en lugar de '.'
*
*  entradas:
*            cadena[6]
*
*  salidas:
*            num --> variable float correspondiente a cadena[6]
*
*  llamadas a otras funciones:
*
*-------------------------------------------------------------------------*/
float cadena2float(char cadena[6])
{
    float num = 0;
    char i;
    char numeros[6];
    for(i = 0;i<6;i++)
    {
        numeros[i]=cadena[i];
        if(cadena[i] == ',')
        {
            numeros[i] = '.';
        }
    }
    num = atof(numeros);
    return num;
}

/*-------------------------------------------------------------------------
            MAQUINA DE ESTADOS PARA LECTURA DE TRAMA GPS NMEA
-------------------------------------------------------------------------
*
*                                función deco_nmea
*
* Esta función decodifica la trama NMEA en formato GPRMC de los datos serie del
* GPS
*
*  entradas:
*        caracter del puerto serie --> c
*
*  salidas:
*        s_GPS --> estado de la maquina de estados
*        ch --> contador hora
*        clat --> contador latitud
*        clong --> contador longitud
*        cfec --> contador fecha
*        hora[10]
*        fecha[7]
*        latitud[11]
*        longitud[12]
*        validez
*
*  llamadas a otras funciones:
*
*-------------------------------------------------------------------------*/
boolean deco_nmea(char c)
{
    boolean analizado = false;
    switch (c)
    {
//-------------------------------------------------------------------------
        case '$':
            if (s_GPS == 0)
            {
                s_GPS++;
                clat = clong = ch = cfec = 0;
            }
            break;
//-------------------------------------------------------------------------
```

```
3138            case 'g':
3139            case 'G':
3140                if (s_GPS == 1)
3141                    s_GPS++;
3142                else
3143                    s_GPS = 0;
3144                break;
3145    //------------------------------------------------------------------------------
3146            case 'p':
3147            case 'P':
3148                if (s_GPS == 2)
3149                    s_GPS++;
3150                else
3151                    s_GPS = 0;
3152                break;
3153    //------------------------------------------------------------------------------
3154            case 'r':
3155            case 'R':
3156                if (s_GPS == 3)
3157                    s_GPS++;
3158                else
3159                    s_GPS = 0;
3160                break;
3161    //------------------------------------------------------------------------------
3162            case 'm':
3163            case 'M':
3164                if (s_GPS == 4)
3165                    s_GPS++;
3166                else
3167                    s_GPS = 0;
3168                break;
3169    //------------------------------------------------------------------------------
3170            case 'c':
3171            case 'C':
3172                if (s_GPS == 5)
3173                    s_GPS++;
3174                else
3175                    s_GPS = 0;
3176                break;
3177    //------------------------------------------------------------------------------
3178            case 'a':
3179            case 'A':
3180            case 'v':
3181            case 'V':
3182                if (s_GPS == 8)
3183                {
3184                    validez = c;
3185                    s_GPS++;
3186                }
3187                else
3188                    s_GPS = 0;
3189                break;
3190    //------------------------------------------------------------------------------
3191            case ',':
3192                if( s_GPS == 6 || s_GPS == 7 || s_GPS == 9 || s_GPS == 11 ||
3193                    s_GPS == 13 || s_GPS == 15 || s_GPS == 17 || s_GPS == 18
3194                    || s_GPS == 19 )
3195                {
3196                    s_GPS++;
3197                }else if(s_GPS == 20)
3198                {
3199                    s_GPS == 0;
3200                    analizado = true;
3201                } else
3202                    s_GPS == 0;
3203                break;
3204    //------------------------------------------------------------------------------
3205            case '0':
3206            case '1':
3207            case '2':
3208            case '3':
3209            case '4':
3210            case '5':
```

```
3211            case '6':
3212            case '7':
3213            case '8':
3214            case '9':
3215                if (s_GPS == 7)
3216                {
3217                    hora[ch++] = c;
3218                    if (ch > 8)
3219                    {
3220                        hora[9] = '\0';
3221                        ch = 0;
3222                    }
3223                } else if (s_GPS == 11 || s_GPS == 10)
3224                {
3225                    latitud[clat++] = c;
3226                    if (clat > 9)
3227                    {
3228                        latitud[10] = '\0';
3229                        clat = 0;
3230                    }
3231                } else if (s_GPS == 15 || s_GPS == 14)
3232                {
3233                    longitud[clong++] = c;
3234                    if (clong > 10)
3235                    {
3236                        longitud[11] = '\0';
3237                        clong = 0;
3238                    }
3239                }else if (s_GPS == 20)
3240                {
3241                    fecha[cfec++] = c;
3242                    if (cfec > 5)
3243                    {
3244                        fecha[6] = '\0';
3245                        cfec = 0;
3246                    }
3247                }else if(s_GPS == 17 || s_GPS == 18 || s_GPS == 19)
3248                {
3249                //NO HAGO NADA
3250                }
3251                break;
3252    //-------------------------------------------------------------------------------
3253            case '.':
3254                if (s_GPS == 7 || s_GPS == 10 || s_GPS == 14 )
3255                {
3256                    if (s_GPS == 7)
3257                    {
3258                        hora[6] = '.';
3259                        ch++;
3260                    }
3261                    if (s_GPS == 10)
3262                    {
3263                        latitud[4] = '.';
3264                        clat++;
3265                        s_GPS++;
3266                    }
3267                    if (s_GPS == 14)
3268                    {
3269                        longitud[5] = '.';
3270                        clong++;
3271                        s_GPS++;
3272                    }
3273                }else if(s_GPS == 17 || s_GPS == 18 || s_GPS == 19)
3274                {
3275                    //NO HAGO NADA
3276                }
3277                else
3278                {
3279                    s_GPS = 0;
3280                }
3281
3282                break;
3283    //-------------------------------------------------------------------------------
```

```
3284            case 's':
3285            case 'S':
3286            case 'n':
3287            case 'N':
3288                if (s_GPS == 12)
3289                {
3290                    latitud[9] = c;
3291                    s_GPS++;
3292                } else
3293                    s_GPS = 0;
3294                break;
3295    //-------------------------------------------------------------------------------
3296            case 'e':
3297            case 'E':
3298            case 'w':
3299            case 'W':
3300                if (s_GPS == 16)
3301                {
3302                    longitud[10] = c;
3303                    s_GPS++;
3304                } else
3305                    s_GPS = 0;
3306                break;
3307        }
3308        return analizado;
3309    }
3310
3311
3312    /*-------------------------------------------------------------------------------
3313                                        LECTURA DEL GPS
3314    -------------------------------------------------------------------------------
3315    *
3316    *                                 función leeGPS
3317    *
3318    * Esta función actualiza el valor de la fecha, hora, latitud y longitud
3319    *
3320    *  entradas:
3321    *
3322    *  salidas:
3323    *             fecha
3324    *             hora
3325    *             latitud
3326    *             longitud
3327    *             GPS_actualizado (es el return)
3328    *
3329    *  llamadas a otras funciones:
3330    *             deco_nmea
3331    *
3332    *------------------------------------------------------------------------------*/
3333
3334
3335    bool leeGPS(int &anho, int &mes, int &dia, int &horas, int &minut, double &seg,
3336        int &lat_grad, int &long_grad, float &lat_min, float &long_min)
3337    {
3338        char i;
3339        int ndatos = 0;
3340        bool GPS_analizado = 0;
3341        //float hora_float, fecha_float, lat_float, long_float, lat_min, long_min;
3342        float hora_float, fecha_float, lat_float, long_float;
3343        //int anho, mes, dia, horas, min, lat_grad, long_grad, aux_min, mes_aux;
3344        int aux_min, mes_aux, seg_aux;
3345
3346        if (gps.available() && !deco_nmea(gps.read()))
3347        {
3348            delay(5);
3349
3350            if (deco_nmea(gps.read()))
3351            {
3352    /*-------------------------------------------------------------------------------
3353                                        HORA UTC
3354    -------------------------------------------------------------------------------*/
3355                hora_float = atof(hora);
3356                horas = hora_float/10000;
```

```
3357                aux_min = hora_float/100;
3358                minut = aux_min - horas*100;
3359                seg_aux = (hora_float/1) - aux_min*100;
3360                seg = seg_aux;
3361                //horas = horas + 2;
3362                if(debug_GPS)
3363                {
3364                    Serial.println("Posición NMEA decodificada:");
3365                    Serial.print("Hora: ");
3366                    Serial.print(horas);Serial.print(":");Serial.print(minut);
3367                    Serial.print(":");Serial.println(seg);
3368                    Serial.flush();
3369                }
3370    /*---------------------------------------------------------------------------
3371                                    LATITUD
3372    ---------------------------------------------------------------------------*/
3373                lat_float = atof(latitud);
3374                lat_grad = lat_float/100;
3375                lat_min = lat_float - lat_grad*100;
3376                if(latitud[9] == 'S' || latitud[9] == 's')
3377                {
3378                    lat_grad = -1*lat_grad;
3379                    lat_min = -1*lat_min;
3380                }
3381
3382                if(debug_GPS)
3383                {
3384                    Serial.print("Latitud: ");
3385                    Serial.print(lat_grad); Serial.print("°");
3386                    Serial.print(lat_min);
3387                    Serial.println("'");//Serial.println(latitud[9]);
3388                    Serial.flush();
3389                }
3390    /*---------------------------------------------------------------------------
3391                                    LONGITUD
3392    ---------------------------------------------------------------------------*/
3393                long_float = atof(longitud);
3394                long_grad = long_float/100;
3395                long_min = long_float - long_grad*100;
3396                if(longitud[10] == 'W' || longitud[10] == 'w')
3397                {
3398                    long_grad = -1*long_grad;
3399                    long_min = -1*long_min;
3400                }
3401                if(debug_GPS)
3402                {
3403                    Serial.print("Longitud: ");
3404                    Serial.print(long_grad); Serial.print("°");
3405                    Serial.print(long_min);
3406                    Serial.println("'");//Serial.println(longitud[10]);
3407                    Serial.flush();
3408
3409    /*---------------------------------------------------------------------------
3410                                    VALIDEZ
3411    ---------------------------------------------------------------------------*/
3412                    Serial.print("Lectura válida: ");
3413                    Serial.println(validez);
3414                    Serial.flush();
3415    /*---------------------------------------------------------------------------
3416                                    FECHA
3417    ---------------------------------------------------------------------------*/
3418                    Serial.print("Fecha: ");
3419                }
3420                fecha_float = atof(fecha);
3421                dia = fecha_float/10000;
3422                mes_aux = fecha_float/100;
3423                mes = mes_aux - dia*100;
3424                anho = (fecha_float/1) - mes_aux*100 + 2000;
3425
3426                if(debug_GPS)
3427                {
3428                    Serial.print(dia);Serial.print("/");Serial.print(mes);
3429                    Serial.print("/");Serial.println(anho);
```

```
3430                    Serial.println
3431                    ("-----------------------------------------------");
3432                    Serial.flush();
3433                }
3434    /*-------------------------------------------------------------------------------
3435                                    REINICIO DE VARIABLES
3436    -------------------------------------------------------------------------------*/
3437                for(i = 0; i<10; i++)
3438                {
3439                    hora[i] = '0';
3440                    latitud[i] = '0';
3441                }
3442                hora[6] = '.';
3443                latitud[9] = 'N';
3444                latitud[4] = '.';
3445
3446                for(i = 0; i<6; i++)
3447                {
3448                    fecha[i] = '0';
3449                }
3450
3451                for(i = 0; i<11; i++)
3452                {
3453                    longitud[i] = '0';
3454                }
3455                longitud[5] = '.';
3456                longitud[10] = 'W';
3457
3458                GPS_analizado = 1;
3459            }
3460        }
3461
3462        return GPS_analizado;
3463    }
3464
3465    /*-------------------------------------------------------------------------------
3466                            CÁLCULOS AZ/EL
3467    -------------------------------------------------------------------------------
3468    *
3469    *                               función calc_kepler
3470    *
3471    * Esta función calcula los ángulos de apuntamiento de la antena (AZ,EL) en
3472    * función del instante y del lugar de observación
3473    *
3474    *  entradas:
3475    *            año --> ano
3476    *            mes --> mes
3477    *            día --> dia
3478    *            horas --> horas
3479    *            minuto --> minu
3480    *            segundo --> seg
3481    *            grados latitud --> lat_grados
3482    *            minutos latitud -->lat_minu
3483    *            grados latitud --> lon_grados
3484    *            minutos latitud --> lon_minu
3485    *
3486    *  salidas:
3487    *            az --> azimut
3488    *            el --> elevación
3489    *
3490    *  llamadas a otras funciones:
3491    *            deco_nmea
3492    *
3493    *-------------------------------------------------------------------------------*/
3494
3495    void calc_kepler
3496    (int ano, int mes, int dia, int horas, int minu, double seg,
3497    int lat_grados, int lon_grados, float lat_minu, float lon_minu,
3498    float &az, float &el)
3499    {
3500                if(debug_GPS)
3501                {
3502                    Serial.print("AÑO: ");Serial.println(ano);
```

```
                Serial.print("MES: ");Serial.println(mes);
                Serial.print("DÍA: ");Serial.println(dia);
                Serial.print("HORA: ");Serial.println(horas);
                Serial.print("MINUTOS: ");Serial.println(minu);
                Serial.print("SEG: ");Serial.println(seg);
                Serial.print("LAT_grad: ");Serial.println(lat_grados);
                Serial.print("lat_minut: ");Serial.println(lat_minu);
                Serial.print("long_grad: ");Serial.println(lon_grados);
                Serial.print("long_min: ");Serial.println(lon_minu);

            }


    //------------------------  variables locales  -------------------------
    gravconsttype wgs = constDef;           // (gravconsttype es una
                                            // enumeración definida en
                                            // sgp4unit.h) del estándar
                                            // world geodetic system
    char    modoOperacion = opDef;
    //char linea1[70], linea2[70];

    elsetrec datosSat;

    long int diaj;
    double  mpe,
            fraccion_dj,
            diajReduc,
            epocaTotal,
            lat,
            //alt,
            lon,
            thetaLoc;
    double rSat[3], vSat[3], rLoc[3], dif[3], rSez[3];

    int i, j;



    // ----------------- datos lugar e instante observación --------------------
    if(debug_kepler_loop)
    {
        ano = 2019; mes = 8; dia = 26;
        horas = 11; minu = 8; seg = 51.0;
        lat_grados = 43; lat_minu = 35.82;
        lon_grados = -8; lon_minu = -11.79;
    }
/* Observacion de referencia verificada
    ano = 2019; mes = 8; dia = 11;
    horas = 15; minu = 50; seg = 0.0;
    lat_grados = 43; lat_minu = 35.82;
    lon_grados = -8; lon_minu = -11.79;
*/



    //----------------------------  código  ---------------------------------
    if(debug_kepler)
    {
        Serial.print(
"*******************************************************************************\n");
        Serial.print(
        "\t\t\t\tDATOS TLE:\nLinea 1:\n");

        for(i = 0; i < 69; i++)
        {
            linea1[i] = EEPROM.read(i);
            if(debug_kepler)
            {
                Serial.print(linea1[i]);
            }
        }
        linea1[i++] = '\0';

```

```
3576            if(debug_kepler)
3577            {
3578                Serial.print(
3579                "\nLinea 2:\n");
3580            }
3581            for(j = 0; i < 139; i++, j++)
3582            {
3583                linea2[j] = EEPROM.read(i);
3584                if(debug_kepler)
3585                {
3586                    Serial.print(linea2[j]);
3587                }
3588            }
3589            linea2[j] = '\0';
3590            if(debug_kepler)
3591            {
3592                Serial.println();
3593                Serial.print(
3594        "*********************************************************************\n");
3595            }
3596        }
3597    /*
3598            Serial.print(
3599    "*********************************************************************\n");
3600            Serial.print(
3601            "\t\t\t\tDATOS TLE:\nLínea 1:\n");
3602        }
3603
3604
3605        for(i = 0; i < 69; i++)
3606        {
3607            if(debug_kepler)
3608                Serial.print(*(tle + i + 1));       // hai que sumar 1 por
3609                                                    // culpa do intro do principio
3610            linea1[i] = *(tle  + i + 1);
3611        }
3612        linea1[i++] = '\0';
3613
3614
3615        if(debug_kepler)
3616        {
3617            Serial.print(
3618            "\nLinea 2:\n");
3619        }
3620
3621
3622        for(j = 0; i < 139; i++, j++)
3623        {
3624            if(debug_kepler)
3625                Serial.print(*(tle + i + 1));
3626            linea2[j] = *(t
3627            + i + 1);
3628        }
3629        linea2[j] = '\0';
3630
3631        if(debug_kepler)
3632        {
3633            Serial.println();
3634            Serial.print(
3635    "*********************************************************************\n");
3636        }
3637    */
3638
3639
3640        // ------------------- cálculo vectores posición --------------------------
3641        //--------- satélite -------------------------------------------------------
3642        tleArv(linea1, linea2, modoOperacion, wgs, datosSat);
3643                                                    // decodificamos datos TLE y
3644                                                    // guardamos información en
3645                                                    // datosSat
3646        diajul(ano, mes, dia, horas, minu, seg, diaj, fraccion_dj);
3647                                                    // devuelve día juliano en diaj
3648                                                    // y la fraccion del día en
```

```
                                                    // fraccion_dj (en días)
    diajReduc = (diaj - datosSat.jdsatepoch) + fraccion_dj;
    //epocaReduc = (datosSat.jdsatepoch - 2458484)+datosSat.jdsatepoch_fraccion;
        // para ganar precisión en el instante de observación, restamos las
        // partes enteras del dia de observación y la época TLE, y queda una
        // parte entera de 1 o 2 dígitos, que permite mejorar el número de
        // decimales
    mpe = (diajReduc - datosSat.jdsatepoch_fraccion) * 1440.0;
                                                    // minutos desde época TLE
    sgp4 (wgs, datosSat,  mpe, rSat,  vSat);    // devuelve posición y velocidad
                                                    // del satélite en rSat y vSat
    //--------- Lugar de observación ---------------------------------------------
    lat = gradArad(lat_grados, lat_minu);
    lon = gradArad(lon_grados, lon_minu);       // pasamos a radianes
    thetaLoc = theta(lon, diaj, fraccion_dj);   // devuelve ángulo respecto a
                                                    // punto aries
    geoide(lat, thetaLoc,/* alt,*/ rLoc);
                                                    // devuelve posición y velocidad
                                                    // locales corregidas
    for(i = 0; i < 3; i++)
        dif[i] = rSat[i] - rLoc[i];

    // ------------------- cambio a coordenadas SEZ ----------------------------

    rSez[0] =       sin(lat) * cos(thetaLoc) * dif[0] +
                    sin(lat) * sin(thetaLoc) * dif[1] -
                    cos(lat) * dif[2];
    rSez[1] =      -sin(thetaLoc) * dif[0] +
                    cos(thetaLoc) * dif[1];
    rSez[2] =       cos(lat) * cos(thetaLoc) * dif[0] +
                    cos(lat) * sin(thetaLoc) * dif[1] +
                    sin(lat) * dif[2];

    // -------------- cálculo ángulos azimut y elevación -----------------------

    az = atan(-rSez[1] / rSez[0]);
    if (rSez[0] > 0.0)
    {
        az += pi;
    }

    if (az < 0.0)
    {
        az += 2.0 * pi;
    }
    az = 180 * az / pi;
    el = asin(rSez[2] /
        sqrt(pow(rSez[0], 2.0) + pow(rSez[1], 2.0) + pow(rSez[2], 2.0)));
    el = 180 * el / pi;

    // ----------------- presentación resultados depuración---------------------
    if(debug_kepler)
    {
        Serial.print(
"*********************************************************************\n");
    Serial.print(
    "\t\t\t\tRESULTADOS\n");
    Serial.print(
"*********************************************************************\n");
    Serial.print(
    "Instante de la observacion:\t");
    Serial.print(ano); Serial.print("\t");
    Serial.print(mes); Serial.print("\t");
    Serial.print(dia); Serial.print("\t");
    Serial.print(hora); Serial.print("\t");
    Serial.print(minu); Serial.print("\t");
    Serial.print(seg); Serial.print("\n");

    Serial.print(
    "Dia de la observacion en juliana:\t");
    Serial.print(diaj); Serial.print("\n");

    Serial.print(
```

```
3722        "Fraccion del dia en segundos:\t\t");
3723        Serial.print(fraccion_dj,8); Serial.print("\n");
3724
3725        /*
3726        Serial.print(
3727        "Instante de la observacion en juliana:\t");
3728        //Serial.print(diajFloat,8);Serial.print("\t");
3729        //Serial.print(fraccion_dj / segxDia_,8); Serial.print("\t");
3730        //Serial.print(diajLong,8); Serial.print("\n");
3731        */
3732        Serial.print(
3733        "época TLE en juliana:\t\t\t");
3734        Serial.print(datosSat.jdsatepoch);  Serial.print("\n");
3735
3736        Serial.print(
3737        "año TLE: \t\t\t\t");
3738        Serial.print(datosSat.epochyr);     Serial.print("\n");
3739
3740        Serial.print(
3741        "dias TLE (n° de dias del año TLE):\t");
3742        Serial.print(datosSat.epochdays);   Serial.print("\n");
3743
3744        Serial.print(
3745        "minutos desde la epoca:\t\t\t");
3746        Serial.print(mpe,8); Serial.print("\n");
3747
3748        Serial.print(
3749        "Latitud local (radianes):\t\t");
3750        Serial.print(lat);
3751
3752        Serial.print(
3753        "\nLongitud local (radianes):\t\t");
3754        Serial.print(lon);
3755        Serial.print("\n");
3756
3757        Serial.print(
3758        "Hora sideral Greenwich (GMST):\t\t");
3759        Serial.println(thetaLoc);
3760
3761        Serial.print(
3762    "*******************************************************************************\n");
3763        Serial.print(
3764        "Vector velocidad:\t\t\t");
3765        Serial.print(vSat[0]); Serial.print("\t");
3766        Serial.print(vSat[1]); Serial.print("\t");
3767        Serial.print(vSat[2]);
3768        Serial.println();
3769
3770        Serial.print(
3771        "Vector de posicion satelite:\t\t");
3772        Serial.print(rSat[0]); Serial.print("\t");
3773        Serial.print(rSat[1]); Serial.print("\t");
3774        Serial.print(rSat[2]);
3775        Serial.println();
3776
3777        Serial.print(
3778        "Vector de posicion local (ECI):\t\t");
3779        Serial.print(rLoc[0]); Serial.print("\t");
3780        Serial.print(rLoc[1]); Serial.print("\t");
3781        Serial.print(rLoc[2]);
3782        Serial.println();
3783
3784        Serial.print(
3785        "Vector de posicion local (SEZ):\t\t");
3786        Serial.print(rSez[0]); Serial.print("\t");
3787        Serial.print(rSez[1]); Serial.print("\t");
3788        Serial.print(rSez[2]);
3789        Serial.println();
3790
3791        Serial.print(
3792        "Angulo de azimut:\t\t");
3793        Serial.print((float)az);
3794        Serial.println();
```

```
3795
3796        Serial.print(
3797        "Angulo de elevación:\t\t");
3798        Serial.print((float)el);
3799        Serial.println();
3800        Serial.print(
3801    "*********************************************************************\n");
3802
3803            //while(1);
3804
3805        }
3806    }
3807
3808
3809    /*-------------------------------------------------------------------------
3810                          ACTUALIZACIÓN DE DATOS TLE
3811    -------------------------------------------------------------------------
3812    *
3813    *                               función actualizaTLE
3814    *
3815    * Permite actualizar los datos TLE, y guardarlos en la memoria y en las variable
3816    * linea1 y linea2, que se utilizará para los cálculos keplerianos
3817    *
3818    *  entradas:
3819    *            lectura puerto serie de datos TLE
3820    *
3821    *  salidas:
3822    *            linea1[70] --> línea 1 del TLE
3823    *            linea2[70] --> línea 2 del TLE
3824    *            nivel_setup--> permite regresar al menú setup
3825    *
3826    *  llamadas a otras funciones:
3827    *
3828    *
3829    *-------------------------------------------------------------------------*/
3830
3831    void actualizaTLE()
3832    {
3833        int i, j, eeDir;
3834        //char linea1[70], linea2[70];
3835        String buffer_S;
3836
3837        Serial.println("Copia los datos TLE de la siguiente dirección: ");
3838        Serial.println("https://www.celestrak.com/NORAD/elements/");
3839        Serial.println
3840        ("Copia únicamente las líneas 1 y 2, sin el nombre del satélite");
3841        Serial.println
3842        ("Pega en el puerto serie los datos TLE del satélite y envía");
3843        Serial.end();
3844        Serial.begin(9600);
3845        // Se apaga y se enciende el puerto serie para reiniciar porque flush() no
3846        // funciona correctamete
3847        while(!Serial.available());
3848        buffer_S = Serial.readString();
3849        Serial.println("Grabando en EEPROM...");
3850
3851    // ------------------ grabación en EEPROM
3852        eeDir = 0;
3853        while(buffer_S[eeDir] != '\0')
3854        {
3855            EEPROM.update(eeDir, buffer_S[eeDir]);
3856            //Serial.print(eeDir);
3857            //Serial.print('\t');
3858            //Serial.println(buffer_S[eeDir]);
3859            eeDir++;
3860        }
3861        EEPROM.update(eeDir, '\0');
3862        delay(1000);
3863        Serial.print(
3864    "*********************************************************************\n");
3865        Serial.print(
3866        "\t\t\t VERIFICACIÓN DE LOS DATOS GRABADOS EN EEPROM:\nLínea 1:\n");
3867        for(eeDir = 0; eeDir < 69; eeDir++)
```

```
3868        {
3869            linea1[eeDir] = EEPROM.read(eeDir);
3870            Serial.print(linea1[eeDir]);
3871        }
3872        linea1[eeDir++] = '\0';
3873
3874        Serial.print(
3875        "\nLinea 2:\n");
3876        for(j = 0; eeDir < 139; eeDir++, j++)
3877        {
3878             linea2[eeDir] = EEPROM.read(eeDir);
3879            Serial.print(linea2[eeDir]);
3880        }
3881        linea2[j] = '\0';
3882        Serial.println();
3883        Serial.print(
3884    "**************************************************************************\n");
3885        nivel_setup = 0;
3886        lcd.clear();
3887        lcd.setCursor(0,0);
3888        lcd.print("SETUP: CONFIGURACION");
3889        lcd.setCursor(0,1);
3890        lcd.print("TLE actualizado");
3891        delay(1500);
3892        lcd.clear();
3893        lcd.setCursor(0,0);
3894        lcd.print("SETUP: CONFIGURACION");
3895        lcd.setCursor(0,1);
3896        lcd.print(menu_setup[z]);
3897
3898    }
```

18.5. SGP4.cpp

```
1
2   #include "sgp4Completo.h"
3   #include <stdlib.h>
4
5
6   const char help = 'n';
7   FILE *dbgfile;
8
9
10  /* ----------- local functions - only ever used internally by sgp4 ---------- */
11  static void dpper
12        (
13          double e3,     double ee2,    double peo,     double pgho,   double pho,
14          double pinco,  double plo,    double se2,     double se3,    double sgh2,
15          double sgh3,   double sgh4,   double sh2,     double sh3,    double si2,
16          double si3,    double sl2,    double sl3,     double sl4,    double t,
17          double xgh2,   double xgh3,   double xgh4,    double xh2,    double xh3,
18          double xi2,    double xi3,    double xl2,     double xl3,    double xl4,
19          double zmol,   double zmos,   double inclo,
20          char init,
21          double& ep,    double& inclp, double& nodep,  double& argpp, double& mp,
22          char opsmode
23        );
24
25  static void dscom
26        (
27          double epoch,  double ep,     double argpp,   double tc,     double inclp,
28          double nodep,  double np,
29          double& snodm, double& cnodm, double& sinim,  double& cosim, double& sinomm,
30          double& cosomm,double& day,   double& e3,     double& ee2,   double& em,
31          double& emsq,  double& gam,   double& peo,    double& pgho,  double& pho,
32          double& pinco, double& plo,   double& rtemsq, double& se2,   double& se3,
33          double& sgh2,  double& sgh3,  double& sgh4,   double& sh2,   double& sh3,
34          double& si2,   double& si3,   double& sl2,    double& sl3,   double& sl4,
35          double& s1,    double& s2,    double& s3,     double& s4,    double& s5,
36          double& s6,    double& s7,    double& ss1,    double& ss2,   double& ss3,
37          double& ss4,   double& ss5,   double& ss6,    double& ss7,   double& sz1,
38          double& sz2,   double& sz3,   double& sz11,   double& sz12,  double& sz13,
39          double& sz21,  double& sz22,  double& sz23,   double& sz31,  double& sz32,
40          double& sz33,  double& xgh2,  double& xgh3,   double& xgh4,  double& xh2,
41          double& xh3,   double& xi2,   double& xi3,    double& xl2,   double& xl3,
42          double& xl4,   double& nm,    double& z1,     double& z2,    double& z3,
43          double& z11,   double& z12,   double& z13,    double& z21,   double& z22,
44          double& z23,   double& z31,   double& z32,    double& z33,   double& zmol,
45          double& zmos
46        );
47
48  static void dsinit
49        (
50          gravconsttype whichconst,
51          double cosim,  double emsq,   double argpo,   double s1,     double s2,
52          double s3,     double s4,     double s5,     double sinim,  double ss1,
53          double ss2,    double ss3,    double ss4,    double ss5,    double sz1,
54          double sz3,    double sz11,   double sz13,   double sz21,   double sz23,
55          double sz31,   double sz33,   double t,      double tc,     double gsto,
56          double mo,     double mdot,   double no,     double nodeo,  double nodedot,
57          double xpidot, double z1,     double z3,     double z11,    double z13,
58          double z21,    double z23,    double z31,    double z33,    double ecco,
59          double eccsq,  double& em,    double& argpm, double& inclm, double& mm,
60          double& nm,    double& nodem,
61          int& irez,
62          double& atime, double& d2201, double& d2211, double& d3210, double& d3222,
63          double& d4410, double& d4422, double& d5220, double& d5232, double& d5421,
64          double& d5433, double& dedt,  double& didt,  double& dmdt,  double& dndt,
65          double& dnodt, double& domdt, double& del1,  double& del2,  double& del3,
66          double& xfact, double& xlamo, double& xli,   double& xni
67        );
68
69  static void dspace
70        (
71          int irez,
72          double d2201,  double d2211,  double d3210,  double d3222,  double d4410,
73          double d4422,  double d5220,  double d5232,  double d5421,  double d5433,
```

```
74          double dedt,   double del1,   double del2,    double del3,   double didt,
75          double dmdt,   double dnodt,  double domdt,   double argpo,  double argpdot,
76          double t,      double tc,     double gsto,    double xfact,  double xlamo,
77          double no,
78          double& atime, double& em,    double& argpm,  double& inclm, double& xli,
79          double& mm,    double& xni,   double& nodem,  double& dndt,  double& nm
80        );
81
82    static void initl
83        (
84          int satn,      gravconsttype whichconst,
85          double ecco,   double epoch,  double inclo,   double& no,
86          char& method,
87          double& ainv,  double& ao,    double& con41,  double& con42, double& cosio,
88          double& cosio2,double& eccsq, double& omeosq, double& posq,
89          double& rp,    double& rteosq,double& sinio , double& gsto, char opsmode
90        );
91
92    /* -----------------------------------------------------------------------
93    *
94    *                           procedure dpper
95    *
96    *  this procedure provides deep space long period periodic contributions
97    *    to the mean elements.  by design, these periodics are zero at epoch.
98    *    this used to be dscom which included initialization, but it's really a
99    *    recurring function.
100   *
101   *  author        : david vallado                  719-573-2600   28 jun 2005
102   *
103   *  inputs        :
104   *    e3          -
105   *    ee2         -
106   *    peo         -
107   *    pgho        -
108   *    pho         -
109   *    pinco       -
110   *    plo         -
111   *    se2 , se3 , sgh2, sgh3, sgh4, sh2, sh3, si2, si3, sl2, sl3, sl4 -
112   *    t           -
113   *    xh2, xh3, xi2, xi3, xl2, xl3, xl4 -
114   *    zmol        -
115   *    zmos        -
116   *    ep          - eccentricity                           0.0 - 1.0
117   *    inclo       - inclination - needed for lyddane modification
118   *    nodep       - right ascension of ascending node
119   *    argpp       - argument of perigee
120   *    mp          - mean anomaly
121   *
122   *  outputs       :
123   *    ep          - eccentricity                           0.0 - 1.0
124   *    inclp       - inclination
125   *    nodep        - right ascension of ascending node
126   *    argpp       - argument of perigee
127   *    mp          - mean anomaly
128   *
129   *  locals        :
130   *    alfdp       -
131   *    betdp       -
132   *    cosip  , sinip  , cosop  , sinop  ,
133   *    dalf        -
134   *    dbet        -
135   *    dls         -
136   *    f2, f3      -
137   *    pe          -
138   *    pgh         -
139   *    ph          -
140   *    pinc        -
141   *    pl          -
142   *    sel   , ses   , sghl  , sghs  , shl   , shs   , sil   , sinzf , sis   ,
143   *    sll   , sls
144   *    xls         -
145   *    xnoh        -
146   *    zf          -
```

```
147   *    zm          -
148   *
149   *  coupling      :
150   *    none.
151   *
152   *  references    :
153   *    hoots, roehrich, norad spacetrack report #3 1980
154   *    hoots, norad spacetrack report #6 1986
155   *    hoots, schumacher and glover 2004
156   *    vallado, crawford, hujsak, kelso  2006
157     ----------------------------------------------------------------------*/
158
159   static void dpper
160        (
161          double e3,     double ee2,    double peo,     double pgho,   double pho,
162          double pinco,  double plo,    double se2,     double se3,    double sgh2,
163          double sgh3,   double sgh4,   double sh2,     double sh3,    double si2,
164          double si3,    double sl2,    double sl3,     double sl4,    double t,
165          double xgh2,   double xgh3,   double xgh4,    double xh2,    double xh3,
166          double xi2,    double xi3,    double xl2,     double xl3,    double xl4,
167          double zmol,   double zmos,   double inclo,
168          char init,
169          double& ep,    double& inclp, double& nodep,  double& argpp, double& mp,
170          char opsmode
171        )
172   {
173        /* --------------------- local variables ------------------------ */
174        const double twopi = 2.0 * pi;
175        double alfdp, betdp, cosip, cosop, dalf, dbet, dls,
176             f2,    f3,    pe,    pgh,   ph,   pinc, pl ,
177             sel,   ses,   sghl,  sghs,  shll, shs,  sil,
178             sinip, sinop, sinzf, sis,   sll,  sls,  xls,
179             xnoh,  zf,    zm,    zel,   zes,  znl,  zns;
180
181        /* --------------------- constants ---------------------------- */
182        zns   = 1.19459e-5;
183        zes   = 0.01675;
184        znl   = 1.5835218e-4;
185        zel   = 0.05490;
186
187        /* --------------- calculate time varying periodics ----------- */
188        zm    = zmos + zns * t;
189        // be sure that the initial call has time set to zero
190        if (init == 'y')
191            zm = zmos;
192        zf    = zm + 2.0 * zes * sin(zm);
193        sinzf = sin(zf);
194        f2    =  0.5 * sinzf * sinzf - 0.25;
195        f3    = -0.5 * sinzf * cos(zf);
196        ses   = se2* f2 + se3 * f3;
197        sis   = si2 * f2 + si3 * f3;
198        sls   = sl2 * f2 + sl3 * f3 + sl4 * sinzf;
199        sghs  = sgh2 * f2 + sgh3 * f3 + sgh4 * sinzf;
200        shs   = sh2 * f2 + sh3 * f3;
201        zm    = zmol + znl * t;
202        if (init == 'y')
203            zm = zmol;
204        zf    = zm + 2.0 * zel * sin(zm);
205        sinzf = sin(zf);
206        f2    =  0.5 * sinzf * sinzf - 0.25;
207        f3    = -0.5 * sinzf * cos(zf);
208        sel   = ee2 * f2 + e3 * f3;
209        sil   = xi2 * f2 + xi3 * f3;
210        sll   = xl2 * f2 + xl3 * f3 + xl4 * sinzf;
211        sghl  = xgh2 * f2 + xgh3 * f3 + xgh4 * sinzf;
212        shll  = xh2 * f2 + xh3 * f3;
213        pe    = ses + sel;
214        pinc  = sis + sil;
215        pl    = sls + sll;
216        pgh   = sghs + sghl;
217        ph    = shs + shll;
218
219        if (init == 'n')
```

```
        {
        pe    = pe - peo;
        pinc  = pinc - pinco;
        pl    = pl - plo;
        pgh   = pgh - pgho;
        ph    = ph - pho;
        inclp = inclp + pinc;
        ep    = ep + pe;
        sinip = sin(inclp);
        cosip = cos(inclp);

        /* ----------------- apply periodics directly ------------ */
        //  sgp4fix for lyddane choice
        //  strn3 used original inclination - this is technically feasible
        //  gsfc used perturbed inclination - also technically feasible
        //  probably best to readjust the 0.2 limit value and limit discontinuity
        //  0.2 rad = 11.45916 deg
        //  use next line for original strn3 approach and original inclination
        //  if (inclo >= 0.2)
        //  use next line for gsfc version and perturbed inclination
        if (inclp >= 0.2)
          {
            ph     = ph / sinip;
            pgh    = pgh - cosip * ph;
            argpp  = argpp + pgh;
            nodep  = nodep + ph;
            mp     = mp + pl;
          }
          else
          {
            /* ---- apply periodics with lyddane modification ---- */
            sinop  = sin(nodep);
            cosop  = cos(nodep);
            alfdp  = sinip * sinop;
            betdp  = sinip * cosop;
            dalf   =  ph * cosop + pinc * cosip * sinop;
            dbet   = -ph * sinop + pinc * cosip * cosop;
            alfdp  = alfdp + dalf;
            betdp  = betdp + dbet;
            nodep  = fmod(nodep, twopi);
            //  sgp4fix for afspc written intrinsic functions
            // nodep used without a trigonometric function ahead
            if ((nodep < 0.0) && (opsmode == 'a'))
                nodep = nodep + twopi;
            xls    = mp + argpp + cosip * nodep;
            dls    = pl + pgh - pinc * nodep * sinip;
            xls    = xls + dls;
            xnoh   = nodep;
            nodep  = atan2(alfdp, betdp);
            //  sgp4fix for afspc written intrinsic functions
            // nodep used without a trigonometric function ahead
            if ((nodep < 0.0) && (opsmode == 'a'))
                nodep = nodep + twopi;
            if (fabs(xnoh - nodep) > pi)
              if (nodep < xnoh)
                 nodep = nodep + twopi;
              else
                 nodep = nodep - twopi;
            mp    = mp + pl;
            argpp = xls - mp - cosip * nodep;
          }
        }   // if init == 'n'

  //#include "debug1.cpp"
  }  // end dpper

  /*-----------------------------------------------------------------------------
  *
  *                           procedure dscom
  *
  *  this procedure provides deep space common items used by both the secular
  *    and periodics subroutines.  input is provided as shown. this routine
  *    used to be called dpper, but the functions inside weren't well organized.
```

```
293   *
294   *  author        : david vallado                  719-573-2600   28 jun 2005
295   *
296   *  inputs        :
297   *    epoch       -
298   *    ep          - eccentricity
299   *    argpp       - argument of perigee
300   *    tc          -
301   *    inclp       - inclination
302   *    nodep       - right ascension of ascending node
303   *    np          - mean motion
304   *
305   *  outputs       :
306   *    sinim  , cosim  , sinomm , cosomm , snodm  , cnodm
307   *    day         -
308   *    e3          -
309   *    ee2         -
310   *    em          - eccentricity
311   *    emsq        - eccentricity squared
312   *    gam         -
313   *    peo         -
314   *    pgho        -
315   *    pho         -
316   *    pinco       -
317   *    plo         -
318   *    rtemsq      -
319   *    se2, se3         -
320   *    sgh2, sgh3, sgh4        -
321   *    sh2, sh3, si2, si3, sl2, sl3, sl4         -
322   *    s1, s2, s3, s4, s5, s6, s7          -
323   *    ss1, ss2, ss3, ss4, ss5, ss6, ss7, sz1, sz2, sz3         -
324   *    sz11, sz12, sz13, sz21, sz22, sz23, sz31, sz32, sz33        -
325   *    xgh2, xgh3, xgh4, xh2, xh3, xi2, xi3, xl2, xl3, xl4         -
326   *    nm          - mean motion
327   *    z1, z2, z3, z11, z12, z13, z21, z22, z23, z31, z32, z33         -
328   *    zmol        -
329   *    zmos        -
330   *
331   *  locals        :
332   *    a1, a2, a3, a4, a5, a6, a7, a8, a9, a10         -
333   *    betasq      -
334   *    cc          -
335   *    ctem, stem        -
336   *    x1, x2, x3, x4, x5, x6, x7, x8          -
337   *    xnodce      -
338   *    xnoi        -
339   *    zcosg  , zsing  , zcosgl , zsingl , zcosh  , zsinh  , zcoshl , zsinhl ,
340   *    zcosi  , zsini  , zcosil , zsinil ,
341   *    zx          -
342   *    zy          -
343   *
344   *  coupling      :
345   *    none.
346   *
347   *  references    :
348   *    hoots, roehrich, norad spacetrack report #3 1980
349   *    hoots, norad spacetrack report #6 1986
350   *    hoots, schumacher and glover 2004
351   *    vallado, crawford, hujsak, kelso  2006
352     ----------------------------------------------------------------------*/
353
354   static void dscom
355        (
356          double epoch,
357          double ep,     double argpp,   double tc,     double inclp,
358          double nodep,  double np,
359          double& snodm, double& cnodm, double& sinim,  double& cosim, double& sinomm,
360          double& cosomm,double& day,   double& e3,     double& ee2,   double& em,
361          double& emsq,  double& gam,   double& peo,    double& pgho,  double& pho,
362          double& pinco, double& plo,   double& rtemsq, double& se2,   double& se3,
363          double& sgh2,  double& sgh3,  double& sgh4,   double& sh2,   double& sh3,
364          double& si2,   double& si3,   double& sl2,    double& sl3,   double& sl4,
365          double& s1,    double& s2,    double& s3,     double& s4,    double& s5,
```

```
366           double& s6,    double& s7,    double& ss1,    double& ss2,   double& ss3,
367           double& ss4,   double& ss5,   double& ss6,    double& ss7,   double& sz1,
368           double& sz2,   double& sz3,   double& sz11,   double& sz12,  double& sz13,
369           double& sz21,  double& sz22,  double& sz23,   double& sz31,  double& sz32,
370           double& sz33,  double& xgh2,  double& xgh3,   double& xgh4,  double& xh2,
371           double& xh3,   double& xi2,   double& xi3,    double& xl2,   double& xl3,
372           double& xl4,   double& nm,    double& z1,     double& z2,    double& z3,
373           double& z11,   double& z12,   double& z13,    double& z21,   double& z22,
374           double& z23,   double& z31,   double& z32,    double& z33,   double& zmol,
375           double& zmos
376         )
377   {
378         /* ------------------------ constants ------------------------- */
379         const double zes     =  0.01675;
380         const double zel     =  0.05490;
381         const double c1ss    =  2.9864797e-6;
382         const double c1l     =  4.7968065e-7;
383         const double zsinis  =  0.39785416;
384         const double zcosis  =  0.91744867;
385         const double zcosgs  =  0.1945905;
386         const double zsings  = -0.98088458;
387         const double twopi   =  2.0 * pi;
388
389         /* --------------------- local variables ------------------------ */
390         int lsflg;
391         double a1    , a2    , a3    , a4    , a5    , a6    , a7    ,
392            a8    , a9    , a10   , betasq, cc    , ctem  , stem  ,
393            x1    , x2    , x3    , x4    , x5    , x6    , x7    ,
394            x8    , xnodce, xnoi  , zcosg , zcosgl, zcosh , zcoshl,
395            zcosi , zcosil, zsing , zsingl, zsinh , zsinhl, zsini ,
396            zsinil, zx    , zy;
397
398         nm     = np;
399         em     = ep;
400         snodm  = sin(nodep);
401         cnodm  = cos(nodep);
402         sinomm = sin(argpp);
403         cosomm = cos(argpp);
404         sinim  = sin(inclp);
405         cosim  = cos(inclp);
406         emsq   = em * em;
407         betasq = 1.0 - emsq;
408         rtemsq = sqrt(betasq);
409
410         /* ----------------- initialize lunar solar terms --------------- */
411         peo    = 0.0;
412         pinco  = 0.0;
413         plo    = 0.0;
414         pgho   = 0.0;
415         pho    = 0.0;
416         day    = epoch + 18261.5 + tc / 1440.0;
417         xnodce = fmod(4.5236020 - 9.2422029e-4 * day, twopi);
418         stem   = sin(xnodce);
419         ctem   = cos(xnodce);
420         zcosil = 0.91375164 - 0.03568096 * ctem;
421         zsinil = sqrt(1.0 - zcosil * zcosil);
422         zsinhl = 0.089683511 * stem / zsinil;
423         zcoshl = sqrt(1.0 - zsinhl * zsinhl);
424         gam    = 5.8351514 + 0.0019443680 * day;
425         zx     = 0.39785416 * stem / zsinil;
426         zy     = zcoshl * ctem + 0.91744867 * zsinhl * stem;
427         zx     = atan2(zx, zy);
428         zx     = gam + zx - xnodce;
429         zcosgl = cos(zx);
430         zsingl = sin(zx);
431
432         /* ------------------------ do solar terms ---------------------- */
433         zcosg = zcosgs;
434         zsing = zsings;
435         zcosi = zcosis;
436         zsini = zsinis;
437         zcosh = cnodm;
438         zsinh = snodm;
```

```
      cc    = c1ss;
      xnoi  = 1.0 / nm;

      for (lsflg = 1; lsflg <= 2; lsflg++)
        {
          a1  =   zcosg * zcosh + zsing * zcosi * zsinh;
          a3  =  -zsing * zcosh + zcosg * zcosi * zsinh;
          a7  =  -zcosg * zsinh + zsing * zcosi * zcosh;
          a8  =   zsing * zsini;
          a9  =   zsing * zsinh + zcosg * zcosi * zcosh;
          a10 =   zcosg * zsini;
          a2  =   cosim * a7 + sinim * a8;
          a4  =   cosim * a9 + sinim * a10;
          a5  =  -sinim * a7 + cosim * a8;
          a6  =  -sinim * a9 + cosim * a10;

          x1  =  a1 * cosomm + a2 * sinomm;
          x2  =  a3 * cosomm + a4 * sinomm;
          x3  = -a1 * sinomm + a2 * cosomm;
          x4  = -a3 * sinomm + a4 * cosomm;
          x5  =  a5 * sinomm;
          x6  =  a6 * sinomm;
          x7  =  a5 * cosomm;
          x8  =  a6 * cosomm;

          z31 = 12.0 * x1 * x1 - 3.0 * x3 * x3;
          z32 = 24.0 * x1 * x2 - 6.0 * x3 * x4;
          z33 = 12.0 * x2 * x2 - 3.0 * x4 * x4;
          z1  =  3.0 *  (a1 * a1 + a2 * a2) + z31 * emsq;
          z2  =  6.0 *  (a1 * a3 + a2 * a4) + z32 * emsq;
          z3  =  3.0 *  (a3 * a3 + a4 * a4) + z33 * emsq;
          z11 = -6.0 * a1 * a5 + emsq *  (-24.0 * x1 * x7-6.0 * x3 * x5);
          z12 = -6.0 *  (a1 * a6 + a3 * a5) + emsq *
                (-24.0 * (x2 * x7 + x1 * x8) - 6.0 * (x3 * x6 + x4 * x5));
          z13 = -6.0 * a3 * a6 + emsq * (-24.0 * x2 * x8 - 6.0 * x4 * x6);
          z21 =  6.0 * a2 * a5 + emsq * (24.0 * x1 * x5 - 6.0 * x3 * x7);
          z22 =  6.0 *  (a4 * a5 + a2 * a6) + emsq *
                (24.0 * (x2 * x5 + x1 * x6) - 6.0 * (x4 * x7 + x3 * x8));
          z23 =  6.0 * a4 * a6 + emsq * (24.0 * x2 * x6 - 6.0 * x4 * x8);
          z1  = z1 + z1 + betasq * z31;
          z2  = z2 + z2 + betasq * z32;
          z3  = z3 + z3 + betasq * z33;
          s3  = cc * xnoi;
          s2  = -0.5 * s3 / rtemsq;
          s4  = s3 * rtemsq;
          s1  = -15.0 * em * s4;
          s5  = x1 * x3 + x2 * x4;
          s6  = x2 * x3 + x1 * x4;
          s7  = x2 * x4 - x1 * x3;

          /* ----------------------- do lunar terms ------------------- */
          if (lsflg == 1)
            {
              ss1   = s1;
              ss2   = s2;
              ss3   = s3;
              ss4   = s4;
              ss5   = s5;
              ss6   = s6;
              ss7   = s7;
              sz1   = z1;
              sz2   = z2;
              sz3   = z3;
              sz11  = z11;
              sz12  = z12;
              sz13  = z13;
              sz21  = z21;
              sz22  = z22;
              sz23  = z23;
              sz31  = z31;
              sz32  = z32;
              sz33  = z33;
              zcosg = zcosgl;
```

```
512                 zsing = zsingl;
513                 zcosi = zcosil;
514                 zsini = zsinil;
515                 zcosh = zcoshl * cnodm + zsinhl * snodm;
516                 zsinh = snodm * zcoshl - cnodm * zsinhl;
517                 cc    = c1l;
518               }
519           }
520
521         zmol = fmod(4.7199672 + 0.22997150  * day - gam, twopi);
522         zmos = fmod(6.2565837 + 0.017201977 * day, twopi);
523
524         /* ------------------------ do solar terms ---------------------- */
525         se2  =   2.0 * ss1 * ss6;
526         se3  =   2.0 * ss1 * ss7;
527         si2  =   2.0 * ss2 * sz12;
528         si3  =   2.0 * ss2 * (sz13 - sz11);
529         sl2  =  -2.0 * ss3 * sz2;
530         sl3  =  -2.0 * ss3 * (sz3 - sz1);
531         sl4  =  -2.0 * ss3 * (-21.0 - 9.0 * emsq) * zes;
532         sgh2 =   2.0 * ss4 * sz32;
533         sgh3 =   2.0 * ss4 * (sz33 - sz31);
534         sgh4 = -18.0 * ss4 * zes;
535         sh2  =  -2.0 * ss2 * sz22;
536         sh3  =  -2.0 * ss2 * (sz23 - sz21);
537
538         /* ------------------------ do lunar terms ---------------------- */
539         ee2  =   2.0 * s1 * s6;
540         e3   =   2.0 * s1 * s7;
541         xi2  =   2.0 * s2 * z12;
542         xi3  =   2.0 * s2 * (z13 - z11);
543         xl2  =  -2.0 * s3 * z2;
544         xl3  =  -2.0 * s3 * (z3 - z1);
545         xl4  =  -2.0 * s3 * (-21.0 - 9.0 * emsq) * zel;
546         xgh2 =   2.0 * s4 * z32;
547         xgh3 =   2.0 * s4 * (z33 - z31);
548         xgh4 = -18.0 * s4 * zel;
549         xh2  =  -2.0 * s2 * z22;
550         xh3  =  -2.0 * s2 * (z23 - z21);
551
552   //#include "debug2.cpp"
553   }  // end dscom
554
555   /*-----------------------------------------------------------------------------
556   *
557   *                           procedure dsinit
558   *
559   *  this procedure provides deep space contributions to mean motion dot due
560   *    to geopotential resonance with half day and one day orbits.
561   *
562   *  author        : david vallado                  719-573-2600   28 jun 2005
563   *
564   *  inputs        :
565   *    cosim, sinim-
566   *    emsq        - eccentricity squared
567   *    argpo       - argument of perigee
568   *    s1, s2, s3, s4, s5      -
569   *    ss1, ss2, ss3, ss4, ss5 -
570   *    sz1, sz3, sz11, sz13, sz21, sz23, sz31, sz33 -
571   *    t           - time
572   *    tc          -
573   *    gsto        - greenwich sidereal time                   rad
574   *    mo          - mean anomaly
575   *    mdot        - mean anomaly dot (rate)
576   *    no          - mean motion
577   *    nodeo       - right ascension of ascending node
578   *    nodedot     - right ascension of ascending node dot (rate)
579   *    xpidot      -
580   *    z1, z3, z11, z13, z21, z23, z31, z33 -
581   *    eccm        - eccentricity
582   *    argpm       - argument of perigee
583   *    inclm       - inclination
584   *    mm          - mean anomaly
```

```
*    xn          - mean motion
*    nodem       - right ascension of ascending node
*
*  outputs       :
*    em          - eccentricity
*    argpm       - argument of perigee
*    inclm       - inclination
*    mm          - mean anomaly
*    nm          - mean motion
*    nodem       - right ascension of ascending node
*    irez        - flag for resonance           0-none, 1-one day, 2-half day
*    atime       -
*    d2201, d2211, d3210, d3222, d4410, d4422, d5220, d5232, d5421, d5433    -
*    dedt        -
*    didt        -
*    dmdt        -
*    dndt        -
*    dnodt       -
*    domdt       -
*    del1, del2, del3        -
*    ses  , sghl , sghs , sgs  , shl  , shs  , sis  , sls
*    theta       -
*    xfact       -
*    xlamo       -
*    xli         -
*    xni
*
*  locals        :
*    ainv2       -
*    aonv        -
*    cosisq      -
*    eoc         -
*    f220, f221, f311, f321, f322, f330, f441, f442, f522, f523, f542, f543  -
*    g200, g201, g211, g300, g310, g322, g410, g422, g520, g521, g532, g533  -
*    sini2       -
*    temp        -
*    temp1       -
*    theta       -
*    xno2        -
*
*  coupling      :
*    getgravconst
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

static void dsinit
     (
       gravconsttype whichconst,
       double cosim,  double emsq,   double argpo,   double s1,     double s2,
       double s3,     double s4,     double s5,      double sinim,  double ss1,
       double ss2,    double ss3,    double ss4,     double ss5,    double sz1,
       double sz3,    double sz11,   double sz13,    double sz21,   double sz23,
       double sz31,   double sz33,   double t,       double tc,     double gsto,
       double mo,     double mdot,   double no,      double nodeo,  double nodedot,
       double xpidot, double z1,     double z3,      double z11,    double z13,
       double z21,    double z23,    double z31,     double z33,    double ecco,
       double eccsq,  double& em,    double& argpm,  double& inclm, double& mm,
       double& nm,    double& nodem,
       int& irez,
       double& atime, double& d2201, double& d2211,  double& d3210, double& d3222,
       double& d4410, double& d4422, double& d5220,  double& d5232, double& d5421,
       double& d5433, double& dedt,  double& didt,   double& dmdt,  double& dndt,
       double& dnodt, double& domdt, double& del1,   double& del2,  double& del3,
       double& xfact, double& xlamo, double& xli,    double& xni
     )
{
     /* --------------------- local variables ------------------------ */
     const double twopi = 2.0 * pi;
```

```
658
659         double ainv2 , aonv=0.0, cosisq, eoc, f220 , f221  , f311  ,
660              f321  , f322  , f330  , f441  , f442  , f522  , f523  ,
661              f542  , f543  , g200  , g201  , g211  , g300  , g310  ,
662              g322  , g410  , g422  , g520  , g521  , g532  , g533  ,
663              ses   , sgs   , sghl  , sghs  , shs   , shll  , sis   ,
664              sini2 , sls   , temp  , temp1 , theta , xno2  , q22   ,
665              q31   , q33   , root22, root44, root54, rptim , root32,
666              root52, x2o3  , xke   , znl   , emo   , zns   , emsqo,
667              tumin, mu, radioTierra, j2, j3, j4, j3oj2;
668
669         q22    = 1.7891679e-6;
670         q31    = 2.1460748e-6;
671         q33    = 2.2123015e-7;
672         root22 = 1.7891679e-6;
673         root44 = 7.3636953e-9;
674         root54 = 2.1765803e-9;
675         rptim  = 4.37526908801129966e-3; // this equates to 7.29211514668855e-5 rad/sec
676         root32 = 3.7393792e-7;
677         root52 = 1.1428639e-7;
678         x2o3   = 2.0 / 3.0;
679         znl    = 1.5835218e-4;
680         zns    = 1.19459e-5;
681
682         // sgp4fix identify constants and allow alternate values
683         getgravconst( whichconst, tumin, mu, radioTierra, xke, j2, j3, j4, j3oj2 );
684
685         /* -------------------- deep space initialization ------------ */
686         irez = 0;
687         if ((nm < 0.0052359877) && (nm > 0.0034906585))
688             irez = 1;
689         if ((nm >= 8.26e-3) && (nm <= 9.24e-3) && (em >= 0.5))
690             irez = 2;
691
692         /* ------------------------ do solar terms ------------------- */
693         ses  =  ss1 * zns * ss5;
694         sis  =  ss2 * zns * (sz11 + sz13);
695         sls  = -zns * ss3 * (sz1 + sz3 - 14.0 - 6.0 * emsq);
696         sghs =  ss4 * zns * (sz31 + sz33 - 6.0);
697         shs  = -zns * ss2 * (sz21 + sz23);
698         // sgp4fix for 180 deg incl
699         if ((inclm < 5.2359877e-2) || (inclm > pi - 5.2359877e-2))
700           shs = 0.0;
701         if (sinim != 0.0)
702           shs = shs / sinim;
703         sgs  = sghs - cosim * shs;
704
705         /* ------------------------ do lunar terms ------------------ */
706         dedt = ses + s1 * znl * s5;
707         didt = sis + s2 * znl * (z11 + z13);
708         dmdt = sls - znl * s3 * (z1 + z3 - 14.0 - 6.0 * emsq);
709         sghl = s4 * znl * (z31 + z33 - 6.0);
710         shll = -znl * s2 * (z21 + z23);
711         // sgp4fix for 180 deg incl
712         if ((inclm < 5.2359877e-2) || (inclm > pi - 5.2359877e-2))
713             shll = 0.0;
714         domdt = sgs + sghl;
715         dnodt = shs;
716         if (sinim != 0.0)
717           {
718             domdt = domdt - cosim / sinim * shll;
719             dnodt = dnodt + shll / sinim;
720           }
721
722         /* ----------- calculate deep space resonance effects -------- */
723         dndt   = 0.0;
724         theta  = fmod(gsto + tc * rptim, twopi);
725         em     = em + dedt * t;
726         inclm  = inclm + didt * t;
727         argpm  = argpm + domdt * t;
728         nodem  = nodem + dnodt * t;
729         mm     = mm + dmdt * t;
730         //   sgp4fix for negative inclinations
```

```
731         //   the following if statement should be commented out
732         //if (inclm < 0.0)
733         //  {
734         //    inclm  = -inclm;
735         //    argpm  = argpm - pi;
736         //    nodem = nodem + pi;
737         //  }
738
739         /* -------------- initialize the resonance terms ------------- */
740         if (irez != 0)
741           {
742             aonv = pow(nm / xke, x2o3);
743
744             /* ---------- geopotential resonance for 12 hour orbits ------ */
745             if (irez == 2)
746               {
747                 cosisq = cosim * cosim;
748                 emo    = em;
749                 em     = ecco;
750                 emsqo  = emsq;
751                 emsq   = eccsq;
752                 eoc    = em * emsq;
753                 g201   = -0.306 - (em - 0.64) * 0.440;
754
755                 if (em <= 0.65)
756                   {
757                     g211 =    3.616  -  13.2470 * em +  16.2900 * emsq;
758                     g310 =  -19.302  + 117.3900 * em - 228.4190 * emsq +  156.5910 * eoc;
759                     g322 =  -18.9068 + 109.7927 * em - 214.6334 * emsq +  146.5816 * eoc;
760                     g410 =  -41.122  + 242.6940 * em - 471.0940 * emsq +  313.9530 * eoc;
761                     g422 = -146.407  + 841.8800 * em - 1629.014 * emsq + 1083.4350 * eoc;
762                     g520 = -532.114  + 3017.977 * em - 5740.032 * emsq + 3708.2760 * eoc;
763                   }
764                 else
765                   {
766                     g211 =   -72.099 +   331.819 * em -   508.738 * emsq +   266.724 *
                        eoc;
767                     g310 =  -346.844 +  1582.851 * em -  2415.925 * emsq +  1246.113 *
                        eoc;
768                     g322 =  -342.585 +  1554.908 * em -  2366.899 * emsq +  1215.972 *
                        eoc;
769                     g410 = -1052.797 +  4758.686 * em -  7193.992 * emsq +  3651.957 *
                        eoc;
770                     g422 = -3581.690 + 16178.110 * em - 24462.770 * emsq + 12422.520 *
                        eoc;
771                     if (em > 0.715)
772                         g520 =-5149.66 + 29936.92 * em - 54087.36 * emsq + 31324.56 *
                            eoc;
773                       else
774                         g520 = 1464.74 -  4664.75 * em +  3763.64 * emsq;
775                   }
776                 if (em < 0.7)
777                   {
778                     g533 = -919.22770 + 4988.6100 * em - 9064.7700 * emsq + 5542.21  *
                        eoc;
779                     g521 = -822.71072 + 4568.6173 * em - 8491.4146 * emsq + 5337.524 *
                        eoc;
780                     g532 = -853.66600 + 4690.2500 * em - 8624.7700 * emsq + 5341.4   *
                        eoc;
781                   }
782                 else
783                   {
784                     g533 =-37995.780 + 161616.52 * em - 229838.20 * emsq + 109377.94 *
                        eoc;
785                     g521 =-51752.104 + 218913.95 * em - 309468.16 * emsq + 146349.42 *
                        eoc;
786                     g532 =-40023.880 + 170470.89 * em - 242699.48 * emsq + 115605.82 *
                        eoc;
787                   }
788
789                 sini2=  sinim * sinim;
790                 f220 =  0.75 * (1.0 + 2.0 * cosim+cosisq);
791                 f221 =  1.5 * sini2;
```

```
792               f321 =  1.875 * sinim  *  (1.0 - 2.0 * cosim - 3.0 * cosisq);
793               f322 = -1.875 * sinim  *  (1.0 + 2.0 * cosim - 3.0 * cosisq);
794               f441 = 35.0 * sini2 * f220;
795               f442 = 39.3750 * sini2 * sini2;
796               f522 =  9.84375 * sinim * (sini2 * (1.0 - 2.0 * cosim- 5.0 * cosisq) +
797                       0.33333333 * (-2.0 + 4.0 * cosim + 6.0 * cosisq) );
798               f523 = sinim * (4.92187512 * sini2 * (-2.0 - 4.0 * cosim +
799                      10.0 * cosisq) + 6.56250012 * (1.0+2.0 * cosim - 3.0 * cosisq));
800               f542 = 29.53125 * sinim * (2.0 - 8.0 * cosim+cosisq *
801                      (-12.0 + 8.0 * cosim + 10.0 * cosisq));
802               f543 = 29.53125 * sinim * (-2.0 - 8.0 * cosim+cosisq *
803                      (12.0 + 8.0 * cosim - 10.0 * cosisq));
804               xno2  =  nm * nm;
805               ainv2 =  aonv * aonv;
806               temp1 =  3.0 * xno2 * ainv2;
807               temp  =  temp1 * root22;
808               d2201 =  temp * f220 * g201;
809               d2211 =  temp * f221 * g211;
810               temp1 =  temp1 * aonv;
811               temp  =  temp1 * root32;
812               d3210 =  temp * f321 * g310;
813               d3222 =  temp * f322 * g322;
814               temp1 =  temp1 * aonv;
815               temp  =  2.0 * temp1 * root44;
816               d4410 =  temp * f441 * g410;
817               d4422 =  temp * f442 * g422;
818               temp1 =  temp1 * aonv;
819               temp  =  temp1 * root52;
820               d5220 =  temp * f522 * g520;
821               d5232 =  temp * f523 * g532;
822               temp  =  2.0 * temp1 * root54;
823               d5421 =  temp * f542 * g521;
824               d5433 =  temp * f543 * g533;
825               xlamo =  fmod(mo + nodeo + nodeo-theta - theta, twopi);
826               xfact =  mdot + dmdt + 2.0 * (nodedot + dnodt - rptim) - no;
827               em    = emo;
828               emsq  = emsqo;
829             }
830
831           /* ---------------- synchronous resonance terms -------------- */
832           if (irez == 1)
833             {
834               g200  = 1.0 + emsq * (-2.5 + 0.8125 * emsq);
835               g310  = 1.0 + 2.0 * emsq;
836               g300  = 1.0 + emsq * (-6.0 + 6.60937 * emsq);
837               f220  = 0.75 * (1.0 + cosim) * (1.0 + cosim);
838               f311  = 0.9375 * sinim * sinim * (1.0 + 3.0 * cosim) - 0.75 * (1.0 +
                  cosim);
839               f330  = 1.0 + cosim;
840               f330  = 1.875 * f330 * f330 * f330;
841               del1  = 3.0 * nm * nm * aonv * aonv;
842               del2  = 2.0 * del1 * f220 * g200 * q22;
843               del3  = 3.0 * del1 * f330 * g300 * q33 * aonv;
844               del1  = del1 * f311 * g310 * q31 * aonv;
845               xlamo = fmod(mo + nodeo + argpo - theta, twopi);
846               xfact = mdot + xpidot - rptim + dmdt + domdt + dnodt - no;
847             }
848
849           /* ------------ for sgp4, initialize the integrator ---------- */
850           xli   = xlamo;
851           xni   = no;
852           atime = 0.0;
853           nm    = no + dndt;
854         }
855
856   //#include "debug3.cpp"
857   }  // end dsinit
858
859   /*-----------------------------------------------------------------------------
860   *
861   *                           procedure dspace
862   *
863   *  this procedure provides deep space contributions to mean elements for
```

```
*    perturbing third body.  these effects have been averaged over one
*    revolution of the sun and moon.  for earth resonance effects, the
*    effects have been averaged over no revolutions of the satellite.
*    (mean motion)
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    d2201, d2211, d3210, d3222, d4410, d4422, d5220, d5232, d5421, d5433 -
*    dedt        -
*    del1, del2, del3  -
*    didt        -
*    dmdt        -
*    dnodt       -
*    domdt       -
*    irez        - flag for resonance           0-none, 1-one day, 2-half day
*    argpo       - argument of perigee
*    argpdot     - argument of perigee dot (rate)
*    t           - time
*    tc          -
*    gsto        - gst
*    xfact       -
*    xlamo       -
*    no          - mean motion
*    atime       -
*    em          - eccentricity
*    ft          -
*    argpm       - argument of perigee
*    inclm       - inclination
*    xli         -
*    mm          - mean anomaly
*    xni         - mean motion
*    nodem       - right ascension of ascending node
*
*  outputs       :
*    atime       -
*    em          - eccentricity
*    argpm       - argument of perigee
*    inclm       - inclination
*    xli         -
*    mm          - mean anomaly
*    xni         -
*    nodem       - right ascension of ascending node
*    dndt        -
*    nm          - mean motion
*
*  locals        :
*    delt        -
*    ft          -
*    theta       -
*    x2li        -
*    x2omi       -
*    xl          -
*    xldot       -
*    xnddt       -
*    xndt        -
*    xomi        -
*
*  coupling      :
*    none        -
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------*/

static void dspace
     (
       int irez,
       double d2201,  double d2211,  double d3210,   double d3222,  double d4410,
       double d4422,  double d5220,  double d5232,   double d5421,  double d5433,
```

```
          double dedt,   double del1,   double del2,   double del3,   double didt,
          double dmdt,   double dnodt,  double domdt,   double argpo,  double argpdot,
          double t,      double tc,     double gsto,    double xfact,  double xlamo,
          double no,
          double& atime, double& em,    double& argpm,  double& inclm, double& xli,
          double& mm,    double& xni,   double& nodem,  double& dndt,  double& nm
        )
{
        const double twopi = 2.0 * pi;
        int iretn , iret;
        double delt, ft, theta, x2li, x2omi, xl, xldot , xnddt, xndt, xomi, g22, g32,
             g44, g52, g54, fasx2, fasx4, fasx6, rptim , step2, stepn , stepp;

        fasx2 = 0.13130908;
        fasx4 = 2.8843198;
        fasx6 = 0.37448087;
        g22   = 5.7686396;
        g32   = 0.95240898;
        g44   = 1.8014998;
        g52   = 1.0508330;
        g54   = 4.4108898;
        rptim = 4.37526908801129966e-3; // this equates to 7.29211514668855e-5 rad/sec
        stepp =    720.0;
        stepn =   -720.0;
        step2 = 259200.0;

        /* ----------- calculate deep space resonance effects ----------- */
        dndt   = 0.0;
        theta  = fmod(gsto + tc * rptim, twopi);
        em     = em + dedt * t;

        inclm  = inclm + didt * t;
        argpm  = argpm + domdt * t;
        nodem  = nodem + dnodt * t;
        mm     = mm + dmdt * t;

        //   sgp4fix for negative inclinations
        //   the following if statement should be commented out
        //  if (inclm < 0.0)
        // {
        //    inclm = -inclm;
        //    argpm = argpm - pi;
        //    nodem = nodem + pi;
        //  }

        /* - update resonances : numerical (euler-maclaurin) integration - */
        /* ------------------------- epoch restart ----------------------  */
        //   sgp4fix for propagator problems
        //   the following integration works for negative time steps and periods
        //   the specific changes are unknown because the original code was so convoluted

        // sgp4fix take out atime = 0.0 and fix for faster operation
        ft    = 0.0;
        if (irez != 0)
          {
            // sgp4fix streamline check
            if ((atime == 0.0) || (t * atime <= 0.0) || (fabs(t) < fabs(atime)) )
              {
                atime  = 0.0;
                xni    = no;
                xli    = xlamo;
              }
            // sgp4fix move check outside loop
            if (t > 0.0)
                delt = stepp;
              else
                delt = stepn;

            iretn = 381; // added for do loop
            iret  =   0; // added for loop
            while (iretn == 381)
              {
                /* ------------------ dot terms calculated ------------ */
```

```
1010              /* ----------- near - synchronous resonance terms ------- */
1011              if (irez != 2)
1012                {
1013                  xndt  = del1 * sin(xli - fasx2) + del2 * sin(2.0 * (xli - fasx4)) +
1014                          del3 * sin(3.0 * (xli - fasx6));
1015                  xldot = xni + xfact;
1016                  xnddt = del1 * cos(xli - fasx2) +
1017                          2.0 * del2 * cos(2.0 * (xli - fasx4)) +
1018                          3.0 * del3 * cos(3.0 * (xli - fasx6));
1019                  xnddt = xnddt * xldot;
1020                }
1021                else
1022                {
1023                  /* --------- near - half-day resonance terms -------- */
1024                  xomi  = argpo + argpdot * atime;
1025                  x2omi = xomi + xomi;
1026                  x2li  = xli + xli;
1027                  xndt  = d2201 * sin(x2omi + xli - g22) + d2211 * sin(xli - g22) +
1028                        d3210 * sin(xomi + xli - g32)  + d3222 * sin(-xomi + xli -
                          g32)+
1029                        d4410 * sin(x2omi + x2li - g44)+ d4422 * sin(x2li - g44) +
1030                        d5220 * sin(xomi + xli - g52)  + d5232 * sin(-xomi + xli -
                          g52)+
1031                        d5421 * sin(xomi + x2li - g54) + d5433 * sin(-xomi + x2li -
                          g54);
1032                  xldot = xni + xfact;
1033                  xnddt = d2201 * cos(x2omi + xli - g22) + d2211 * cos(xli - g22) +
1034                        d3210 * cos(xomi + xli - g32) + d3222 * cos(-xomi + xli -
                          g32) +
1035                        d5220 * cos(xomi + xli - g52) + d5232 * cos(-xomi + xli -
                          g52) +
1036                        2.0 * (d4410 * cos(x2omi + x2li - g44) +
1037                        d4422 * cos(x2li - g44) + d5421 * cos(xomi + x2li - g54) +
1038                        d5433 * cos(-xomi + x2li - g54));
1039                  xnddt = xnddt * xldot;
1040                }
1041
1042              /* ----------------------- integrator ------------------- */
1043              // sgp4fix move end checks to end of routine
1044              if (fabs(t - atime) >= stepp)
1045                {
1046                  iret  = 0;
1047                  iretn = 381;
1048                }
1049                else // exit here
1050                {
1051                  ft    = t - atime;
1052                  iretn = 0;
1053                }
1054
1055              if (iretn == 381)
1056                {
1057                  xli   = xli + xldot * delt + xndt * step2;
1058                  xni   = xni + xndt * delt + xnddt * step2;
1059                  atime = atime + delt;
1060                }
1061            }  // while iretn = 381
1062
1063            nm = xni + xndt * ft + xnddt * ft * ft * 0.5;
1064            xl = xli + xldot * ft + xndt * ft * ft * 0.5;
1065            if (irez != 1)
1066              {
1067                mm   = xl - 2.0 * nodem + 2.0 * theta;
1068                dndt = nm - no;
1069              }
1070              else
1071              {
1072                mm   = xl - nodem - argpm + theta;
1073                dndt = nm - no;
1074              }
1075            nm = no + dndt;
1076          }
1077
```

```
//#include "debug4.cpp"
}  // end dsspace

/*-----------------------------------------------------------------------------
*
*                           procedure initl
*
*  this procedure initializes the spg4 propagator. all the initialization is
*    consolidated here instead of having multiple loops inside other routines.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    ecco        - eccentricity                           0.0 - 1.0
*    epoch       - epoch time in days from jan 0, 1950. 0 hr
*    inclo       - inclination of satellite
*    no          - mean motion of satellite
*    satn        - satellite number
*
*  outputs       :
*    ainv        - 1.0 / a
*    ao          - semi major axis
*    con41       -
*    con42       - 1.0 - 5.0 cos(i)
*    cosio       - cosine of inclination
*    cosio2      - cosio squared
*    eccsq       - eccentricity squared
*    method      - flag for deep space                    'd', 'n'
*    omeosq      - 1.0 - ecco * ecco
*    posq        - semi-parameter squared
*    rp          - radius of perigee
*    rteosq      - square root of (1.0 - ecco*ecco)
*    sinio       - sine of inclination
*    gsto        - gst at time of observation               rad
*    no          - mean motion of satellite
*
*  locals        :
*    ak          -
*    d1          -
*    del         -
*    adel        -
*    po          -
*
*  coupling      :
*    getgravconst
*    gstime      - find greenwich sidereal time from the julian date
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

static void initl
     (
       int satn,      gravconsttype whichconst,
       double ecco,   long int epoch,  double epoch_fraccion,
       double inclo,  double& no, char& method, double& ainv,
       double& ao,    double& con41,  double& con42, double& cosio,
       double& cosio2, double& eccsq, double& omeosq, double& posq,
       double& rp,    double& rteosq,double& sinio , double& gsto,
       char opsmode
     )
{
     /* --------------------- local variables ------------------------ */
     double ak, d1, del, adel, po, x2o3, j2, xke,
            tumin, mu, radiusearthkm, j3, j4, j3oj2;

     // sgp4fix use old way of finding gst
     double ds70;
     double ts70, /*tfrac,*/ c1, thgr70, fk5r, c1p2p, ts70_2;
     const double twopi = 2.0 * pi;
```

```
1151
1152        /* ----------------------- earth constants ---------------------- */
1153        // sgp4fix identify constants and allow alternate values
1154        getgravconst( whichconst, tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2 );
1155        x2o3   = 2.0 / 3.0;
1156
1157        /* ------------ calculate auxillary epoch quantities ---------- */
1158        eccsq  = ecco * ecco;
1159        omeosq = 1.0 - eccsq;
1160        rteosq = sqrt(omeosq);
1161        cosio  = cos(inclo);
1162        cosio2 = cosio * cosio;
1163
1164        /* ------------------ un-kozai the mean motion ----------------- */
1165        ak    = pow(xke / no, x2o3);
1166        d1    = 0.75 * j2 * (3.0 * cosio2 - 1.0) / (rteosq * omeosq);
1167        del   = d1 / (ak * ak);
1168        adel  = ak * (1.0 - del * del - del *
1169                (1.0 / 3.0 + 134.0 * del * del / 81.0));
1170        del   = d1/(adel * adel);
1171        no    = no / (1.0 + del);
1172
1173        ao    = pow(xke / no, x2o3);
1174        sinio = sin(inclo);
1175        po    = ao * omeosq;
1176        con42 = 1.0 - 5.0 * cosio2;
1177        con41 = -con42-cosio2-cosio2;
1178        ainv  = 1.0 / ao;
1179        posq  = po * po;
1180        rp    = ao * (1.0 - ecco);
1181        method = 'n';
1182
1183        // sgp4fix modern approach to finding sidereal time
1184        if (opsmode == 'a')
1185           {
1186            // sgp4fix use old way of finding gst
1187            // count integer number of days from 0 jan 1970
1188
1189            /*
1190            ts70  = epoch - 7305.0;
1191            ds70 = floor(ts70 + 1.0e-8);
1192            tfrac = ts70 - ds70;
1193            */
1194
1195            ds70 = epoch - 7305.0;
1196            ts70 = ds70 + epoch_fraccion;
1197            // find greenwich location at epoch
1198            c1    = 1.72027916940703639e-2;
1199            thgr70= 1.7321343856509374;
1200            fk5r  = 5.07555419432269442e-15;
1201            c1p2p = c1 + twopi;
1202            gsto  = fmod( thgr70 + c1*ds70 + c1p2p*epoch_fraccion + ts70*ts70*fk5r,
                twopi);
1203            if ( gsto < 0.0 )
1204                gsto = gsto + twopi;
1205          }
1206          else
1207           gsto = gstime(epoch + 2433281.5);
1208
1209
1210  //#include "debug5.cpp"
1211  }  // end initl
1212
1213  /*-----------------------------------------------------------------------
1214  *
1215  *                           procedure sgp4init
1216  *
1217  *  this procedure initializes variables for sgp4.
1218  *
1219  *  author        : david vallado                  719-573-2600   28 jun 2005
1220  *
1221  *  inputs        :
1222  *    opsmode     - mode of operation afspc or improved 'a', 'i'
```

```
1223   *    whichconst  - which set of constants to use  72, 84
1224   *    satn        - satellite number
1225   *    bstar       - sgp4 type drag coefficient              kg/m2er
1226   *    ecco        - eccentricity
1227   *    epoch       - epoch time in days from jan 0, 1950. 0 hr
1228   *    argpo       - argument of perigee (output if ds)
1229   *    inclo       - inclination
1230   *    mo          - mean anomaly (output if ds)
1231   *    no          - mean motion
1232   *    nodeo       - right ascension of ascending node
1233   *
1234   *  outputs       :
1235   *    satrec      - common values for subsequent calls
1236   *    return code - non-zero on error.
1237   *                   1 - mean elements, ecc >= 1.0 or ecc < -0.001 or a < 0.95 er
1238   *                   2 - mean motion less than 0.0
1239   *                   3 - pert elements, ecc < 0.0  or  ecc > 1.0
1240   *                   4 - semi-latus rectum < 0.0
1241   *                   5 - epoch elements are sub-orbital
1242   *                   6 - satellite has decayed
1243   *
1244   *  locals        :
1245   *    cnodm  , snodm  , cosim  , sinim  , cosomm , sinomm
1246   *    cc1sq  , cc2    , cc3
1247   *    coef   , coef1
1248   *    cosio4      -
1249   *    day         -
1250   *    dndt        -
1251   *    em          - eccentricity
1252   *    emsq        - eccentricity squared
1253   *    eeta        -
1254   *    etasq       -
1255   *    gam         -
1256   *    argpm       - argument of perigee
1257   *    nodem       -
1258   *    inclm       - inclination
1259   *    mm          - mean anomaly
1260   *    nm          - mean motion
1261   *    perige      - perigee
1262   *    pinvsq      -
1263   *    psisq       -
1264   *    qzms24      -
1265   *    rtemsq      -
1266   *    s1, s2, s3, s4, s5, s6, s7          -
1267   *    sfour       -
1268   *    ss1, ss2, ss3, ss4, ss5, ss6, ss7         -
1269   *    sz1, sz2, sz3
1270   *    sz11, sz12, sz13, sz21, sz22, sz23, sz31, sz32, sz33        -
1271   *    tc          -
1272   *    temp        -
1273   *    temp1, temp2, temp3       -
1274   *    tsi         -
1275   *    xpidot      -
1276   *    xhdot1      -
1277   *    z1, z2, z3          -
1278   *    z11, z12, z13, z21, z22, z23, z31, z32, z33         -
1279   *
1280   *  coupling      :
1281   *    getgravconst-
1282   *    initl       -
1283   *    dscom       -
1284   *    dpper       -
1285   *    dsinit      -
1286   *    sgp4        -
1287   *
1288   *  references    :
1289   *    hoots, roehrich, norad spacetrack report #3 1980
1290   *    hoots, norad spacetrack report #6 1986
1291   *    hoots, schumacher and glover 2004
1292   *    vallado, crawford, hujsak, kelso  2006
1293     ----------------------------------------------------------------------*/
1294
1295   bool sgp4init
```

```
1296        (
1297          gravconsttype whichconst, char opsmode, const int satn,
1298          const long int epoch, const double epoch_fraccion,
1299          const double xbstar,  const double xecco,
1300          const double xargpo, const double xinclo,  const double xmo,
1301          const double xno, const double xnodeo,  elsetrec& satrec
1302        )
1303    {
1304        /* ---------------------- local variables ------------------------ */
1305        double ao, ainv,   con42, cosio, sinio, cosio2, eccsq,
1306             omeosq, posq,   rp,     rteosq,
1307             cnodm , snodm , cosim , sinim , cosomm, sinomm, cc1sq ,
1308             cc2   , cc3   , coef  , coef1 , cosio4, day   , dndt  ,
1309             em    , emsq  , eeta  , etasq , gam   , argpm , nodem ,
1310             inclm , mm    , nm    , perige, pinvsq, psisq , qzms24,
1311             rtemsq, s1    , s2    , s3    , s4    , s5    , s6    ,
1312             s7    , sfour , ss1   , ss2   , ss3   , ss4   , ss5   ,
1313             ss6   , ss7   , sz1   , sz2   , sz3   , sz11  , sz12  ,
1314             sz13  , sz21  , sz22  , sz23  , sz31  , sz32  , sz33  ,
1315             tc    , temp  , temp1 , temp2 , temp3 , tsi   , xpidot,
1316             xhdot1, z1    , z2    , z3    , z11   , z12   , z13   ,
1317             z21   , z22   , z23   , z31   , z32   , z33,
1318             qzms2t, ss, j2, j3oj2, j4, x2o3, r[3], v[3],
1319             tumin, mu, radiusearthkm, xke, j3, delmotemp, qzms2ttemp, qzms24temp;
1320         double epoca;
1321
1322        /* ------------------------ initialization --------------------- */
1323        // sgp4fix divisor for divide by zero check on inclination
1324        // the old check used 1.0 + cos(pi-1.0e-9), but then compared it to
1325        // 1.5 e-12, so the threshold was changed to 1.5e-12 for consistency
1326        const double temp4    =   1.5e-12;
1327
1328        /* ----------- set all near earth variables to zero ------------ */
1329        satrec.isimp   = 0;   satrec.method = 'n'; satrec.aycof    = 0.0;
1330        satrec.con41   = 0.0; satrec.cc1    = 0.0; satrec.cc4      = 0.0;
1331        satrec.cc5     = 0.0; satrec.d2     = 0.0; satrec.d3       = 0.0;
1332        satrec.d4      = 0.0; satrec.delmo  = 0.0; satrec.eta      = 0.0;
1333        satrec.argpdot = 0.0; satrec.omgcof = 0.0; satrec.sinmao   = 0.0;
1334        satrec.t       = 0.0; satrec.t2cof  = 0.0; satrec.t3cof    = 0.0;
1335        satrec.t4cof   = 0.0; satrec.t5cof  = 0.0; satrec.x1mth2   = 0.0;
1336        satrec.x7thm1  = 0.0; satrec.mdot   = 0.0; satrec.nodedot  = 0.0;
1337        satrec.xlcof   = 0.0; satrec.xmcof  = 0.0; satrec.nodecf   = 0.0;
1338
1339        /* ----------- set all deep space variables to zero ------------ */
1340        satrec.irez  = 0;   satrec.d2201 = 0.0; satrec.d2211 = 0.0;
1341        satrec.d3210 = 0.0; satrec.d3222 = 0.0; satrec.d4410 = 0.0;
1342        satrec.d4422 = 0.0; satrec.d5220 = 0.0; satrec.d5232 = 0.0;
1343        satrec.d5421 = 0.0; satrec.d5433 = 0.0; satrec.dedt  = 0.0;
1344        satrec.del1  = 0.0; satrec.del2  = 0.0; satrec.del3  = 0.0;
1345        satrec.didt  = 0.0; satrec.dmdt  = 0.0; satrec.dnodt = 0.0;
1346        satrec.domdt = 0.0; satrec.e3    = 0.0; satrec.ee2   = 0.0;
1347        satrec.peo   = 0.0; satrec.pgho  = 0.0; satrec.pho   = 0.0;
1348        satrec.pinco = 0.0; satrec.plo   = 0.0; satrec.se2   = 0.0;
1349        satrec.se3   = 0.0; satrec.sgh2  = 0.0; satrec.sgh3  = 0.0;
1350        satrec.sgh4  = 0.0; satrec.sh2   = 0.0; satrec.sh3   = 0.0;
1351        satrec.si2   = 0.0; satrec.si3   = 0.0; satrec.sl2   = 0.0;
1352        satrec.sl3   = 0.0; satrec.sl4   = 0.0; satrec.gsto  = 0.0;
1353        satrec.xfact = 0.0; satrec.xgh2  = 0.0; satrec.xgh3  = 0.0;
1354        satrec.xgh4  = 0.0; satrec.xh2   = 0.0; satrec.xh3   = 0.0;
1355        satrec.xi2   = 0.0; satrec.xi3   = 0.0; satrec.xl2   = 0.0;
1356        satrec.xl3   = 0.0; satrec.xl4   = 0.0; satrec.xlamo = 0.0;
1357        satrec.zmol  = 0.0; satrec.zmos  = 0.0; satrec.atime = 0.0;
1358        satrec.xli   = 0.0; satrec.xni   = 0.0;
1359
1360        // sgp4fix - note the following variables are also passed directly via satrec.
1361        // it is possible to streamline the sgp4init call by deleting the "x"
1362        // variables, but the user would need to set the satrec.* values first. we
1363        // include the additional assignments in case twoline2rv is not used.
1364        satrec.bstar   = xbstar;
1365        satrec.ecco    = xecco;
1366        satrec.argpo   = xargpo;
1367        satrec.inclo   = xinclo;
1368        satrec.mo      = xmo;
```

```
1369        satrec.no      = xno;
1370        satrec.nodeo   = xnodeo;
1371
1372        // sgp4fix add opsmode
1373        satrec.operationmode = opsmode;
1374
1375        /* ------------------------ earth constants ----------------------- */
1376        // sgp4fix identify constants and allow alternate values
1377        getgravconst( whichconst, tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2 );
1378        ss     = 78.0 / radiusearthkm + 1.0;
1379        // sgp4fix use multiply for speed instead of pow
1380        qzms2ttemp = (120.0 - 78.0) / radiusearthkm;
1381        qzms2t = qzms2ttemp * qzms2ttemp * qzms2ttemp * qzms2ttemp;
1382        x2o3   =  2.0 / 3.0;
1383
1384        satrec.init = 'y';
1385        satrec.t    = 0.0;
1386
1387        initl
1388            (
1389              satn, whichconst, satrec.ecco, epoch, epoch_fraccion,
1390              satrec.inclo, satrec.no, satrec.method,
1391              ainv, ao, satrec.con41, con42, cosio, cosio2, eccsq, omeosq,
1392              posq, rp, rteosq, sinio, satrec.gsto, satrec.operationmode
1393            );
1394        satrec.error = 0;
1395
1396        // sgp4fix remove this check as it is unnecessary
1397        // the mrt check in sgp4 handles decaying satellite cases even if the starting
1398        // condition is below the surface of te earth
1399  //      if (rp < 1.0)
1400  //        {
1401  //          printf("# *** satn%d epoch elts sub-orbital ***\n", satn);
1402  //          satrec.error = 5;
1403  //        }
1404
1405        if ((omeosq >= 0.0 ) || ( satrec.no >= 0.0))
1406          {
1407            satrec.isimp = 0;
1408            if (rp < (220.0 / radiusearthkm + 1.0))
1409                satrec.isimp = 1;
1410            sfour  = ss;
1411            qzms24 = qzms2t;
1412            perige = (rp - 1.0) * radiusearthkm;
1413
1414            /* - for perigees below 156 km, s and qoms2t are altered - */
1415            if (perige < 156.0)
1416              {
1417                sfour = perige - 78.0;
1418                if (perige < 98.0)
1419                    sfour = 20.0;
1420                // sgp4fix use multiply for speed instead of pow
1421                qzms24temp =  (120.0 - sfour) / radiusearthkm;
1422                qzms24 = qzms24temp * qzms24temp * qzms24temp * qzms24temp;
1423                sfour  = sfour / radiusearthkm + 1.0;
1424              }
1425            pinvsq = 1.0 / posq;
1426
1427            tsi  = 1.0 / (ao - sfour);
1428            satrec.eta  = ao * satrec.ecco * tsi;
1429            etasq = satrec.eta * satrec.eta;
1430            eeta  = satrec.ecco * satrec.eta;
1431            psisq = fabs(1.0 - etasq);
1432            coef  = qzms24 * pow(tsi, 4.0);
1433            coef1 = coef / pow(psisq, 3.5);
1434            cc2   = coef1 * satrec.no * (ao * (1.0 + 1.5 * etasq + eeta *
1435                          (4.0 + etasq)) + 0.375 * j2 * tsi / psisq * satrec.con41 *
1436                          (8.0 + 3.0 * etasq * (8.0 + etasq)));
1437            satrec.cc1   = satrec.bstar * cc2;
1438            cc3   = 0.0;
1439            if (satrec.ecco > 1.0e-4)
1440                cc3 = -2.0 * coef * tsi * j3oj2 * satrec.no * sinio / satrec.ecco;
1441            satrec.x1mth2 = 1.0 - cosio2;
```

```
            satrec.cc4    = 2.0* satrec.no * coef1 * ao * omeosq *
                            (satrec.eta * (2.0 + 0.5 * etasq) + satrec.ecco *
                            (0.5 + 2.0 * etasq) - j2 * tsi / (ao * psisq) *
                            (-3.0 * satrec.con41 * (1.0 - 2.0 * eeta + etasq *
                            (1.5 - 0.5 * eeta)) + 0.75 * satrec.x1mth2 *
                            (2.0 * etasq - eeta * (1.0 + etasq)) * cos(2.0 *
                            satrec.argpo)));
            satrec.cc5 = 2.0 * coef1 * ao * omeosq * (1.0 + 2.75 *
                         (etasq + eeta) + eeta * etasq);
            cosio4 = cosio2 * cosio2;
            temp1  = 1.5 * j2 * pinvsq * satrec.no;
            temp2  = 0.5 * temp1 * j2 * pinvsq;
            temp3  = -0.46875 * j4 * pinvsq * pinvsq * satrec.no;
            satrec.mdot     = satrec.no + 0.5 * temp1 * rteosq * satrec.con41 + 0.0625 *
                              temp2 * rteosq * (13.0 - 78.0 * cosio2 + 137.0 * cosio4);
            satrec.argpdot  = -0.5 * temp1 * con42 + 0.0625 * temp2 *
                              (7.0 - 114.0 * cosio2 + 395.0 * cosio4) +
                              temp3 * (3.0 - 36.0 * cosio2 + 49.0 * cosio4);
            xhdot1            = -temp1 * cosio;
            satrec.nodedot = xhdot1 + (0.5 * temp2 * (4.0 - 19.0 * cosio2) +
                             2.0 * temp3 * (3.0 - 7.0 * cosio2)) * cosio;
            xpidot            =  satrec.argpdot+ satrec.nodedot;
            satrec.omgcof   = satrec.bstar * cc3 * cos(satrec.argpo);
            satrec.xmcof    = 0.0;
            if (satrec.ecco > 1.0e-4)
                satrec.xmcof = -x2o3 * coef * satrec.bstar / eeta;
            satrec.nodecf = 3.5 * omeosq * xhdot1 * satrec.cc1;
            satrec.t2cof   = 1.5 * satrec.cc1;
            // sgp4fix for divide by zero with xinco = 180 deg
            if (fabs(cosio+1.0) > 1.5e-12)
                satrec.xlcof = -0.25 * j3oj2 * sinio * (3.0 + 5.0 * cosio) / (1.0 +
                cosio);
              else
                satrec.xlcof = -0.25 * j3oj2 * sinio * (3.0 + 5.0 * cosio) / temp4;
            satrec.aycof   = -0.5 * j3oj2 * sinio;
            // sgp4fix use multiply for speed instead of pow
            delmotemp = 1.0 + satrec.eta * cos(satrec.mo);
            satrec.delmo   = delmotemp * delmotemp * delmotemp;
            satrec.sinmao  = sin(satrec.mo);
            satrec.x7thm1  = 7.0 * cosio2 - 1.0;

            /* --------------- deep space initialization ------------ */
            if ((2*pi / satrec.no) >= 225.0)
              {
                satrec.method = 'd';
                satrec.isimp  = 1;
                tc    =  0.0;
                inclm = satrec.inclo;
                epoca = epoch + epoch_fraccion;
                dscom
                    (
                      epoca, satrec.ecco, satrec.argpo, tc, satrec.inclo, satrec.nodeo,
                      satrec.no, snodm, cnodm,  sinim, cosim,sinomm,     cosomm,
                      day, satrec.e3, satrec.ee2, em,         emsq, gam,
                      satrec.peo,  satrec.pgho,   satrec.pho, satrec.pinco,
                      satrec.plo,  rtemsq,        satrec.se2, satrec.se3,
                      satrec.sgh2, satrec.sgh3,   satrec.sgh4,
                      satrec.sh2,  satrec.sh3,    satrec.si2, satrec.si3,
                      satrec.sl2,  satrec.sl3,    satrec.sl4, s1, s2, s3, s4, s5,
                      s6,   s7,   ss1,  ss2,  ss3,  ss4,  ss5,  ss6,  ss7, sz1, sz2, sz3,
                      sz11, sz12, sz13, sz21, sz22, sz23, sz31, sz32, sz33,
                      satrec.xgh2, satrec.xgh3,   satrec.xgh4, satrec.xh2,
                      satrec.xh3,  satrec.xi2,    satrec.xi3,  satrec.xl2,
                      satrec.xl3,  satrec.xl4,    nm, z1, z2, z3, z11,
                      z12, z13, z21, z22, z23, z31, z32, z33,
                      satrec.zmol, satrec.zmos
                    );
                dpper
                    (
                      satrec.e3, satrec.ee2, satrec.peo, satrec.pgho,
                      satrec.pho, satrec.pinco, satrec.plo, satrec.se2,
                      satrec.se3, satrec.sgh2, satrec.sgh3, satrec.sgh4,
                      satrec.sh2, satrec.sh3, satrec.si2,  satrec.si3,
```

```
                    satrec.sl2, satrec.sl3, satrec.sl4,  satrec.t,
                    satrec.xgh2,satrec.xgh3,satrec.xgh4, satrec.xh2,
                    satrec.xh3, satrec.xi2, satrec.xi3,  satrec.xl2,
                    satrec.xl3, satrec.xl4, satrec.zmol, satrec.zmos, inclm,
                    satrec.init,
                    satrec.ecco, satrec.inclo, satrec.nodeo, satrec.argpo, satrec.mo,
                    satrec.operationmode
                  );

              argpm  = 0.0;
              nodem  = 0.0;
              mm     = 0.0;

              dsinit
                  (
                    whichconst,
                    cosim, emsq, satrec.argpo, s1, s2, s3, s4, s5, sinim, ss1, ss2,
                    ss3, ss4,
                    ss5, sz1, sz3, sz11, sz13, sz21, sz23, sz31, sz33, satrec.t, tc,
                    satrec.gsto, satrec.mo, satrec.mdot, satrec.no, satrec.nodeo,
                    satrec.nodedot, xpidot, z1, z3, z11, z13, z21, z23, z31, z33,
                    satrec.ecco, eccsq, em, argpm, inclm, mm, nm, nodem,
                    satrec.irez,  satrec.atime,
                    satrec.d2201, satrec.d2211, satrec.d3210, satrec.d3222 ,
                    satrec.d4410, satrec.d4422, satrec.d5220, satrec.d5232,
                    satrec.d5421, satrec.d5433, satrec.dedt,  satrec.didt,
                    satrec.dmdt,  dndt,         satrec.dnodt, satrec.domdt ,
                    satrec.del1,  satrec.del2,  satrec.del3,  satrec.xfact,
                    satrec.xlamo, satrec.xli,   satrec.xni
                  );
            }

        /* ----------- set variables if not deep space ----------- */
        if (satrec.isimp != 1)
          {
            cc1sq          = satrec.cc1 * satrec.cc1;
            satrec.d2    = 4.0 * ao * tsi * cc1sq;
            temp           = satrec.d2 * tsi * satrec.cc1 / 3.0;
            satrec.d3    = (17.0 * ao + sfour) * temp;
            satrec.d4    = 0.5 * temp * ao * tsi * (221.0 * ao + 31.0 * sfour) *
                           satrec.cc1;
            satrec.t3cof = satrec.d2 + 2.0 * cc1sq;
            satrec.t4cof = 0.25 * (3.0 * satrec.d3 + satrec.cc1 *
                           (12.0 * satrec.d2 + 10.0 * cc1sq));
            satrec.t5cof = 0.2 * (3.0 * satrec.d4 +
                           12.0 * satrec.cc1 * satrec.d3 +
                           6.0 * satrec.d2 * satrec.d2 +
                           15.0 * cc1sq * (2.0 * satrec.d2 + cc1sq));
          }
        } // if omeosq = 0 ...

        /* finally propogate to zero epoch to initialize all others. */
        // sgp4fix take out check to let satellites process until they are actually
        below earth surface
//        if(satrec.error == 0)
        sgp4(whichconst, satrec, 0.0, r, v);

        satrec.init = 'n';

//#include "debug6.cpp"
        //sgp4fix return boolean. satrec.error contains any error codes
        return true;
    }  // end sgp4init

    /*-----------------------------------------------------------------------------
    *
    *                           procedure sgp4
    *
    *  this procedure is the sgp4 prediction model from space command. this is an
    *    updated and combined version of sgp4 and sdp4, which were originally
    *    published separately in spacetrack report #3. this version follows the
    *    methodology from the aiaa paper (2006) describing the history and
    *    development of the code.
```

```
1583  *
1584  *  author        : david vallado                  719-573-2600   28 jun 2005
1585  *
1586  *  inputs        :
1587  *    satrec  - initialised structure from sgp4init() call.
1588  *    tsince  - time eince epoch (minutes)
1589  *
1590  *  outputs       :
1591  *    r           - position vector                     km
1592  *    v           - velocity                            km/sec
1593  *  return code - non-zero on error.
1594  *                   1 - mean elements, ecc >= 1.0 or ecc < -0.001 or a < 0.95 er
1595  *                   2 - mean motion less than 0.0
1596  *                   3 - pert elements, ecc < 0.0  or  ecc > 1.0
1597  *                   4 - semi-latus rectum < 0.0
1598  *                   5 - epoch elements are sub-orbital
1599  *                   6 - satellite has decayed
1600  *
1601  *  locals        :
1602  *    am          -
1603  *    axnl, aynl        -
1604  *    betal       -
1605  *    cosim   , sinim   , cosomm  , sinomm  , cnod    , snod    , cos2u   ,
1606  *    sin2u   , coseo1  , sineo1  , cosi    , sini    , cosip   , sinip   ,
1607  *    cosisq  , cossu   , sinsu   , cosu    , sinu
1608  *    delm        -
1609  *    delomg      -
1610  *    dndt        -
1611  *    eccm        -
1612  *    emsq        -
1613  *    ecose       -
1614  *    el2         -
1615  *    eo1         -
1616  *    eccp        -
1617  *    esine       -
1618  *    argpm       -
1619  *    argpp       -
1620  *    omgadf      -
1621  *    pl          -
1622  *    r           -
1623  *    rtemsq      -
1624  *    rdotl       -
1625  *    rl          -
1626  *    rvdot       -
1627  *    rvdotl      -
1628  *    su          -
1629  *    t2  , t3   , t4    , tc
1630  *    tem5, temp , temp1 , temp2  , tempa  , tempe  , templ
1631  *    u   , ux   , uy    , uz     , vx     , vy     , vz
1632  *    inclm       - inclination
1633  *    mm          - mean anomaly
1634  *    nm          - mean motion
1635  *    nodem       - right asc of ascending node
1636  *    xinc        -
1637  *    xincp       -
1638  *    xl          -
1639  *    xlm         -
1640  *    mp          -
1641  *    xmdf        -
1642  *    xmx         -
1643  *    xmy         -
1644  *    nodedf      -
1645  *    xnode       -
1646  *    nodep       -
1647  *    np          -
1648  *
1649  *  coupling      :
1650  *    getgravconst-
1651  *    dpper
1652  *    dpspace
1653  *
1654  *  references    :
1655  *    hoots, roehrich, norad spacetrack report #3 1980
```

```
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

bool sgp4
     (
       gravconsttype whichconst, elsetrec& satrec,  double tsince,
       double r[3],  double v[3]
     )
{
     double am   , axnl  , aynl , betal ,  cosim , cnod  ,
         cos2u, coseo1, cosi , cosip ,  cosisq, cossu , cosu,
         delm , delomg, em   , emsq  ,  ecose , el2   , eo1 ,
         ep   , esine , argpm, argpp ,  argpdf, pl,     mrt = 0.0,
         mvt  , rdotl , rl   , rvdot ,  rvdotl, sinim ,
         sin2u, sineo1, sini , sinip ,  sinsu , sinu  ,
         snod , su    , t2   , t3    ,  t4    , tem5  , temp,
         temp1, temp2 , tempa, tempe ,  templ , u     , ux  ,
         uy   , uz    , vx   , vy    ,  vz    , inclm , mm  ,
         nm   , nodem, xinc , xincp ,  xl    , xlm   , mp  ,
         xmdf , xmx   , xmy  , nodedf, xnode , nodep, tc  , dndt,
         twopi, x2o3  , j2   , j3    , tumin, j4 , xke   , j3oj2, radiusearthkm,
         mu, vkmpersec, delmtemp;
     int ktr;

     /* ------------------ set mathematical constants --------------- */
     // sgp4fix divisor for divide by zero check on inclination
     // the old check used 1.0 + cos(pi-1.0e-9), but then compared it to
     // 1.5 e-12, so the threshold was changed to 1.5e-12 for consistency
     const double temp4 =   1.5e-12;
     twopi = 2.0 * pi;
     x2o3  = 2.0 / 3.0;
     // sgp4fix identify constants and allow alternate values
     getgravconst( whichconst, tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2 );
     vkmpersec     = radiusearthkm * xke/60.0;

     /* --------------------- clear sgp4 error flag ----------------- */
     satrec.t     = tsince;
     satrec.error = 0;

     /* ------- update for secular gravity and atmospheric drag ----- */
     xmdf    = satrec.mo + satrec.mdot * satrec.t;
     argpdf  = satrec.argpo + satrec.argpdot * satrec.t;
     nodedf  = satrec.nodeo + satrec.nodedot * satrec.t;
     argpm   = argpdf;
     mm      = xmdf;
     t2      = satrec.t * satrec.t;
     nodem   = nodedf + satrec.nodecf * t2;
     tempa   = 1.0 - satrec.cc1 * satrec.t;
     tempe   = satrec.bstar * satrec.cc4 * satrec.t;
     templ   = satrec.t2cof * t2;

     if (satrec.isimp != 1)
       {
         delomg = satrec.omgcof * satrec.t;
         // sgp4fix use mutliply for speed instead of pow
         delmtemp =  1.0 + satrec.eta * cos(xmdf);
         delm   = satrec.xmcof *
                  (delmtemp * delmtemp * delmtemp -
                  satrec.delmo);
         temp   = delomg + delm;
         mm     = xmdf + temp;
         argpm  = argpdf - temp;
         t3     = t2 * satrec.t;
         t4     = t3 * satrec.t;
         tempa  = tempa - satrec.d2 * t2 - satrec.d3 * t3 -
                          satrec.d4 * t4;
         tempe  = tempe + satrec.bstar * satrec.cc5 * (sin(mm) -
                          satrec.sinmao);
         templ  = templ + satrec.t3cof * t3 + t4 * (satrec.t4cof +
                          satrec.t * satrec.t5cof);
       }
```

```
1729
1730        nm    = satrec.no;
1731        em    = satrec.ecco;
1732        inclm = satrec.inclo;
1733        if (satrec.method == 'd')
1734          {
1735            tc = satrec.t;
1736            dspace
1737                (
1738                  satrec.irez,
1739                  satrec.d2201, satrec.d2211, satrec.d3210,
1740                  satrec.d3222, satrec.d4410, satrec.d4422,
1741                  satrec.d5220, satrec.d5232, satrec.d5421,
1742                  satrec.d5433, satrec.dedt,  satrec.del1,
1743                  satrec.del2,  satrec.del3,  satrec.didt,
1744                  satrec.dmdt,  satrec.dnodt, satrec.domdt,
1745                  satrec.argpo, satrec.argpdot, satrec.t, tc,
1746                  satrec.gsto, satrec.xfact, satrec.xlamo,
1747                  satrec.no, satrec.atime,
1748                  em, argpm, inclm, satrec.xli, mm, satrec.xni,
1749                  nodem, dndt, nm
1750                );
1751          } // if method = d
1752
1753        if (nm <= 0.0)
1754          {
1755  //          printf("# error nm %f\n", nm);
1756            satrec.error = 2;
1757            // sgp4fix add return
1758            return false;
1759          }
1760        am = pow((xke / nm),x2o3) * tempa * tempa;
1761        nm = xke / pow(am, 1.5);
1762        em = em - tempe;
1763
1764        // fix tolerance for error recognition
1765        // sgp4fix am is fixed from the previous nm check
1766        if ((em >= 1.0) || (em < -0.001)/* || (am < 0.95)*/ )
1767          {
1768  //          printf("# error em %f\n", em);
1769            satrec.error = 1;
1770            // sgp4fix to return if there is an error in eccentricity
1771            return false;
1772          }
1773        // sgp4fix fix tolerance to avoid a divide by zero
1774        if (em < 1.0e-6)
1775            em  = 1.0e-6;
1776        mm     = mm + satrec.no * templ;
1777        xlm    = mm + argpm + nodem;
1778        emsq   = em * em;
1779        temp   = 1.0 - emsq;
1780
1781        nodem  = fmod(nodem, twopi);
1782        argpm  = fmod(argpm, twopi);
1783        xlm    = fmod(xlm, twopi);
1784        mm     = fmod(xlm - argpm - nodem, twopi);
1785
1786        /* ------------------ compute extra mean quantities ------------- */
1787        sinim = sin(inclm);
1788        cosim = cos(inclm);
1789
1790        /* -------------------- add lunar-solar periodics -------------- */
1791        ep     = em;
1792        xincp  = inclm;
1793        argpp  = argpm;
1794        nodep  = nodem;
1795        mp     = mm;
1796        sinip  = sinim;
1797        cosip  = cosim;
1798        if (satrec.method == 'd')
1799          {
1800            dpper
1801                (
```

```
1802                  satrec.e3,   satrec.ee2,  satrec.peo,
1803                  satrec.pgho, satrec.pho,  satrec.pinco,
1804                  satrec.plo,  satrec.se2,  satrec.se3,
1805                  satrec.sgh2, satrec.sgh3, satrec.sgh4,
1806                  satrec.sh2,  satrec.sh3,  satrec.si2,
1807                  satrec.si3,  satrec.sl2,  satrec.sl3,
1808                  satrec.sl4,  satrec.t,    satrec.xgh2,
1809                  satrec.xgh3, satrec.xgh4, satrec.xh2,
1810                  satrec.xh3,  satrec.xi2,  satrec.xi3,
1811                  satrec.xl2,  satrec.xl3,  satrec.xl4,
1812                  satrec.zmol, satrec.zmos, satrec.inclo,
1813                  'n', ep, xincp, nodep, argpp, mp, satrec.operationmode
1814                );
1815            if (xincp < 0.0)
1816              {
1817                xincp  = -xincp;
1818                nodep = nodep + pi;
1819                argpp  = argpp - pi;
1820              }
1821            if ((ep < 0.0 ) || ( ep > 1.0))
1822              {
1823  //              printf("# error ep %f\n", ep);
1824                satrec.error = 3;
1825                // sgp4fix add return
1826                return false;
1827              }
1828          } // if method = d
1829
1830        /* -------------------- long period periodics ------------------ */
1831        if (satrec.method == 'd')
1832          {
1833            sinip =  sin(xincp);
1834            cosip =  cos(xincp);
1835            satrec.aycof = -0.5*j3oj2*sinip;
1836            // sgp4fix for divide by zero for xincp = 180 deg
1837            if (fabs(cosip+1.0) > 1.5e-12)
1838                satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) / (1.0 +
                    cosip);
1839              else
1840                satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) / temp4;
1841          }
1842        axnl = ep * cos(argpp);
1843        temp = 1.0 / (am * (1.0 - ep * ep));
1844        aynl = ep* sin(argpp) + temp * satrec.aycof;
1845        xl   = mp + argpp + nodep + temp * satrec.xlcof * axnl;
1846
1847        /* --------------------- solve kepler's equation --------------- */
1848        u    = fmod(xl - nodep, twopi);
1849        eo1  = u;
1850        tem5 = 9999.9;
1851        ktr = 1;
1852        //   sgp4fix for kepler iteration
1853        //   the following iteration needs better limits on corrections
1854        while (( fabs(tem5) >= 1.0e-12) && (ktr <= 10) )
1855          {
1856            sineo1 = sin(eo1);
1857            coseo1 = cos(eo1);
1858            tem5   = 1.0 - coseo1 * axnl - sineo1 * aynl;
1859            tem5   = (u - aynl * coseo1 + axnl * sineo1 - eo1) / tem5;
1860            if(fabs(tem5) >= 0.95)
1861                tem5 = tem5 > 0.0 ? 0.95 : -0.95;
1862            eo1    = eo1 + tem5;
1863            ktr = ktr + 1;
1864          }
1865
1866        /* ------------- short period preliminary quantities ----------- */
1867        ecose = axnl*coseo1 + aynl*sineo1;
1868        esine = axnl*sineo1 - aynl*coseo1;
1869        el2   = axnl*axnl + aynl*aynl;
1870        pl    = am*(1.0-el2);
1871        if (pl < 0.0)
1872          {
1873  //          printf("# error pl %f\n", pl);
```

```
1874            satrec.error = 4;
1875            // sgp4fix add return
1876            return false;
1877          }
1878          else
1879          {
1880            rl     = am * (1.0 - ecose);
1881            rdotl  = sqrt(am) * esine/rl;
1882            rvdotl = sqrt(pl) / rl;
1883            betal  = sqrt(1.0 - el2);
1884            temp   = esine / (1.0 + betal);
1885            sinu   = am / rl * (sineo1 - aynl - axnl * temp);
1886            cosu   = am / rl * (coseo1 - axnl + aynl * temp);
1887            su     = atan2(sinu, cosu);
1888            sin2u  = (cosu + cosu) * sinu;
1889            cos2u  = 1.0 - 2.0 * sinu * sinu;
1890            temp   = 1.0 / pl;
1891            temp1  = 0.5 * j2 * temp;
1892            temp2  = temp1 * temp;
1893
1894            /* -------------- update for short period periodics ------------ */
1895            if (satrec.method == 'd')
1896              {
1897                cosisq                 = cosip * cosip;
1898                satrec.con41  = 3.0*cosisq - 1.0;
1899                satrec.x1mth2 = 1.0 - cosisq;
1900                satrec.x7thm1 = 7.0*cosisq - 1.0;
1901              }
1902            mrt   = rl * (1.0 - 1.5 * temp2 * betal * satrec.con41) +
1903                    0.5 * temp1 * satrec.x1mth2 * cos2u;
1904            su    = su - 0.25 * temp2 * satrec.x7thm1 * sin2u;
1905            xnode = nodep + 1.5 * temp2 * cosip * sin2u;
1906            xinc  = xincp + 1.5 * temp2 * cosip * sinip * cos2u;
1907            mvt   = rdotl - nm * temp1 * satrec.x1mth2 * sin2u / xke;
1908            rvdot = rvdotl + nm * temp1 * (satrec.x1mth2 * cos2u +
1909                    1.5 * satrec.con41) / xke;
1910
1911            /* --------------------- orientation vectors ------------------- */
1912            sinsu =  sin(su);
1913            cossu =  cos(su);
1914            snod  =  sin(xnode);
1915            cnod  =  cos(xnode);
1916            sini  =  sin(xinc);
1917            cosi  =  cos(xinc);
1918            xmx   = -snod * cosi;
1919            xmy   =  cnod * cosi;
1920            ux    =  xmx * sinsu + cnod * cossu;
1921            uy    =  xmy * sinsu + snod * cossu;
1922            uz    =  sini * sinsu;
1923            vx    =  xmx * cossu - cnod * sinsu;
1924            vy    =  xmy * cossu - snod * sinsu;
1925            vz    =  sini * cossu;
1926
1927            /* --------- position and velocity (in km and km/sec) ---------- */
1928            r[0] = (mrt * ux)* radiusearthkm;
1929            r[1] = (mrt * uy)* radiusearthkm;
1930            r[2] = (mrt * uz)* radiusearthkm;
1931            v[0] = (mvt * ux + rvdot * vx) * vkmpersec;
1932            v[1] = (mvt * uy + rvdot * vy) * vkmpersec;
1933            v[2] = (mvt * uz + rvdot * vz) * vkmpersec;
1934          }  // if pl > 0
1935
1936        // sgp4fix for decaying satellites
1937        if (mrt < 1.0)
1938          {
1939  //          printf("# decay condition %11.6f \n",mrt);
1940            satrec.error = 6;
1941            return false;
1942          }
1943
1944  //#include "debug7.cpp"
1945        return true;
1946    }  // end sgp4
```

```
/* -----------------------------------------------------------------------------
*
*                           function gstime
*
*  this function finds the greenwich sidereal time.
*
*  author        : david vallado                  719-573-2600    1 mar 2001
*
*  inputs          description                    range / units
*    jdut1       - julian date in ut1             days from 4713 bc
*
*  outputs       :
*    gstime      - greenwich sidereal time        0 to 2pi rad
*
*  locals        :
*    temp        - temporary variable for doubles   rad
*    tut1        - julian centuries from the
*                  jan 1, 2000 12 h epoch (ut1)
*
*  coupling      :
*    none
*
*  references    :
*    vallado       2004, 191, eq 3-45
* --------------------------------------------------------------------------- */

double  gstime
        (
          double jdut1
        )
   {
     const double twopi = 2.0 * pi;
     const double deg2rad = pi / 180.0;
     double       temp, tut1;

     tut1 = (jdut1 - 2451545.0) / 36525.0;
     temp = -6.2e-6* tut1 * tut1 * tut1 + 0.093104 * tut1 * tut1 +
             (876600.0*3600 + 8640184.812866) * tut1 + 67310.54841;  // sec
     temp = fmod(temp * deg2rad / 240.0, twopi); //360/86400 = 1/240, to deg, to rad

     // ------------------------ check quadrants ---------------------
     if (temp < 0.0)
         temp += twopi;

     return temp;
   }  // end gstime


/* -----------------------------------------------------------------------------
*
*                           function getgravconst
*
*  this function gets constants for the propagator. note that mu is identified to
*    facilitiate comparisons with newer models. the common useage is wgs72.
*
*  author        : david vallado                  719-573-2600   21 jul 2006
*
*  inputs        :
*    whichconst  - which set of constants to use  wgs72old, wgs72, wgs84
*
*  outputs       :
*    tumin       - minutes in one time unit
*    mu          - earth gravitational parameter
*    radiusearthkm - radius of the earth in km
*    xke         - reciprocal of tumin
*    j2, j3, j4  - un-normalized zonal harmonic values
*    j3oj2       - j3 divided by j2
*
*  locals        :
*
*  coupling      :
*    none
*
```

```
2020   *  references    :
2021   *    norad spacetrack report #3
2022   *    vallado, crawford, hujsak, kelso  2006
2023     ---------------------------------------------------------------------- */
2024
2025   void getgravconst
2026        (
2027         gravconsttype whichconst,
2028         double& tumin,
2029         double& mu,
2030         double& radiusearthkm,
2031         double& xke,
2032         double& j2,
2033         double& j3,
2034         double& j4,
2035         double& j3oj2
2036        )
2037        {
2038
2039          switch (whichconst)
2040            {
2041              // -- wgs-72 low precision str#3 constants --
2042              case wgs72old:
2043              mu     = 398600.79964;        // in km3 / s2
2044              radiusearthkm = 6378.135;     // km
2045              xke    = 0.0743669161;
2046              tumin  = 1.0 / xke;
2047              j2     =   0.001082616;
2048              j3     =  -0.00000253881;
2049              j4     =  -0.00000165597;
2050              j3oj2  =  j3 / j2;
2051            break;
2052              // ------------ wgs-72 constants ------------
2053              case wgs72:
2054              mu     = 398600.8;            // in km3 / s2
2055              radiusearthkm = 6378.135;     // km
2056              xke    = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
2057              tumin  = 1.0 / xke;
2058              j2     =   0.001082616;
2059              j3     =  -0.00000253881;
2060              j4     =  -0.00000165597;
2061              j3oj2  =  j3 / j2;
2062            break;
2063              case wgs84:
2064              // ------------ wgs-84 constants ------------
2065              mu     = 398600.5;            // in km3 / s2
2066              radiusearthkm = 6378.137;     // km
2067              xke    = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
2068              tumin  = 1.0 / xke;
2069              j2     =   0.00108262998905;
2070              j3     =  -0.00000253215306;
2071              j4     =  -0.00000161098761;
2072              j3oj2  =  j3 / j2;
2073            break;
2074            default:
2075              fprintf(stderr,"unknown gravity option (%d)\n",whichconst);
2076            break;
2077            }
2078
2079        }   // end getgravconst
2080
2081
2082   /* -----------------------------------------------------------------------
2083   *
2084   *                           procedure days2mdhms
2085   *
2086   *  this procedure converts the day of the year, days, to the equivalent month
2087   *    day, hour, minute and second.
2088   *
2089   *  algorithm     : set up array for the number of days per month
2090   *                  find leap year - use 1900 because 2000 is a leap year
2091   *                  loop through a temp value while the value is < the days
2092   *                  perform int conversions to the correct day and month
```

```
2093   *                  convert remainder into h m s using type conversions
2094   *
2095   *  author        : david vallado                  719-573-2600    1 mar 2001
2096   *
2097   *  inputs          description                    range / units
2098   *    year        - year                           1900 .. 2100
2099   *    days        - julian day of the year         0.0  .. 366.0
2100   *
2101   *  outputs       :
2102   *    mon         - month                          1 .. 12
2103   *    day         - day                            1 .. 28,29,30,31
2104   *    hr          - hour                           0 .. 23
2105   *    min         - minute                         0 .. 59
2106   *    sec         - second                         0.0 .. 59.999
2107   *
2108   *  locals        :
2109   *    dayofyr     - day of year
2110   *    temp        - temporary extended values
2111   *    inttemp     - temporary int value
2112   *    i           - index
2113   *    lmonth[12]  - int array containing the number of days per month
2114   *
2115   *  coupling      :
2116   *    none.
2117   * ------------------------------------------------------------------------- */
2118
2119   void    days2mdhms
2120           (
2121             int year, long int days, double days_fraccion,
2122             int& mon, int& day, int& hr, int& minute, double& sec
2123           )
2124     {
2125       int i, inttemp, dayofyr;
2126       double    temp;
2127       int lmonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
2128
2129       //dayofyr = (int)floor(days);
2130       dayofyr = days;
2131       // ----------------- find month and day of month ----------------
2132       if ( (year % 4) == 0 )
2133         lmonth[1] = 29;
2134
2135       i = 1;
2136       inttemp = 0;
2137       while ((dayofyr > inttemp + lmonth[i-1]) && (i < 12))
2138       {
2139         inttemp = inttemp + lmonth[i-1];
2140         i++;
2141       }
2142       mon = i;
2143       day = dayofyr - inttemp;
2144
2145       // ----------------- find hours minutes and seconds -------------
2146       //temp = (days - dayofyr) * 24.0;
2147       temp = days_fraccion * 24.0;
2148       hr   = (int)floor(temp);
2149       temp = (temp - hr) * 60.0;
2150       minute  = (int)floor(temp);
2151       sec  = (temp - minute) * 60.0;
2152     }  // end days2mdhms
2153
2154
2155   /*-----------------------------------------------------------------------
2156   *
2157   *                              función tleArv
2158   *
2159   *  Esta función toma los datos de las cadenas de caracteres obtenidas de los
2160   *  datos TLE, los arregla y carga en la estructura datosSat, e inicializa las
2161   *  variables necesarias para aplicar el propagador de órbitas sgp4, que es el
2162   *  que finalmente devuelve los vectores de posición y velocidad del satélite
2163   *
2164   *  entradas:
2165   *    linea1 --> primera línea TLE
```

```
*    linea2 -->  segunda línea TLE
*    modoOp -->  modo operación (requerido para llamar a sgp4())
*                             Valores: 'a' (afspc estándar, el usado aquí)
*                                      'i' (improved)
*    wgs -->  conjunto de constantes:
*                             Valores: "wgs72old"
*                                      "wgs72"(recomendado para este software)
*                                      "wgs84"
*    mpe -->  minutos desde la época(dato TLE del instante en que se tomaron
*                             los últimos datos fiables)
*
*  salidas:
*    datosSat --> estructura con todos los datos del satélite necesarios para
*                             aplicar el propagador de órbitas sgp4
*
*  llamadas a otras funciones:
*    getgravconst--> proporciona los elementos orbitales en función del juego de
                              constantes (wgs) seleccionado
*    days2mdhms  --> convierte de días a meses, días, horas, minutos y segundos
*    diajul -->  convierte de año, mes, día, hora, minuto, segundo a días
                              julianos
*    sgp4init --> inicializa todas las variables necesarias para aplicar el
                              propagador sgp4
*-----------------------------------------------------------------------------*/
void tleArv
     (
      char linea1[70], char linea2[70],
      char modoOp,
      gravconsttype wgs,
      elsetrec& datosSat
     )
     {
       const double deg2rad  =   pi / 180.0;            // 0.0174532925199433
       const double xpdotp   =  1440.0 / (2.0 *pi);     // 229.1831180523293

       double seg,                  // segundos
            radioTierra,            // radio de la Tierra
            mu,                     // parámetro gravitacional de la Tierra =
                                    // G * M
                                    // (G constante gravitatoria, M masa
                                    // de la Tierra)
            tumin,                  // minutos en una unidad de tiempo
            xke,                    // 1/tumin. Se calcula xke en getgravconst()
                                    // a partir de mu y del rado de la ,
                                    // Tierra y tumin a partir de xke
                                    // (1/xke)
            j2, j3, j4,             // valores armónicos de zona, no
                                    // normalizados
            j3oj2;                  // j3/j2
      // double startsec, stopsec, startdayofyr, stopdayofyr, jdstart, jdstop;
       //int startyear, stopyear, startmon, stopmon, startday, stopday,
       //    starthr, stophr, startmin, stopmin;
       int nlin, num, j;
       long numrev = 0,             // número de órbitas completadas en el
                                          // momento de actualización de
                                          // datos (época)
            numel = 0;              // número de actualizaciones de datos TLE
                                          // generados por USSPACECOM
       char clasificacion,          // clasificación del satélite
            desigInternac[11];      // designación interncional
       int ano = 0;                 // año
       int mes, dia, hr, minu,      // mes,día, horas, minutos
            nexp, ibexp;
       char buffer[15];
       char *p;
       long int epoca;
       double epoca_frac;

       datosSat.error = 0;

       // Depuramos los datos TLE
       for (j = 10; j <= 15; j++)
           if (linea1[j] == ' ')
```

```
2239                linea1[j] = '_';
2240
2241            if (linea1[44] != ' ')
2242                linea1[43] = linea1[44];
2243            linea1[44] = '.';
2244            if (linea1[7] == ' ')
2245                linea1[7] = 'U';
2246            if (linea1[9] == ' ')
2247                linea1[9] = '.';
2248            for (j = 45; j <= 49; j++)
2249                if (linea1[j] == ' ')
2250                    linea1[j] = '0';
2251            if (linea1[51] == ' ')
2252                linea1[51] = '0';
2253            if (linea1[53] != ' ')
2254                linea1[52] = linea1[53];
2255            linea1[53] = '.';
2256            linea2[25] = '.';
2257            for (j = 26; j <= 32; j++)
2258                if (linea2[j] == ' ')
2259                    linea2[j] = '0';
2260            if (linea1[62] == ' ')
2261                linea1[62] = '0';
2262            if (linea1[68] == ' ')
2263                linea1[68] = '0';
2264             /*
2265              * nlin              - número de línea
2266              * satnum            - número de satélite
2267              * clasificacion     - clasificación del satélite
2268              * desigInternac     - desigación internacional
2269              * epochyr           - año de la época
2270              * epochdays         - días de la época
2271              * ndot              - coeficiente balístico (derivada del movimiento
2272              *                         medio)
2273              * nddot             - segunda derivada del movimiento medio
2274              *                         (generalmente 0)
2275              * nexp              - potencia de 10 aplicada al coeficiente anterior
2276              * bstar             - término de arrastre o coeficiene de presión de
2277              *                         radiación
2278              * ibexp             - potencia de 10 aplicada al coeficiente anterior
2279              * num               - tipo de efemérides
2280              * numel             - número de actualizaciones de datos TLE generados
2281              *                         por USSPACECOM
2282              *
2283              * inclo             - inclinación de la órbita
2284              * nodeo             - ascensión recta del nodo ascendente
2285              * ecco              - excetricidad de la órbita
2286              * argpo             - argumento del perigeo (ángulo entre el nodo
2287              *                         ascendentey el perigeo)
2288              * mo                - anomalía media (ángulo con respecto al semieje
2289              *                         mayor de la posición del satélite si su
2290              *                         órbita fuera una circunferencia con radio =
2291              *                         semieje mayor)
2292              * no                - movimiento medio, número medio de órbitas por día
2293              * numrev            - número de órbitas desde la época
2294              *
2295              */
2296
2297            //0
2298            p = linea1;
2299            strncpy(buffer, p, 2);
2300            buffer[2] = '\0';
2301            p += 2;
2302            nlin = atoi(buffer);
2303            //2
2304            strncpy(buffer, p, 5);
2305            buffer[5] = '\0';
2306            p += 5;
2307            datosSat.satnum = atol(buffer);
2308            //7
2309            clasificacion = *p;
2310            p ++;
2311            //8
```

```
2312          strncpy(desigInternac, p, 10);
2313          buffer[10] = '\0';
2314          p += 10;
2315          //18
2316          strncpy(buffer, p, 2);
2317          buffer[2] = '\0';
2318          p += 2;
2319          datosSat.epochyr = atoi(buffer);
2320          //20
2321          /*
2322          strncpy(buffer, p, 12);
2323          buffer[12] = '\0';
2324          p += 12;
2325          datosSat.epochdays = atof(buffer);
2326          */
2327          strncpy(buffer, p, 3);
2328          buffer[3] = '\0';
2329          p += 3;
2330          datosSat.epochdays = atoi(buffer);
2331          //23
2332          strncpy(buffer, p, 9);
2333          buffer[9] = '\0';
2334          p += 9;
2335          datosSat.epochdays_fraccion = atof(buffer);
2336
2337          //32
2338          strncpy(buffer, p, 11);
2339          buffer[11] = '\0';
2340          p += 11;
2341          datosSat.ndot = atof(buffer);
2342          //43
2343          strncpy(buffer, p, 7);
2344          buffer[7] = '\0';
2345          p += 7;
2346          datosSat.nddot = atof(buffer);
2347          //50
2348          strncpy(buffer, p, 2);
2349          buffer[2] = '\0';
2350          p += 2;
2351          nexp = atoi(buffer);
2352          //52
2353          strncpy(buffer, p, 7);
2354          buffer[7] = '\0';
2355          p += 7;
2356          datosSat.bstar = atof(buffer);
2357          //59
2358          strncpy(buffer, p, 2);
2359          buffer[2] = '\0';
2360          p += 2;
2361          ibexp = atoi(buffer);
2362          //61
2363          strncpy(buffer, p, 2);
2364          buffer[2] = '\0';
2365          p += 2;
2366          num = atoi(buffer);
2367          //63
2368          strncpy(buffer, p, 6);
2369          buffer[6] = '\0';
2370          nlin = atol(buffer);
2371
2372      /*  no funciona en arduino
2373          sscanf(linea1,
2374              "%2d %5ld %1c %10s %2d %12lf %11lf %7lf %2d %7lf %2d %2d %6ld ",
2375                  &nlin,&datosSat.satnum,&clasificacion, desigInternac,
2376                  &datosSat.epochyr, &datosSat.epochdays,&datosSat.ndot,
2377                  &datosSat.nddot, &nexp, &datosSat.bstar, &ibexp, &num, &numel );
2378      */
2379
2380           {
2381             if (linea2[52] == ' ')
2382             {
2383                 //0
2384                 p = linea2;
```

```
2385                    strncpy(buffer, p, 2);
2386                    buffer[2] = '\0';
2387                    p += 2;
2388                    nlin = atoi(buffer);
2389                    //2
2390                    strncpy(buffer, p, 5);
2391                    buffer[5] = '\0';
2392                    p += 5;
2393                    datosSat.satnum = atol(buffer);
2394                    //7
2395                    strncpy(buffer, p, 9);
2396                    buffer[9] = '\0';
2397                    p += 9;
2398                    datosSat.inclo = atof(buffer);
2399                    //16
2400                    strncpy(buffer, p, 9);
2401                    buffer[9] = '\0';
2402                    p += 9;
2403                    datosSat.nodeo = atof(buffer);
2404                    //25
2405                    strncpy(buffer, p, 8);
2406                    buffer[8] = '\0';
2407                    p += 8;
2408                    datosSat.ecco = atof(buffer);
2409                    //33
2410                    strncpy(buffer, p, 9);
2411                    buffer[9] = '\0';
2412                    p += 9;
2413                    datosSat.argpo = atof(buffer);
2414                    //42
2415                    strncpy(buffer, p, 9);
2416                    buffer[9] = '\0';
2417                    p += 9;
2418                    datosSat.mo = atof(buffer);
2419                    //51
2420                    strncpy(buffer, p, 10);
2421                    buffer[10] = '\0';
2422                    p += 10;
2423                    datosSat.no = atof(buffer);
2424                    //61
2425                    strncpy(buffer, p, 6);
2426                    buffer[6] = '\0';
2427                    numrev = atof(buffer);
2428                     /*
2429                    sscanf(linea2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %10lf %6ld \n",
2430                            &nlin,&datosSat.satnum, &datosSat.inclo,
2431                            &datosSat.nodeo,&datosSat.ecco, &datosSat.argpo,
2432                            &datosSat.mo, &datosSat.no, &numrev );
2433                            */
2434                  }
2435                  else
2436                  {
2437                    //0
2438                    p = linea2;
2439                    strncpy(buffer, p, 2);
2440                    buffer[2] = '\0';
2441                    p += 2;
2442                    nlin = atoi(buffer);
2443                    //2
2444                    strncpy(buffer, p, 5);
2445                    buffer[5] = '\0';
2446                    p += 5;
2447                    datosSat.satnum = atol(buffer);
2448                    //7
2449                    strncpy(buffer, p, 9);
2450                    buffer[9] = '\0';
2451                    p += 9;
2452                    datosSat.inclo = atof(buffer);
2453                    //16
2454                    strncpy(buffer, p, 9);
2455                    buffer[9] = '\0';
2456                    p += 9;
2457                    datosSat.nodeo = atof(buffer);
```

```
                //25
                strncpy(buffer, p, 8);
                buffer[8] = '\0';
                p += 8;
                datosSat.ecco = atof(buffer);
                //33
                strncpy(buffer, p, 9);
                buffer[9] = '\0';
                p += 9;
                datosSat.argpo = atof(buffer);
                //42
                strncpy(buffer, p, 9);
                buffer[9] = '\0';
                p += 9;
                datosSat.mo = atof(buffer);
                //51
                strncpy(buffer, p, 11);
                buffer[11] = '\0';
                p += 11;
                datosSat.no = atof(buffer);
                //62
                strncpy(buffer, p, 6);
                buffer[6] = '\0';
                numrev = atof(buffer);
                 /*   no funciona en arduino
                sscanf(linea2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %11lf %6ld \n",
                       &nlin,&datosSat.satnum, &datosSat.inclo,
                       &datosSat.nodeo,&datosSat.ecco, &datosSat.argpo,
                       &datosSat.mo, &datosSat.no, &numrev );
                       */
            }
          }

        // ---- adecuamos no, ndot, nddot ----
        datosSat.no   = datosSat.no / xpdotp;
                                      // pasamos mov medio a rad/min
        datosSat.nddot= datosSat.nddot * pow(10.0, nexp);
                                      // aplicamos pot de 10 a 2ª derivada
        datosSat.bstar= datosSat.bstar * pow(10.0, ibexp);
                                      // aplicamos pot de 10 al término de
                                      // arrastre

        // ---- convertimos a unidades empleadas en sgp4 ----
        datosSat.ndot = datosSat.ndot  / (xpdotp*1440.0);
        datosSat.nddot= datosSat.nddot / (xpdotp*1440.0*1440);

        // ---- elementos orbitales keplerianos ----
        datosSat.a    = pow( datosSat.no * tumin_, (-2.0/3.0) ); // semieje mayor
        datosSat.inclo = datosSat.inclo  * deg2rad;  // inclinación de la órbita
        datosSat.nodeo = datosSat.nodeo  * deg2rad;  // longitud nodo ascendente
        datosSat.argpo = datosSat.argpo  * deg2rad;  // argumento del perigeo
        datosSat.mo    = datosSat.mo     * deg2rad;  // anomalía media
                                          // al último elemento orbital
                                          // (excentricidad) no hay que
                                          // hacerle nada

        // ---- distancias del apogeo y perigeo al centro de la Tierra ----
        datosSat.alta = datosSat.a * (1.0 + datosSat.ecco) - 1.0;
        datosSat.altp = datosSat.a * (1.0 - datosSat.ecco) - 1.0;


        // ---------------------------------------------------------------
        // ---------- Determinación de la época de los datos TLE ----------
        // ----------       en formato de fecha juliana          ----------
        // ---------------------------------------------------------------
        // Hay que tener en cuenta que este software utiliza la versión
        // de sgp4 que toma como referencia el 01/01/1950 a las 00:00 horas, por
        // lo que hay que restar esa fecha juliana en las llamadas a las
        // funciones del propagador sgp4
        // ---------------------------------------------------------------

        // ----- ponemos el año de la época en 4 cifras (entre 1957 y 2056) -----
        if (datosSat.epochyr < 57)
```

```
2531            ano = datosSat.epochyr + 2000;
2532          else
2533            ano = datosSat.epochyr + 1900;
2534
2535        // ---- determinamos la época TLE en fecha juliana y la colocamos en
2536        // datosSat en jdsatepoch (dias) y jdsatepoch_fraccion (fracción)
2537        days2mdhms (ano, datosSat.epochdays, datosSat.epochdays_fraccion,
2538        mes, dia, hr, minu, seg );
2539        diajul(ano, mes, dia, hr, minu, seg, datosSat.jdsatepoch,
            datosSat.jdsatepoch_fraccion);
2540
2541        // ----inicialización de los datos de la órbita para la época TLE -------
2542        // hay que restar la fecha juliana de 01/01/1950 (2433281.5) para la
2543        // que está programado este propagador de órbitas sgp4 a la época TLE
2544        restaxSep(datosSat.jdsatepoch, 2433281, epoca,
2545                  datosSat.jdsatepoch_fraccion, 0.5, epoca_frac);
2546        sgp4init(wgs, modoOp, datosSat.satnum, epoca, epoca_frac,
2547                  datosSat.bstar, datosSat.ecco,
2548                  datosSat.argpo, datosSat.inclo, datosSat.mo,
2549                  datosSat.no, datosSat.nodeo, datosSat);
2550
2551      } // fin tleArv
2552
2553
2554  /*-----------------------------------------------------------------------------
2555  *
2556  *                              función theta
2557  *
2558  * Esta función determina la hora sideral del momento de la observación, y con
2559  * ella, la posición del meridiano de Greenwich (hacia el Este) con respecto al
2560  * equinoccio vernal (ángulo thetaG).
2561  * Se emplea la fórmula indicada en "Explanatory Supplement to the Astronomical
2562  * Almanac" (página 50), que toma como tiempo de referencia el 01/01/200 a las
2563  * 12:00.
2564  *
2565  *  constantes empleadas:
2566  *   fecha juliana de 01/01/2000 12:00              -->     2451515.0 días
2567  *   omegaT_ (rotación de la Tierra por día sideral,
2568  *            definida en fic. cabecera)            -->     1.00273790934
2569  *   segxDia_ (seg/día, definida en fic. cabecera)  -->     86400
2570  *
2571  *  entradas:
2572  *   lon             --> longitud local en radianes
2573  *   djut1           --> parte entera del instante de observación en formato de
2574  *                       día juliano, referido a la referencia horaria UT1
                            (también válida la UTC)
2575  *   djut1_fraccion  --> parte decimal del instante de observación en formato
2576  *                       de día juliano, referido a la referencia horaria UT1
2577  *
2578  *  salidas:
2579  *   GMST    --> ángulo entre la posición de observación y el equinoccio vernal
2580                   para el momento de observación (en radianes)
2581  *
2582  *  llamadas a otras funciones:
2583  *
2584  *-----------------------------------------------------------------------------*/
2585  double theta(double lon, long int djut1, double djut1_fraccion)
2586  {
2587      double          UT,             // Universal Time,
2588                      Tu,             // Tiempo desde 01/01/2000
2589                      GMST;           // Greenwich Mean Sidereal Time
2590
2591      UT = djut1_fraccion + 0.5;
2592      if(UT >= 1.0)
2593      {
2594          UT -= 1.0;
2595          djut1 += 1;
2596      }
2597
2598      // se pasa a siglos el número de días desde la referencia 01/01/2000
2599      Tu = (djut1 - 2451545.0) / 36525.0;
2600      // se aplica la fórmula de cálculo de GMST
2601      GMST = 24110.54841 + Tu *
```

```
                    (8640184.812866 + Tu * (0.093104 - Tu * 6.2e-06));

    // cálculo thetaG.
    GMST = fmod((GMST + segxDia_ * omegaT_ * UT), segxDia_);

    if (GMST < 0.0)
    {
        GMST += segxDia_;  // sumamos 1 día en caso de que dé negativo
    }

    GMST = (2.0 * pi * GMST / segxDia_);        // conversión a radianes
    // cálculo theta, para la posición de observación
    GMST = fmod(GMST + lon, (2.0 * pi));
    return GMST;
} // fin theta()

/*-------------------------------------------------------------------------
*
*                              función geoide
*
* Esta función aplica a la latitud del lugar de observación la corrección por
* achatamiento de la Tierra, según la fórmula indicada en "Explanatory
* Supplement to the Astronomical Almanac" (página 126)
*
*  constantes empleadas:
*   achat_ (definida en fich cabecera)      -->     1.0/298.26 (valor WGS72)
*   radioTierra_ (definida en fich cab)     -->     6378.135 (valor WGS72)
*   segxDia_ (seg/día, definida en fich cab)-->     86400
*   h_ (altitud, definida en fich cab)      -->     0
*   pi  (definida en fich cab)              -->     3.14159265358979323846
*
*  entradas:
*   lat             --> latitud (ojo, en radianes)
*   theta           --> ángulo entre la posición de observación y el equinoccio
*                       vernal para el momento de observación (en radianes)
*   h               --> altitud del lugar de observación (en Km) (suponemos 0)
*
*  salidas:
*   rLoc            --> vector de posición del lugar de observación
*
*  llamadas a otras funciones:
*
*------------------------------------------------------------------------*/
void geoide(double lat, double theta, /*double h,*/ double rLoc[3])
{
    double  c, s;

    c = 1.0 / sqrt(1.0 + achat_ * pow((sin(lat)), 2.0) * (achat_ - 2.0));
    s = pow((1.0 - achat_), 2.0) * c;

    rLoc[0] = ((radioTierra_ * c + h_) * cos(lat)) * cos(theta);    // rx (Km)
    rLoc[1] = ((radioTierra_ * c + h_) * cos(lat)) * sin(theta);    // ry (Km)
    rLoc[2] = ( radioTierra_ * s + h_) * sin(lat);                  // rz (Km)
} // fin geoide()

/*-------------------------------------------------------------------------
*
*                           función gradArad
*
* Cambia de grados, minutos, segundos a radianes
*
*  constantes empleadas:
*   pi  (definida en fich cab)              -->     3.14159265358979323846
*
*  entradas:
*   grad        --> grados sexagesimales de arco
*   min         --> minutos de arco
*   seg         --> segundos de arco
*
*  salidas:
*               --> valor del ángulo en radianes
*
*  llamadas a otras funciones:
```

```
2675   *
2676   *-------------------------------------------------------------------------*/
2677
2678   double gradArad(int grad, float minu)
2679   {
2680       return ((grad + minu / 60.0) /(180.0 / pi));
2681   }
2682
2683
2684   /*-------------------------------------------------------------------------
2685   *
2686   *                           función diajul
2687   *
2688   * Sustituye a jday para adaptar a los cálculos en punto foltante de
2689   * Arduino.Calcula la fecha juliana, pero en dias y fracción de días por separado
2690   * La fracción de días la da en días
2691   *
2692   *  constantes empleadas:
2693   *   pi  (definida en fich cab)              -->     3.14159265358979323846
2694   *
2695   *  entradas:
2696   *               -->
2697   *               -->
2698   *               -->
2699   *
2700   *  salidas:
2701   *               -->
2702   *  llamadas a otras funciones:
2703   *
2704   *-------------------------------------------------------------------------*/
2705   void    diajul
2706           (
2707             int ano, int mes, int dia, int hr, int minutos, double seg,
2708             long int& dj, double& fraccion
2709           )
2710      {
2711       float dj_ent;
2712       dj_ent = 367.0 * ano -
2713           floor((7 * (ano + floor((mes + 9) / 12.0))) * 0.25) +
2714           floor( 275 * mes / 9.0 ) +
2715           dia + 1721013.5 ;
2716       fraccion = fmod(dj_ent, 1.0);
2717       dj = dj_ent - fraccion;
2718       fraccion = fraccion +
2719           (hr + (minutos + (seg / 60))/60) /24;
2720       if(fraccion >= 1)
2721       {
2722           dj += 1;
2723           fraccion = (fraccion - 1);
2724       }
2725      }  // fin diajul
2726
2727   /*-------------------------------------------------------------------------
2728   *
2729   *                           función restaxSep
2730   *
2731   * Hace la resta de dos números con decimales, expresados con su parte entera
2732   * y su parte decimal por separado
2733   *
2734   *
2735   *  entradas:
2736   *       e1      --> parte entera del minuendo
2737   *       e2      --> parte entera del sustraendo
2738   *       d1      --> parte decimal del minuendo
2739   *       d2      --> parte decimal del sustraendo
2740   *
2741   *  salidas:
2742   *       eDif    --> parte entera del resultado
2743   *       dDif    --> parte decimal del resultado
2744   *
2745   *  llamadas a otras funciones:
2746   *
2747   *-------------------------------------------------------------------------*/
```

```
2748  void restaxSep(long int e1, long int e2, long int& eDif,
2749              double d1, double d2, double& dDif)
2750  {
2751      int llevamos = 0;
2752
2753      if(e1>e2)
2754      {
2755          dDif = d1 - d2;
2756          if(dDif < 0)
2757          {
2758              dDif += 1;
2759                  llevamos = 1;
2760          }
2761          eDif = e1 - (e2 + llevamos);
2762      }
2763      else if (e1 == e2)
2764      {
2765          eDif = 0;
2766          dDif = d1 - d2;
2767      }
2768      else         // e1<e2
2769      {
2770          dDif = d2 - d1;
2771          if(dDif < 0)
2772          {
2773              dDif += 1;
2774              llevamos = 1;
2775          }
2776          eDif = (e2 - (e1 + llevamos)) * (-1);
2777          if(eDif == 0)
2778              if((d2 - d1) < 0)
2779              {
2780                  dDif *= (-1);
2781              }
2782      }
2783  } // fin restaxSep
2784
```

18.6. SGP4.h

```
#ifndef SGP4COMPLETO_H_INCLUDED
#define SGP4COMPLETO_H_INCLUDED


#include <math.h>
#include <stdio.h>
#include <string.h>

#define SGP4Version  "SGP4 Version 2011-12-30"

#define pi 3.14159265358979323846

////////////////////////////////////////////////////////////////////////////
// -------- directivas preprocesador propias
#define radioTierra_ 6378.135
#define tumin_ (1.0/0.0743669161)
#define achat_  (1.0/298.26)
#define segxDia_ 86400.0
#define omegaT_ 1.00273790934

#define constDef    wgs72       //valor recomendado en SpaceTrack Report 3
#define opDef       'a'
#define h_  0

// ------------------------ structure declarations ---------------------------
typedef enum
{
  wgs72old,
  wgs72,
  wgs84
} gravconsttype;

typedef struct elsetrec
{
  long int  satnum;
  int       epochyr, epochtynumrev;
  int       error;
  char      operationmode;
  char      init, method;

  /* Near Earth */
  int    isimp;
  double aycof  , con41  , cc1    , cc4      , cc5    , d2      , d3   , d4    ,
         delmo  , eta    , argpdot, omgcof   , sinmao , t       , t2cof, t3cof ,
         t4cof  , t5cof  , x1mth2 , x7thm1   , mdot   , nodedot, xlcof , xmcof ,
         nodecf;

  /* Deep Space */
  int    irez;
  long int jdsatepoch, epochdays;       // partes enteras de esas variables
  // jdsatepoch pasa a ser el número de días de la época TLE y
  // jdsatepoch_fraccion la fracción de día
  double d2201  , d2211  , d3210  , d3222    , d4410  , d4422   , d5220 , d5232,
         d5421  , d5433  , dedt   , del1     , del2   , del3    , didt  , dmdt ,
         dnodt  , domdt  , e3     , ee2      , peo    , pgho    , pho   , pinco,
         plo    , se2    , se3    , sgh2     , sgh3   , sgh4    , sh2   , sh3  ,
         si2    , si3    , sl2    , sl3      , sl4    , gsto    , xfact , xgh2 ,
         xgh3   , xgh4   , xh2    , xh3      , xi2    , xi3     , xl2   , xl3  ,
         xl4    , xlamo  , zmol   , zmos     , atime  , xli     , xni;

  double a      , altp   , alta   ,
         /*epochdays*/ epochdays_fraccion,      // parte decimal
         /*jdsatepoch*/ jdsatepoch_fraccion,    // parte decimal
         nddot,   ndot ,   bstar  , rcse   , inclo  , nodeo    , ecco ,
         argpo,   mo   ,   no;
} elsetrec;


// ------------------------ function declarations ---------------------------
bool sgp4init
     (
       gravconsttype whichconst, char opsmode, const int satn,
       const long int epoch, const double epoch_fraccion,
```

```
        const double xbstar,  const double xecco,
        const double xargpo, const double xinclo,  const double xmo,
        const double xno, const double xnodeo,  elsetrec& satrec
      );

bool sgp4
      (
        gravconsttype whichconst, elsetrec& satrec,  double tsince,
        double r[3],  double v[3]
      );

double  gstime
        (
          double jdut1
        );

void getgravconst
      (
       gravconsttype whichconst,
       double& tumin,
       double& mu,
       double& radiusearthkm,
       double& xke,
       double& j2,
       double& j3,
       double& j4,
       double& j3oj2
      );

void    days2mdhms
        (
          int year, long int days, double days_fraccion,
          int& mon, int& day, int& hr, int& minute, double& sec
        );

void tleArv
      (
       char  longstr1[70], char longstr2[70],
       char  modoOp,
       gravconsttype wgs,
       elsetrec& datosSat
      );

double theta
    (double lon, long int djut1, double djut1_fraccion);

void geoide
    (double lat, double theta, /*double h,*/ double rLoc[3]/*,
        double vLoc[3]*/);

double gradArad
    (int grad, float minu);

void    diajul
        (
          int ano, int mes, int dia, int hr, int minutos, double seg,
          long int& dj, double& fraccion);

void restaxSep(long int e1, long int e2, long int& eDif,
                double d1, double d2, double& dDif);

#endif // SGP4COMPLETO_H_INCLUDED

```

19 MANUAL DE USUARIO

19.1. Setup

Al conectar el sistema, se muestra el siguiente texto por pantalla durante tres segundos:

Seguimiento de satélites
UDC

Si en este intervalo de tiempo se presionan simultáneamente las botones de izquierda y derecha, se accede al setup del sistema, en el que se ofrecen las siguientes opciones de configuración:

- Cálculo de ángulos de apuntamiento por software externo
- Cálculos autónomo de ángulos de apuntamiento en base a parámetros keplerianos
- Actualización de datos TLE
- Realimentación mediante encoder
- Realimentación mediante potenciómetro

Para cambiar entre una opción y otra se utilizan las botones de izquierda y derecha, seleccionando la opción deseada con la tecla central.

Para actualizar los datos TLE, el usuario deberá abrir el monitor serie del IDE Arduino y, posteriormente, acceder a la base de datos del NORAD [28] (cuyo enace se mostrará al usuario en el monitor serie) y obtener los datos TLE más actualizados, que se grabarán en la memoria EEPROM del microcontrolador enviándolos por puerto serie.

Al acceder a esta opción de configuración, se mostrará en el monitor serie lo indicado en la figura 19.1:

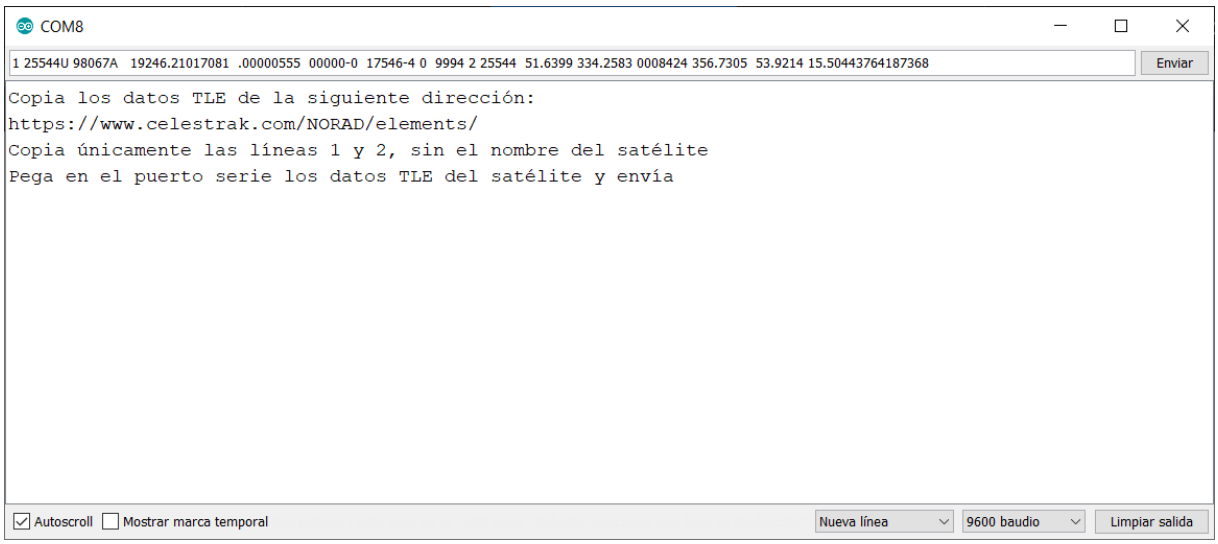


Figura 19.1 – Mensaje por puerto serie

Se deben enviar únicamente las dos líneas del TLE, sin el nombre del satélite, como se indica en la figura 19.2:

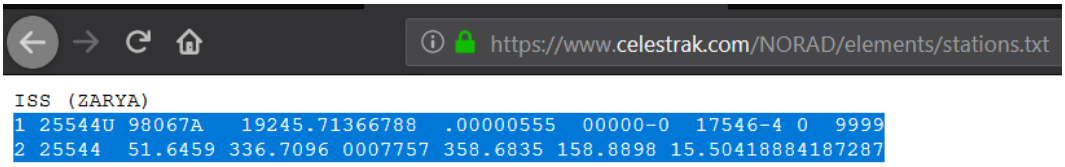


Figura 19.2 – Datos TLE actualizados de la web

Cuando se envían los datos, se verifica en el puerto serie que se han guardado de manera correcta en la EEPROM del microcontrolador.

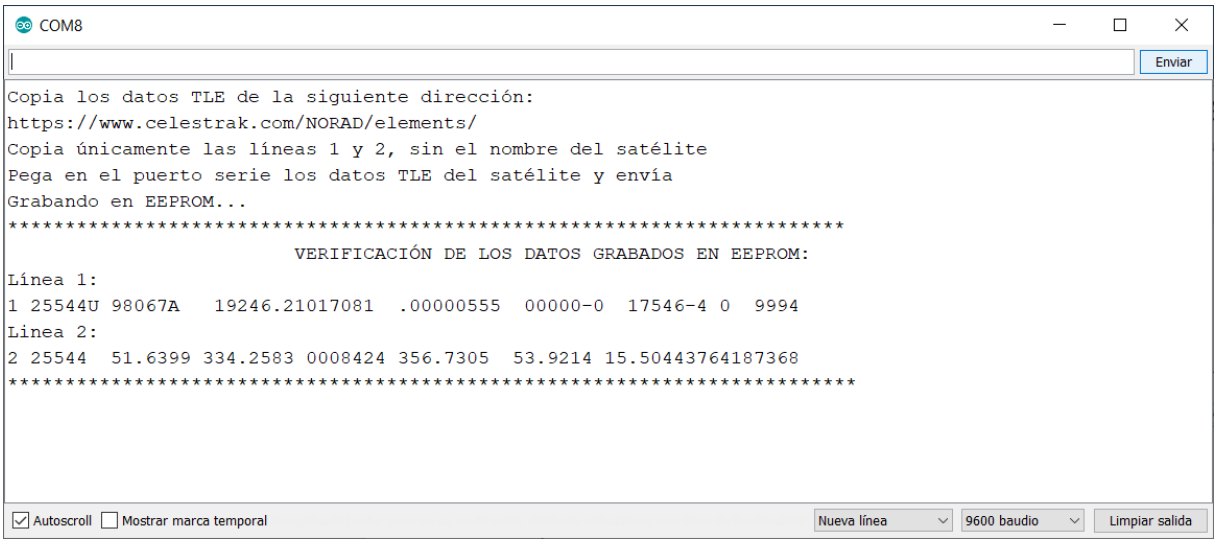


Figura 19.3 – Verificación de guardado

Cabe destacar que, en caso de abrir el monitor serie del IDE Arduino con el sistema en funcionamiento, la propia acción de abrir el monitor serie reseteará el sistema y será necesario acceder de nuevo a la opción de actualización de TLE (con el monitor serie ya abierto).

Realizada la configuración en el setup, se mantiene pulsada la tecla central hasta que se muestre por pantalla el siguiente mensaje:

Configuración realizada

19.2. Menú principal

Una vez realizada la configuración del sistema en el setup se accede al menú principal. En él se selecciona entre distintos modos de funcionamiento disponibles presionando la tecla central:

- Calibración: en este menú se realiza el ajuste de offset y de ganancia del circuito analógico, necesario para optimizar el funcionamiento del conversor A/D en caso de realizar la lectura de la posición a través de encoders.

Se ajusta en primer lugar el offset de azimut, moviendo el motor con la botonera hasta su posiciónde 0º, y ajustando con el potenciómetro de offset hasta que el indicador de tensión en la pantalla indique 0V.

Ajustado el offset, se presiona la tecla central para acceder al ajuste de ganancia, en el que se ajustará el potenciómetro de ganancia hasta los 5V, despues de haber colocado el motor en la posición de 360º, y se presiona de nuevo la tecla central para acceder a la calibración de elevación. Esta se realizará siguiendo el mismo procedimiento que en el caso de azimut.

En la pantalla LCD se visualizará lo siguiente:

Calibración
AZIMUT
Ajuste offset a 0V
Offset: 0.82

Tabla 19.1 – Pantalla LCD en calibración

- Manual: este modo de funcionamiento permite al usuario colocar las antenas en una posición concreta de dos maneras distintas:

  - Consigna: después de seleccionar qué motor se desea mover (azimut o elevación), se introduce a través del teclado el ángulo deseado con hasta dos decimales.

    Al pulsar la tecla aceptar del teclado, el motor correspondiente comenzará a moverse hasta la posición indicada, mostrando en la LCD la posición de consigna y la posición actual.

Modo manual			
Consigna:	170.00		
Azimut:	100.54	AZ:	35.00

Tabla 19.2 – Pantalla LCD en manual-consigna

    Para detener el motor y volver al menú anterior de selección del motor que se desea desplazar se pulsará la tecla central.

  - Manual: manteniendo pulsadas las teclas de izquierda, derecha, arriba y abajo se moverán los motores correspondientes en los sentidos que proceda. Este modo manual se utilizará únicamente para colocar los motores de manera aproximada.
  - En cualquiera de estos modos, para regresar al menú anterior, se mantendrá pulsada la tecla central. En el caso del menú *Consigna*, en el que establecemos la consigna por teclado y se realiza el desplazamiento de motor correspondiente, bastará con una pulsación, para poder frenar los motores rápidamente.
  - Uso del teclado:

- Tecla aceptar: #
- Punto decimal: *
- Borrar dígito: D
- Borrar todo: C

- Automático: este modo permite el seguimiento automático del satélite.

  En la pantalla LCD se muestra en todo momento la posición actual de los motores y la posición de consigna:

Modo automático			
CONSIGNA		POSICIÓN	
AZ:	100.54	AZ:	35.00
EL:	0.00	EL:	94.28

Tabla 19.3 – Pantalla LCD en modo automátio

  Para frenar los motores bastará con pulsar el botón central de la botonera, y se accederá de nuevo al menú principal.

  En caso de que exista un cambio de consigna que implique un cambio de sentido de giro de algún motor, se frenarán los motores para evitar cambios bruscos que dañen tanto el motor como la propia estructura del sistema.

  Si este cambio de sentido es provocado porque el motor ha excedido la posición de consigna, también se frenarán los motores antes de cambiar el sentido de giro.

19.3. Especificaciones técnicas

- **Alimentación**
  Alimentador externo de doble polaridad
  - **Positiva**: 11V a 16V, 1A
  - **Negativa**: -11V a -16V, 150mA
- **Accionamiento de motores**
  - 6 relés (contactos libres de potencial). Máximas prestaciones: 250V, 10A
  - 2 señales PWM (0 a 5V)
  - 2 señales de tensión (0 a 5V)
- **Realimentación de motores**
  - Tensión analógica
  - Encoder TBN50-SA2048RC2SN14 (CANopen)
- **Controles**

  - 5 botones
  - Teclado matricial 4x4

- **Indicadores**
  - Diodos LED
  - Pantalla LCD 20x04

- **Recepción de datos GPS**
  - En interior requiere antena de 26dB de ganancia

- **Conexiones**
  - Alimentación: conectores de apriete por tornillo
  - Salidas: conectores de apriete por tornillo
  - Realimentación analógica: conectores de apriete por tornillo
  - Realimentación CANopen: conector SUB-D DB9 (hembra)
  - Antena GPS: MMCX

- **Controlador**
  - Atmega2560

- **Dimensiones**
  - 215x185x45 mm

- **Peso**
  - 480 g

20 HOJA DE CARACTERÍSTICAS TBN50-SA2048RC2SN14

Steckerbelegung / *Connector arrangement*
Winkelcodierer / *Absolute Encoder* TBN 50-SA 2048R C2 S N14

Zugehöriges Datenblatt / *Related Data Sheet :* TBX 11294

TY 7953 EB

03 / 07

- Ausgabecode: Binär
- Gegenstecker: Binder Sensor Steckverbinder Serie 763
  Kontaktanzahl: 4 (Buchse)
- Arbeitstemperaturbereich: -40°C ... +85°C
- Diskette: EDS-Datei

- *Output code: Binary*
- *Counter plug: Binder Sensor connector series 763*
- *Number of contacts: 4 (socket)*
- *Operating temperature range : -40°C ... +85°C*
- *Disk: EDS-file*

Kontakt-Nr *Pin No*	Belegt mit ...	*Connected to ...*
1	**V+** = +5,5 ... +18 Volt, Io < 80 mA (typ. Io = 60 mA) Spitzenspannung: +26 Volt / 1 Minute	***V+** = +5,5 ... +18 VDC, Io < 80 mA (typ. Io = 60 mA)* *Peak voltage: +26 VDC / 1 minute*
2	**V- / CAN GND**	***V- / CAN GND***
3	**CAN +**	***CAN +***
4	**CAN -**	***CAN -***

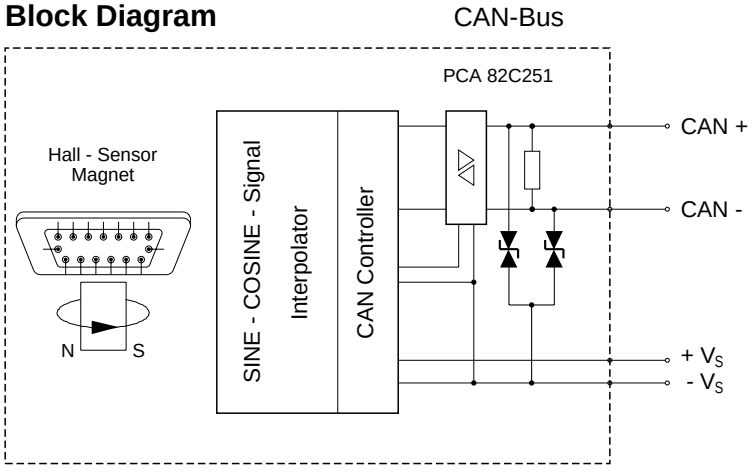
CANopen Profil für Winkelcodierer
CiA DSP 406 Version 3.0
Siehe Anwenderhandbuch TBN 11551
Zusätzliche Informationen siehe Rückseite

CANopen profile for encoders
CiA DSP 406 version 3.0
See user manual TBN 11551
Additional informations p.t.o.

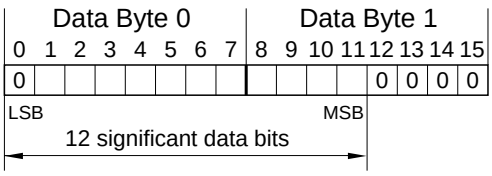
CAN-Bus - Interface

- Treiber nach ISO/DIS 11898
  Driver refers to ISO/DIS 11898

Block Diagram



PDO 1 / PDO 2



TWK-ELEKTRONIK GmbH · D-40041 Düsseldorf · PF. 10 50 63 · Heinrichstr. 85 · T. 02 11/63 20 67 · F. 02 11/63 77 05 · e-mail: info@twk.de

Lieferform

Wenn nicht anders vereinbart, werden die Winkelcodierer mit folgenden Werten eingestellt geliefert:

- Knoten-Nummer: 1
  Einstellbar über LSS oder Objekt 2000
- Baudrate: 20 kBaud

PDO 1:

Index	Sub-Index	Kommentar	Vorgabewert
1800h	1	COB-ID für PDO 1	180h + Knotenadresse
	2	transmission type (asynchron, herstellerdefiniert)	253
	3	inhibit time	0

PDO 2:

Index	Sub-Index	Kommentar	Vorgabewert
1801h	1	COB-ID für PDO 2	280h + Knotenadresse
	2	transmission type (zyklisch synchron)	1

SDO:

SDO (tx) (Codierer -> Master): 580h (1408) + Knotenadresse
SDO (rx) (Master -> Codierer): 600h (1536) + Knotenadresse
Hinweis: Die SDO-Identifier sind nicht änderbar

LSS-Adresse (siehe Identity-Objekt 1018):
Vendor ID = 0x0000 010D
Produkt ID = 0x0000 06000
Revisionsnummer: = 0x0001 0003
Seriennummer = siehe Typenschild

Bemerkung: LSS = Layer Setting Service DSP 305 V1.1.1

Busabschlußwiderstand: 120 Ohm (intern)

Programmed values as supplied

Unless agreed otherwise, the encoders are supplied with the following adjusted values

- *Node number: 1*
  Adjustable via LSS or object 2000
- *Baud rate: 20 kBaud*

PDO1:

Index	*Sub-Index*	*Comment*	*Default value*
1800h	*1*	*COB-ID for PDO 1*	*180h + Node-ID*
	2	*transmission type (asynchronous, manufacturer defined)*	*253*
	3	*inhibit time*	*0*

PDO 2:

Index	*Sub-Index*	*Comment*	*Default value*
1801h	*1*	*COB-ID for PDO 2*	*280h + Node-ID*
	2	*transmission type (cyclic synchronous)*	*1*

SDO:

SDO (tx) (Encoder -> Master): 580h (1408) + Node-ID
SDO (rx) (Master -> Encoder): 600h (1536) + Node-ID
Hint: SDO-identifiers are not changeable

LSS-adress (see identity-object 1018):
Vendor ID = 0x0000 010D
Product ID = 0x0000 6000
Revision number = 0x0001 0003
Serial number = see nameplate

Note: LSS = Layer setting service DSP 305 V1.1.1

Bus terminal resistor: 120 Ohm (internal)

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

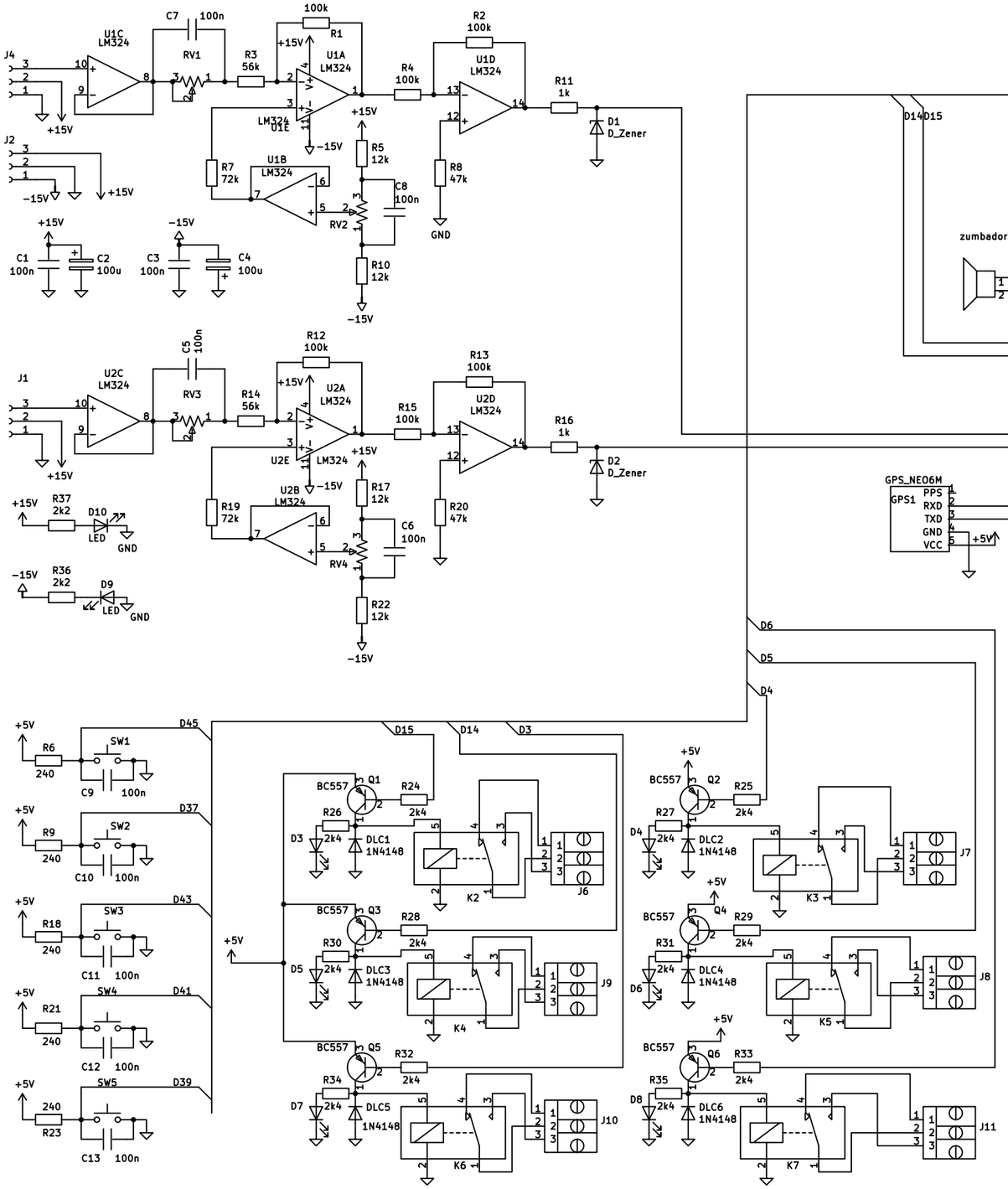
15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

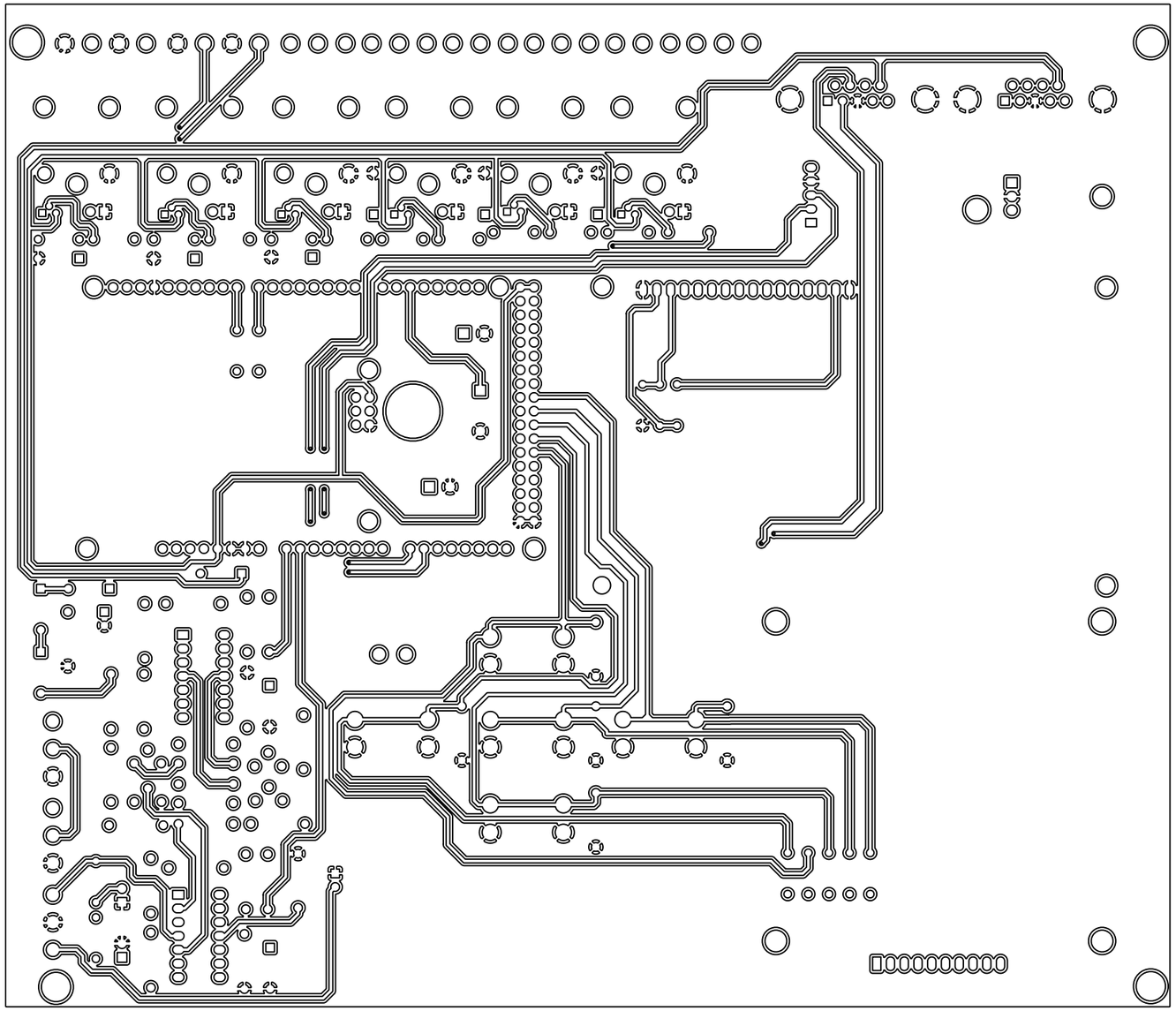
AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice de planos



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		TFG Nº: 770G01A170
TÍTULO DEL TFG SISTEMA DE SEGUIMIENTO DE SATÉLITES		
TÍTULO DEL PLANO ESQUEMA ELÉCTRICO		FECHA: 2/09/19
AUTOR GABRIEL CASAL RODRÍGUEZ	FIRMA	ESCALA: S/E
		PLANO Nº: 01



UNIVERSIDADE DA CORUÑA **ESCUELA UNIVERSITARIA POLITÉCNICA**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº:

770G01A170

TÍTULO DEL TFG

SISTEMA DE SEGUIMIENTO DE SATÉLITES

TÍTULO DEL PLANO

CIRCUITO IMPRESO. CAPA FRONT

FECHA: 2/09/19

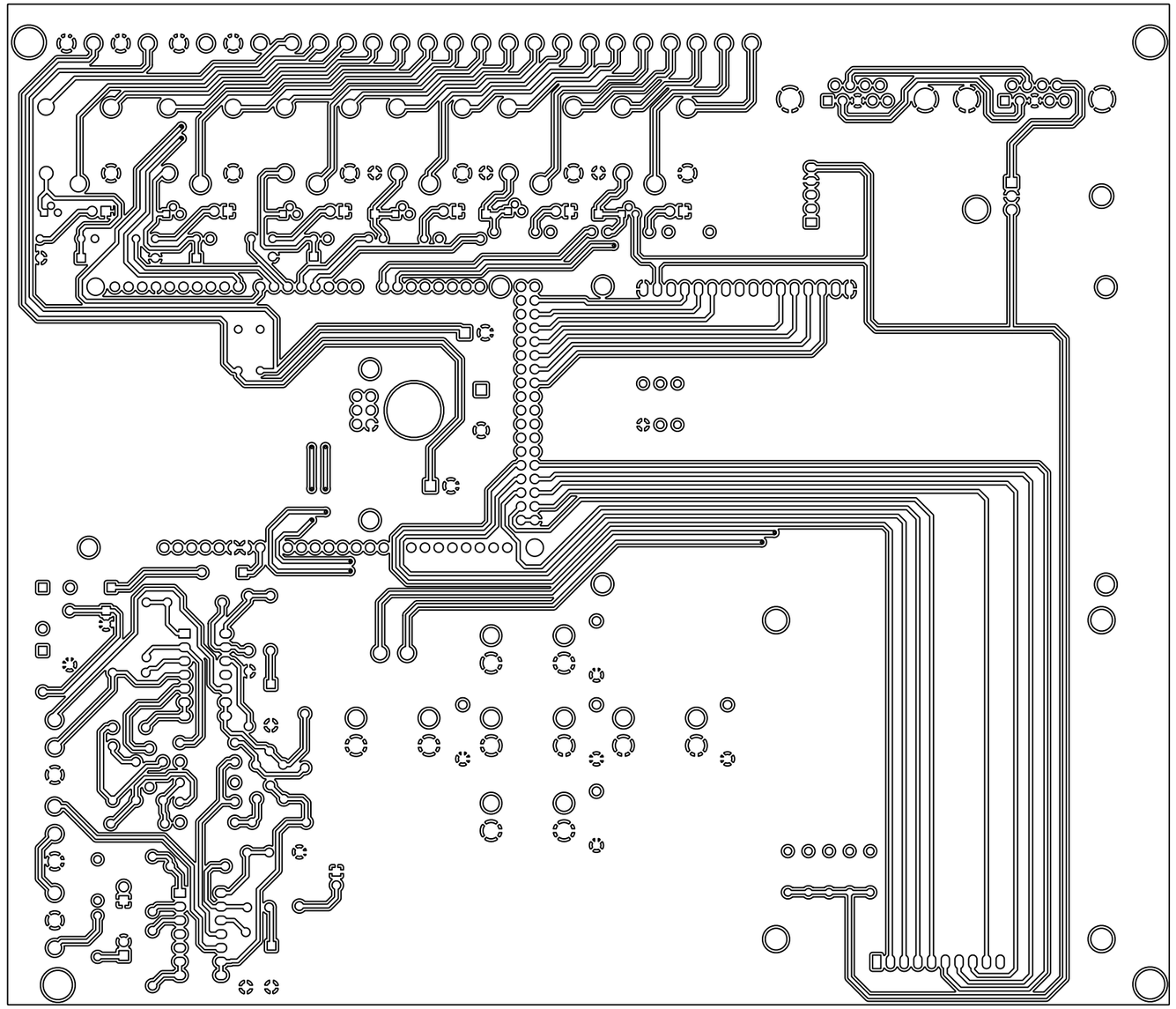
ESCALA: 1:1

AUTOR

GABRIEL CASAL RODRÍGUEZ

FIRMA

PLANO Nº: 02



UNIVERSIDADE DA CORUÑA **ESCUELA UNIVERSITARIA POLITÉCNICA**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A170

TÍTULO DEL TFG

SISTEMA DE SEGUIMIENTO DE SATÉLITES

TÍTULO DEL PLANO

CIRCUITO IMPRESO. CAPA BOTTOM

FECHA: 2/09/19

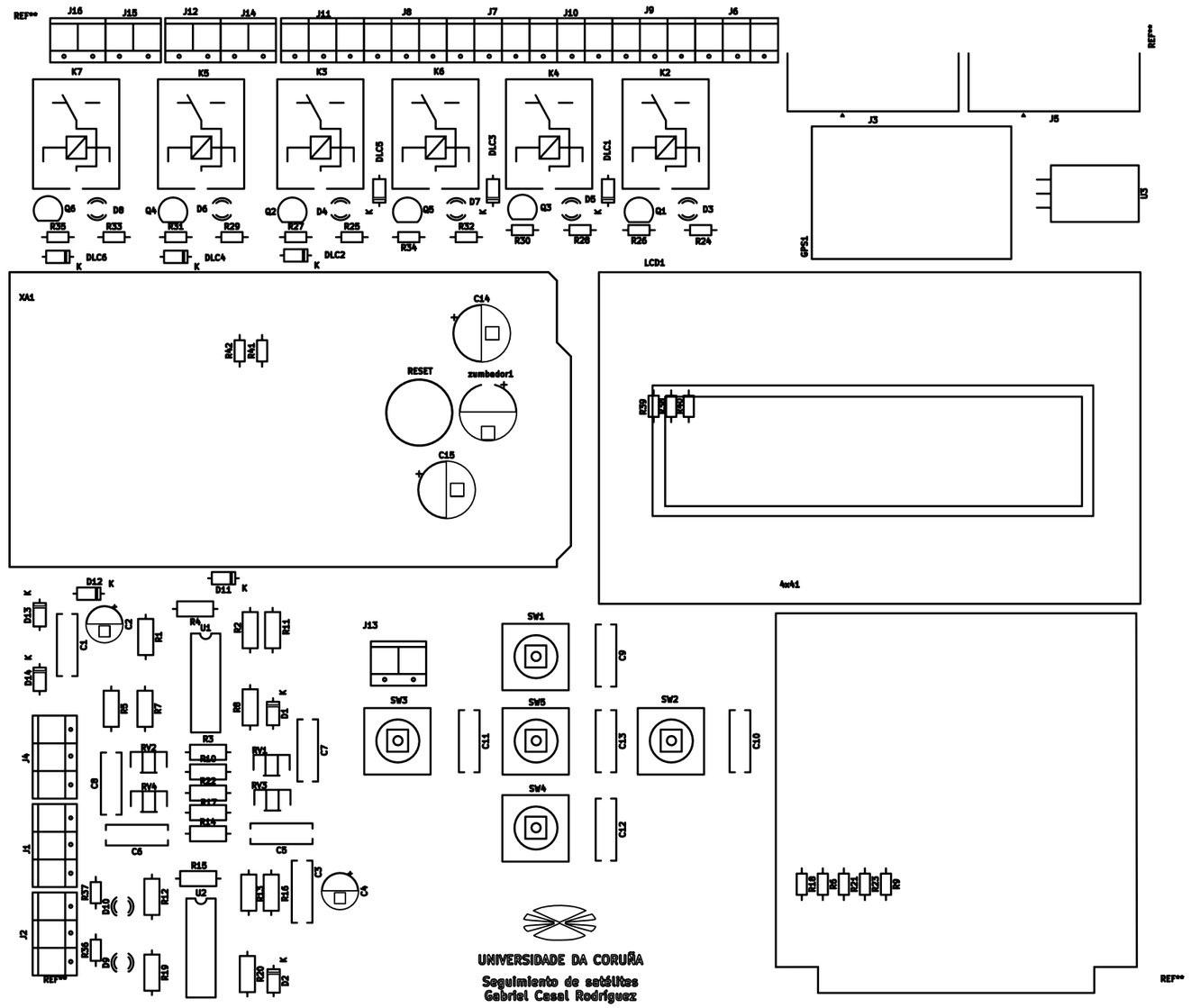
ESCALA: 1:1

PLANO Nº: 03

AUTOR

GABRIEL CASAL RODRÍGUEZ

FIRMA



UNIVERSIDADE DA CORUÑA **ESCUELA UNIVERSITARIA POLITÉCNICA**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº:

770G01A170

TÍTULO DEL TFG

SISTEMA DE SEGUIMIENTO DE SATÉLITES

TÍTULO DEL PLANO

CIRCUITO IMPRESO. SERIGRAFÍA FRONTAL

FECHA: 2/09/19

ESCALA: 1:1

AUTOR

GABRIEL CASAL RODRÍGUEZ

FIRMA

PLANO Nº: 04

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice del documento PLIEGO DE CONDICIONES

20.1. Especificaciones de las unidades de obra

20.1.1. Selección de componentes

Se seleccionarán para la implementación del proyecto, en la medida de lo posible, componentes de fácil acceso económico, cumpliendo en cualquier caso los requisitos de diseño establecidos.

20.1.2. Sustitución de componentes

Para la situación de no disponibilidad o descatalogación de alguno de los componentes requeridos por el sistema, deberán seleccionarse otros de iguales prestaciones, cuando menos, al mismo tiempo que resulten completamente compatibles.

20.2. Verificaciones

20.2.1. Testeo del circuito impreso

La placa de circuito impreso diseñada ha de ser testeada en su proceso de fabricación para garantizar su correcto funcionamiento antes del montaje.

20.2.2. Verificaciones de diseño

Se verificará el funcionamiento de los componentes montados sobre la placa de circuito impreso, garantizando la disponibilidad de todas sus funciones.

Para el correcto funcionamiento del sistema, debe garantizarse la conexión de la alimentación de doble polaridad de la placa antes de conectar señales de realimentación analógicas.

20.2.3. Verificaciones de montaje

Este proyecto está enmarcado en una colaboración con el CIFP Ferrolterra, que deberá proporcionar el sistema de antenas.

En caso de obtener dicho sistema de antenas correctamente implementado antes del 1 de Septiembre de 2019, se realizarán las pruebas pertinentes del sistema implementado en este proyecto sobre el montaje ofrecido por el instituo Ferrolterra.

De no disponer en plazo de su montaje, se realizarán las pruebas pertinentes en una maqueta a escala.

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice del documento MEDICIONES

21 Listado de materiales

A continuación se adjunta la tabla 21.1 con el listado de materiales necesario para la elaboración del proyecto.

Lista de materiales	
Componente	Unidades
Arduino MEGA 2560	1
Cable USB Tipo A	1
Arduino CAN shield	1
Teclado matricial 4x4	1
LCD 20x04	1
Receptor GPS GY-NEO6MV2 NEO-6M	1
Conector SUB-D DB9 hembra	2
Conector SUB-D DB9 macho	2
Carcasa SUB-D DB9	2
Encoder TBN50-SA2048RC2SN14	2
Cable 4 pin M12	2
Relé songle srd-05vdc-sl-c	6
BC557 (yransistor PNP)	6
Diodo 1N4148	6
Diodo 1N4007	4
Diodo LED (5mm)	8
LM324N	2
Potenciómetro multivuelta 500k	2
Potenciómetro multivuelta 5k	2
Condensador tántalo 2,2μF	2
Condensador cerámico 100nF	11
Condensador electrlítico 100μF 35V	2
Pulsador de membrana B3F-4050	5
Diodo Zener BZX85C 1W	2
Resistencia 5 % tolerancia (0,25W)	42
Clema de 3 conectores	9
Clema de 2 conectores	5
Kit de pines compatibles con Arduino	3
Regulador de tensión L7805 (TO-220)	1
Disipador térmico TO-220	1
Placas de circuito impreso	1

Tabla 21.1 – Lista de materiales

TÍTULO: **SISTEMA DE SEGUIMIENTO DE SATÉLITES**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: GABRIEL CASAL RODRÍGUEZ

Índice del documento PRESUPUESTO

22 PRECIOS UNITARIOS DE MANO DE OBRA, SOFTWARE Y ELEMENTOS AUXILIARES

22.1. Mano de obra

Teniendo en cuenta que se considera que el salario en €/hora para un graduado en Ingeniería Electrónica Industrial y Automática es de 15€/hora, el coste por mano de obra es el que se muestra en la tabla 22.1:

Actividad	Horas	Coste (€/hora)	Total (€)
Programación del software	100	15,00	1500,00 €
Diseño del esquema eléctrico	15	15,00	225,00 €
Diseño del circuito impreso	30	15,00	450,00 €
Soldadura del circuito impreso	2	15,00	30,00 €
Redacción del documento	90	15,00	1350,00 €
Total			3555,00 €

Tabla 22.1 – Lista de materiales

22.2. Licencias software

Licencia	Coste (€)
PstRotator	20,00 €
Total	20,00 €

Tabla 22.2 – Coste software

22.3. Elementos auxiliares

Elemento	Coste (€)
Envío de placas PCB	13,20 €
Total	13,20 €

Tabla 22.3 – Costes auxiliares

23 PRECIOS UNITARIOS DE LAS UNIDADES DE OBRA

Concepto	Unidades	Coste unitario (I.I.) (€/ud)	Total (€)
Arduino MEGA 2560 [12]	1	5,61 €	5,61 €
Cable USB Tipo A [26]	1	0,72 €	0,72 €
Arduino CAN shield [58]	1	5,10 €	5,10 €
Teclado matricial 4x4 [71]	1	5,65 €	5,65 €
LCD 20x04 [43]	1	4,99 €	4,99 €
Receptor GPS GY-NEO6MV2 NEO-6M [50]	1	3,70 €	3,70 €
Conector SUB-D DB9 hembra [33]	2	1,75 €	3,50 €
Conector SUB-D DB9 macho [34]	2	0,67 €	1,34 €
Carcasa SUB-D DB9 [38]	2	0,77 €	1,54 €
Encoder TBN50-SA2048RC2SN14 [75]	2	20,99 €	41,98 €
Cable 4 pin M12 [25]	2	10,05 €	20,10 €
Relé songle srd-05vdc-sl-c [67]	6	1,56 €	9,36 €
BC557 (transistor PNP) [74]	6	0,29 €	1,74 €
Diodo 1N4148 [40]	6	0,15 €	0,90 €
Diodo 1N4007 [41]	4	0,13 €	0,52 €
Diodo LED (5mm) [56]	8	0,17 €	1,36 €
LM324N [29]	2	0,96 €	1,92 €
Potenciómetro multivuelta 500k [7]	2	3,52 €	7,04 €
Potenciómetro multivuelta 5k [8]	2	1,91 €	3,82 €
Condensador tántalo 22μF [32]	2	0,66 €	1,32 €
Condensador cerámico 100nF [30]	11	0,27 €	0,54 €
Condensador electrlítico 100μF/ 35V [31]	2	0,22 €	0,44 €
Pulsador de membrana B3F-4050 [64]	5	1,96 €	9,80 €
Diodo Zener BZX85C 1W [82]	2	0,56 €	1,12 €
Resistencia 5 % tolerancia (0,25W) [68]	42	0,04 €	1,68 €
Clema de 3 conectores [22]	9	0,38 €	3,42 €
Clema de 2 conectores [23]	5	0,28 €	1,40 €
Kit de pines compatibles con Arduino [55]	3	2,78 €	8,34 €
Regulador de tensión L7805 (TO-220) [66]	1	0,80 €	0,80 €
Disipador térmico TO-220 [42]	1	0,28 €	0,28 €
Placas de circuito impreso	5	3,19 €	15,95 €
Base imponible: 137,17 €	**IVA: 21 %**	**Cuota IVA: 28,8 €**	
Total			**165,98 €**

Tabla 23.1 – Unidades de obra

24 PRESUPUESTO

Elemento	Coste (€)
Mano de obra	3555,00 €
Licencias software	20,00 €
Elementos auxiliares	13,20 €
Unidades de obra	165,98 €
Total	3754,18 €

Tabla 24.1 – Presupuesto total

El presupuesto total del proyecto asciende a la cantidad de **tres mil setecientos cincuenta y cuatro euros con dieciocho céntimos (3754,18 €)**.