Universidad de Zaragoza

ESCUELA DE INGENIERÍA TÉCNICA INDUSTRIAL DE ZARAGOZA

Dpto. Ingeniería Electrónica y Comunicaciones

"SISTEMA DE CONTROL DE RIEGO AUTOMÁTICO MEDIANTE SMS"

PROYECTO FIN DE CARRERA

INGENIERÍA TÉCNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL

AUTOR:
Iris Aranda Villalba

DIRECTOR DE PROYECTO:
José Ignacio Artigas

CONVOCATORIA:
Septiembre 2012

RESUMEN

En el presente documento se justifica el trabajo que se ha ido efectuando durante la realización del proyecto titulado SISTEMA DE CONTROL DE RIEGO AUTOMÁTICO MEDIANTE SMS.

En el primer capítulo se detallan las siglas, abreviaciones y términos utilizados a lo largo de todo el documento, a continuación una idea global del proyecto y el porqué de la elección de éste.

Para que se pueda observar cómo han transcurrido los hechos a lo largo de la ejecución del proyecto se describen diferentes enfoques. Estos enfoques son las diferentes soluciones para los problemas surgidos.

Para el enfoque final se analizan las posibles soluciones que se pueden adoptar para la construcción de las diferentes partes del circuito, eligiendo de forma razonada una opción válida que se adapte lo mejor posible al sistema.

En lo referente al diseño hardware se detallan las características de cada uno de los circuitos integrados utilizados y se introducen sus diagramas de bloques; se calcula el valor de los componentes y se expone su función. También se comentan las dificultades encontradas durante las pruebas del hardware.

Para la programación del riego se desarrolló una interfaz de usuario sencilla, de la que hay una buena y clara explicación en el documento.

El software del programa principal se encuentra descrito de una manera concisa y representado a través de los diagramas de bloques correspondientes.

Se hace referencia de forma breve a los comandos AT utilizados para la configuración del módulo GSM, el manejo de SMS, la introducción del pin…

Como conclusión del proyecto se exponen los objetivos personales logrados con su elaboración y algunos planteamientos para posibles líneas futuras.

El documento contiene seis anexos con el fin de servir de apoyo a la memoria y no incrementarla en exceso. Estos anexos incluyen el circuito completo, los planos de pistas de las placas, un desarrollo de los comandos AT (qué son, para que sirven y descripción completa de todos los utilizados en el proyecto), los diagramas de bloques que no corresponden al programa principal y el código C del programa.

A MI FAMILIA…

AGRADECIMIENTOS

En primer lugar quiero nombrar a mis abuelos, Lolo y María, recientemente fallecidos y a los que sin duda les hubiera encantado poder acompañarme estos días. Gracias por todos los momentos vividos a vuestro lado. Os quiero.

A mi hermano, Iván, con el que he convivido durante los años de carrera. Gracias a ti todo ha sido mucho más fácil. Gracias especialmente por el apoyo que junto a Patricia me habéis dado durante este último mes. Os quiero.

A mis padres, Juanje y Eva, por su apoyo moral y económico y por estar ahí siempre que los he necesitado. Os quiero.

Sin vosotros esto no hubiera sido posible.

A mi director, José Ignacio Artigas, por prestarme su tiempo y guiarme durante el proyecto.

Gracias a los maestros de taller, especialmente a Álvaro, ya que ha estado disponible siempre que lo he requerido.

ÍNDICE

1 SIGLAS, ABREVIACIONES Y TÉRMINOS 6
2 INTRODUCCIÓN 10
3 ANTECEDENTES 11
3.1 OBJETIVO 11
3.2 ENFOQUE 11
3.2.1 ENFOQUE Nº1 11
3.2.2 ENFOQUE Nº2 12
3.2.3 ENFOQUE Nº3 12
3.2.4 ENFOQUE Nº4 13
3 ANÁLISIS DE SOLUCIONES Y SOLUCIÓN ADOPTADA 15
3.1 GSM 15
3.1.1 OPCIÓN 1: MODEM GSM TIPO TERMINAL 15
3.1.2 OPCIÓN 2: MÓDULO GSM 16
3.2 MICROCONTROLADOR 16
3.2.1 ALTERNATIVA 1: LA SERIE MSP430X1XX 16
3.2.2 ALTERNATIVA 2: LA FAMILIA MSP430AFE2XX 17
3.2.3 ALTERNATIVA 3: EL MICROCONTROLADOR MSP430F2252 17
3.3 FUENTE DE ALIMENTACIÓN MÓDULO GSM. 17
3.4 FUENTE DE ALIMENTACIÓN MICROCONTROLADOR 18
3.5 TARJETA Y CONECTOR SIM 19
3.5.1 TARJETA SIM 19
3.5.2 CONECTOR SIM 19
3.6 ELECTROVÁLVULA O VÁLVULAS DE SOLENOIDE 20
3.6.1 TIPOS DE SOLENOIDES 20
3.7 ETAPA DE ADAPTACIÓN Y CONTROL 25
3.8 SOLENOIDE PARA LA ELECTROVÁLVULA 28
3.8.1 SOLENOIDES DE TRES TERMINALES 29
3.8.2 SOLENOIDE DE DOS TERMINALES 30
3.8.3 CONCLUSIÓN 32

3.9 ETAPA DE ADAPTACIÓN DE LA TENSIÓN ENTRE LOS PINES TX DEL MICRO Y RX DEL MÓDULO 33
3.9.1 SOLUCIÓN 1: DIODO ZENER 33
3.9.2 SOLUCIÓN 2: BUFFER 34
3.9.3 SOLUCIÓN 3: DIODO SCHOTTKY 35
4 HARDWARE 36
4.1 DIAGRAMA DE BLOQUES 36
4.2 FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 37
4.2.1 CARACTERÍSTICAS DEL LT3680 37
4.2.2 DIAGRAMA DE BLOQUES DEL LT3680 37
4.2.3 CIRCUITO 38
4.2.4 COMPONENTES: VALOR Y FUNCIÓN 38
4.2.5 ONDA DE TENSIÓN DE SALIDA OBTENIDA EN EL LABORATORIO 45
4.3 MÓDULO GSM. SAGEM HILO MODULE 46
4.3.1 CARACTERÍSTICAS 46
4.3.2 DIAGRAMA DE BLOQUES DEL MÓDULO GSM 47
4.3.3 CIRUITO DEL CONECTOR DEL MÓDULO GSM 48
4.4 FUENTE DE ALIMENTACIÓN DEL MC 48
4.4.1 CARACTERÍSTICAS DEL TPS71501DCKR 48
4.4.2 DIAGRAMA DE BLOQUES DEL TPS7150DCKR 49
4.4.3 CIRCUITO 49
4.4.4 COMPONENTES: VALOR Y FUNCIÓN 49
4.5 MICROCONTROLADOR MSP430F2252 50
4.5.1 CARACTERÍSTICAS 50
4.5.2 DIAGRAMA DE BLOQUES 51
4.5.3 CIRCUITO 51
4.5.4 COMPONENTES: VALOR Y FUNCIÓN 52
4.6 ETAPA DE ADAPTACIÓN DE TENSIÓN ENTRE TX DEL MC Y RX DEL MÓDULO GSM 54
4.6.1 CIRCUITO 54
4.6.2 COMPONENTES: VALOR Y FUNCIÓN 54
4.7 ETAPA DE ADAPTACIÓN Y CONTROL DE LAS ELECTROVÁLVULAS 55
4.7.1 CIRCUITO 55

4.7.2 COMPONENTES: CÁLCULO Y FUNCIÓN 55
4.8 PRUEBAS DEL HARDWARE 56
5 CIRCUITO DE CONFIGURACIÓN DEL MÓDULO GSM 57
5.1 FUNCIÓN 57
5.2 DIAGRAMA DE BLOQUES 57
6 SOFTWARE 58
6.1 INTERFAZ USUARIO-SISTEMA 58
6.1.1 SIGN UP 58
6.1.2 MODIFICAR LA CLAVE 59
6.1.3 CONFIGURAR RIEGO 59
6.1.4 RIEGO INMEDIATO 59
6.1.5 FINALIZAR RIEGO, DESACTIVAR RIEGO POR CONFIGURACIÓN 59
6.1.6 RESTABLECER RIEGO POR CONFIGURACIÓN 60
6.1.7 REMOVE ACCOUNT 60
6.2 COMANDOS AT 60
6.2.1 COMANDOS DE CONFIGURACIÓN 60
6.2.2 COMANDOS AT PARA EL MANEJO DE SMS 67
6.2.3 COMANDO AT PARA LA OBTENCIÓN DE LA HORA Y FECHA 69
6.2.4 COMANDO AT PARA INTRODUCIR EL PIN 69
6.3 PROGRAMA PRINCIPAL 69
7 CONCLUSIONES Y FUTUROS DESARROLLOS 76
7.1 CONCLUSIONES 76
7.2 FUTUROS DESARROLLOS 76
8 REFERENCIAS 78
8.1 CONVERTIDOR BUCK 78
8.2 MODULO GSM 78
8.3 REGULADOR LINEAL DE VOLTAJE 78
8.4 MICROCOTROLADOR 78
8.5 TARJETA SIM 78
8.6 ELECTROVÁLVULAS 79
8.7 MINMIZACIÓN DE EMI 79
8.8 COMANDOS AT 79

8.9 CÓDIGOS DE CODIFICACIÓN 79
ANEXO Nº1: ESQUEMAS DEL CIRUCITO 80
FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 80
CONECTOR DEL MÓDULO GSM Y SIM 81
MC 82
ETAPA DE SALIDA 83
PLACA PROPORCIONADA POR LIBELIUM 84
ANEXO Nº2: PLANOS DE PISTAS DE LAS PLACAS 85
FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 86
PLANO DE PISTAS CARA TOP 86
PLANO DE PISTAS CARA BOTTOM 87
MÓDULO GSM 88
PLANO DE PISTAS CARA TOP 88
PLANO DE PISTAS CARA BOTTOM 89
MC 90
PLANO DE PISTAS CARA TOP 90
PLANO DE PISTAS CARA BOTTOM 91
ETAPA DE SALIDA 92
PLANO DE PISTAS CARA TOP 92
PLANO DE PISTAS CARA BOTTOM 93
ANEXO Nº3: COMANDOS AT 94
INTRODUCCIÓN 94
SINTAXIS DE LOS COMANDOS AT 94
POSIBLES RESPUESTAS 95
TIPOS DE COMANDOS AT 97
DETALLE DE LOS COMANDOS AT UTILIZADOS 98
PR COMMAND: SET FIXED LOCAL RATE 98
+CREG COMMAND: NETWORK REGISTRATION 98
+CMGF COMMAND: SELECT SMS MESSAGE FORMAT 99
+CSCS COMMAND: SET TE CHARACTER SET 99
+CPMS COMMAND: PREFERRED MESSAGE STORAGE 99
+CSDH COMMAND: SHOW TEXT MODE PARAMETERS 100

+CNMI COMMAND: NEW SMS MESSAGE INDICATION 101
+CMGR COMMAND: READ SMS MESSAGE 102
+CMEE COMMAND: REPORT MOBILE TERMINATION ERROR 103
+CMGS COMMAND: SEND SMS MESSAGE 103
+CMGW COMMAND: WRITE SMS MESSAGE TO MEMORY 104
+CMSS COMMAND: SEND SMS MESSAGE FROM STORAGE 104
+CMGD COMMAND: DELETE SMS MESSAGE 104
ANEXO Nº4: SISTEMAS DE CODIFICACIÓN DE CARACTERES ACEPTADOS POR EL MÓDULO GSM 106
CÓDIGO GSM-7 BIT DFAULT ALPHABET 106
CÓDIGO IRA (ANTERIORMENTE IA5) 108
CÓDIGO UCS2 109
ANEXO Nº5: DIAGRAMAS DE BLOQUES DEL PROGRAMA 110
ANEXOS Nº 6: CÓDIGO DEL PROGRAMA 145
ÍNDICE DE ILUSTRACIONES 174
ÍNDICE DE TABLAS 177

1 SIGLAS, ABREVIACIONES Y TÉRMINOS

ASCII: American Standard Code for Information Interchange o código estándar estadounidense para el intercambio de información.

CBM: Cell Broadcast Message. Envío de SMS masivo.

Cable conversor de USB a RS-232: convierte los niveles de tensión utilizados por los puertos USB a los requeridos por el protocolo RS-232 y viceversa.

Comandos AT: instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal módem/GSM. Para conocer más acerca de los comandos AT ir al anexo número tres.

Conector DB9: conector de 9 clavijas. El conector DB9 se utiliza principalmente para conexiones en serie, ya que permite una transmisión asíncrona de datos según lo establecido en la norma RS-232.

CPU: central processing unit, unidad de procesamiento central.

DCR: Resistencia DC equivalente de una bobina.

DCS 1800: usa 1710.2–1784.8 MHz para enviar información desde la estación móvil a la estación base (uplink) y 1805.2–1879.8 MHz para la otra dirección (downlink).

DRX9: Modo de recepción discontinua en el que el módulo GSM se despertará en periodos de nueve “multiframes” para realizar las operaciones pertinentes. Una “multiframe” equivale a 0.125s por lo que el módulo se despertará cada 1.125s.

encapsulado DFN: en inglés, Dual Flat No Lead. Este tipo de encapsulado utiliza pads en lugar de pines.

EGSM 900: en algunos países se ha ampliado la banda GSM-900 para cubrir un rango de frecuencias mayor. Esta 'extendida GSM', E-GSM, usa 880–915 MHz (uplink) y 925–960 MHz (downlink).

EMI: interferencias electomagnéticas.

ESL: Equivalent Series Inductance. Inductancia serie equivalente. Representa la inductancia en serie correspondiente a los elementos parásitos.

ESR: Equivalent Series Resistance. Representa la resistencia en serie correspondiente a los elementos parásitos.

ETSI: European Telecommunications Standards Institute (ETSI) o instituto europeo de normas de telecomunicaciones. Es una organización de estandarización de la industria de las telecomunicaciones (fabricantes de equipos y operadores de redes) de Europa, con proyección mundial. El ETSI ha tenido gran éxito al estandarizar el sistema de telefonía móvil GSM.

FLASH: tecnología de almacenamiento —derivada de la memoria EEPROM— que permite la lectura-escritura de múltiples posiciones de memoria en la misma operación. Permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM.

GMT: Greenwich Mean Time.

GPRS: General Packet Radio Service o servicio general de paquetes vía radio.

GSM: sistema global para las comunicaciones móviles. Proviene del francés (Groupe Spécial Mobile). Es un sistema digital de telefonía móvil que provee un estándar común para los usuarios, permitiendo el roaming internacional y la capacidad de ofrecer a alta velocidad servicios avanzados de transmisión de voz, datos y vídeo.

GSM-1800: usa 1710–1785 MHz para enviar información desde la estación móvil al transceptor de la estación base (uplink) y 1805–1880 MHz para la otra dirección (downlink). Junto a la GSM-900 es la más utilizada en Europa, ambas se utilizan en España.

GSM 850: usa 824–849 MHz para enviar información desde la estación móvil a la estación base (uplink) y 869–894 MHz para la otra dirección (downlink).

GSM-900: banda de frecuencia GSM que usa 890–915 MHz para enviar información desde la estación móvil a la estación base (uplink) y 935–960 MHz para la otra dirección (downlink). Es la más ampliamente usada.

IA5: International Alphabet No. 5 o alfabeto internacional Nº5.

IRA: International Reference Alphabet o alfabeto internacional de referencia.

IRV: International Reference Version o versión internacional de referencia. Se creó debido a la compatibilidad de varias versiones nacionales. Es la versión desde la que se parte para crear las variantes nacionales, únicamente unos ciertos caracteres de la IRV pueden ser remplazados.

LCD: Liquid Crystal Display o pantalla de cristal líquido.

LED: Light-Emitting Diode, diodo emisor de luz.

Lenguaje C: lenguaje de programación creado en 1972 por Dennis M. Ritchie. La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es extremadamente portátil.

LPMx: Low Power Mode. Modo de bajo consumo de un µC de la familia MSP430 de Texas Instruments. X puede tomar valores de 0 a 5, siendo LPM5 el modo en el que el µC consume la menor cantidad de corriente.

Microcontrolador (µC): circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento (CPU), memoria (RAM Y ROM) y periféricos de entrada/salida.

MMS: Multimedia Messaging System o sistema de mensajería multimedia. Es un estándar de mensajería que permite enviar y recibir contenidos multimedia, incorporando sonido, video u fotos.

Módem GSM o equipo terminal: Los equipos terminales son equipos que el fabricante vende ya en una caja, listos para ser utilizados. La caja dispone normalmente de un puerto de comunicaciones RS232 (a veces también USB), a veces también dispone de un conector para Entradas/Salidas digitales, y siempre dispone de un conector para la antena y otro para la alimentación. Estos equipos suelen utilizarse con equipos ya acabados en los que no es posible hacer un rediseño.

Módulo GSM: es el módulo que va dentro de un equipo terminal. Está preparado para añadirlo físicamente a nuestro circuito como un componente más

MOSFET: en ingles, Metal-Oxide-Semiconductor Field-Effect Transistor.

PCB: Printed Ciruit Board o placa de circuito impreso.

PCS 1900: usa 1850.0–1910.0 MHz para enviar información desde la estación móvil a la estación base (uplink) y 1930.0–1990.0 MHz para la otra dirección (downlink).

PWM: Pulse Width Modulation. Modulación de la anchura de pulso.

RAM: random-access memory, memoria de acceso aleatorio.

Roaming: la capacidad de enviar y recibir llamadas en redes móviles fuera del área de servicio local de la propia compañía, es decir, dentro de la zona de servicio de otra empresa del mismo país, o bien durante una estancia en otro país diferente, con la red de una empresa extranjera.

ROM: read-only memory. Memoria de solo lectura. Es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la información. Los datos almacenados en la ROM no se pueden modificar, o al menos no de manera rápida o fácil. Se utiliza principalmente para contener el programa que va a controlar al hardware.

RS-232: Recommended Standard-232. Estándar para la conexión serial de señales de datos binarias entre un DTE (Equipo terminal de datos) y un DCE (Equipo de terminación del circuito de datos). Un cero lógico se corresponde con un valor de 5 a 15V mientras que un uno lógico se corresponde con un valor que va de -5 a -15V.

RS-232 transceiver: adapta los niveles de tensión de un dispositivo a los requeridos por el estándar RS-232 y viceversa.

RTC: en inglés Real Time Clock. Périferico del µC que mantiene la hora y la fecha.

THD: Through Hole Device o dispositivos de montaje por inserción.

SIM: en inglés, Subscriber Identity Module, en español módulo de identificación del suscriptor. Es una tarjeta inteligente desmontable usada en teléfonos móviles y módems.

SMS: en inglés sigla de servicio de mensajes cortos ("Short Message Service").

SMS-STATUS-REPORT: informe de estado enviado del SMSC al dispositivo móvil.

SMS-SUBMIT: el SMS es enviado desde un dispositivo móvil a la SMSC.

SMSC: Short Message Service Center o central de servicio de mensajes cortos, es un elemento de la red de telefonía móvil cuya función es la de enviar/recibir mensajes SMS. En el momento que un usuario envía un mensaje de texto (SMS) a otro usuario lo que sucede es que el teléfono envía dicho mensaje a la SMSC correspondiente al operador del usuario remitente. La SMSC guarda el mensaje y lo entrega a su destinatario cuando este se encuentra en cobertura.

TA: Terminal Adaptador, por ejemplo una tarjeta de datos.

TE: dispositivo introducido por el usuario que controla el GSM.

Timer: periférico del µC. Es un temporizador o contador con modos de captura o comparación y con la capacidad de generar múltiples salidas o interrupciones. El modo de captura se utiliza para capturar eventos temporales mientras que el modo de comparación es usado para generar ondas PWM o generar interrupciones en intervalos de tiempo específicos.

UART: Universal Asynchronous Receiver/Transmitte. Su misión principal es convertir los datos recibidos del bus del PC/µC en formato paralelo, a un formato serie que será utilizado en la transmisión hacia el exterior. También realiza el proceso contrario: transformar los datos serie recibidos del exterior en un formato paralelo entendible por el bus. Universal indica que el formato de los datos y la velocidad de transmisión son configurables y los niveles de tensión son adaptados por un circuito externo.

USB: en inglés, Universal Serial Bus.

WAP: Wireless Application Protocol o protocolo de aplicaciones inalámbricas. Es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

2 INTRODUCCIÓN

Hace unos meses como todo estudiante de ingeniería técnica me vi envuelta en la elección de mi proyecto fin de carrera, deseaba un proyecto que pudiera ampliar mis conocimientos de electrónica y que fuera útil para facilitar la vida cotidiana de la población. Estas características las encontré en un proyecto de la lista del curso 2010/2011 denominado SISTEMA DE CONTROL DE RIEGO AUTOMÁTICO MEDIANTE SMS.

Este proyecto me ha permitido reforzar y ampliar mis conocimientos de lenguaje C adquiridos en la carrera además de introducirme en un mundo completamente desconocido para mí, los comandos AT. También me ha permitido aplicar estructuras de la electrónica de potencia como los convertidores dc-dc buck y los drivers.

La idea a rasgos generales del proyecto sería el control de dos electroválvulas mediante un microcontrolador (de ahora en adelante µC) para el riego de dos secciones totalmente independientes, pudiendo modificar el programa de riego a través del envío de SMS. Para realizar dicha modificación es necesario un módem GSM el cual reciba los SMS y transmita el contenido del mensaje de texto al µC.

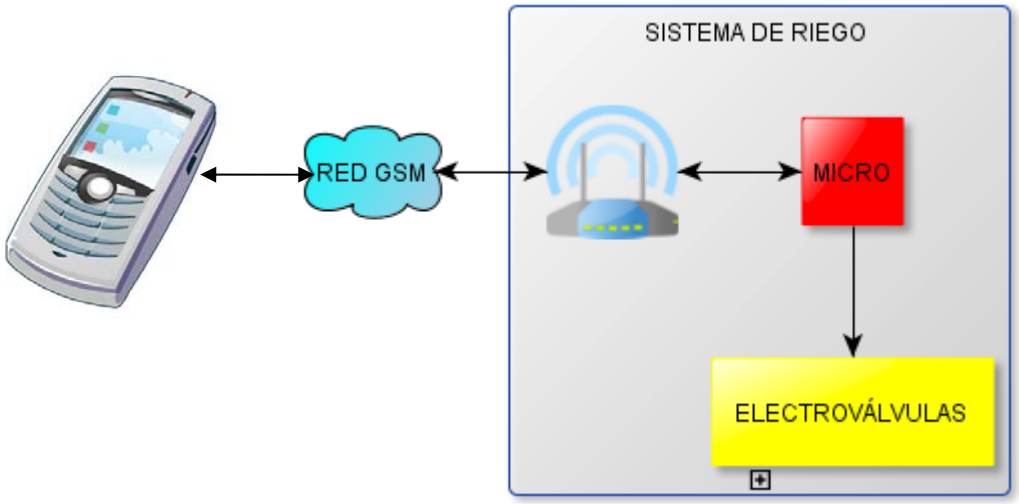


Ilustración 1: IDEA GLOBAL DEL PROYECTO

3 ANTECEDENTES

3.1 OBJETIVO

El objetivo del proyecto es poder llevar a cabo todas las funciones que se realizan con los programadores de riego convencionales pero sin la necesidad de que el usuario esté presente en la instalación, simplemente con el envío de SMS al sistema de riego. Con este sistema se va a poder programar el riego de dos sectores de forma totalmente independiente, ofreciendo a un posible usuario final las siguientes funcionalidades:

- Posibilidad de programar el/los día/días de la semana en los que se va a regar, la hora de comienzo de riego y el tiempo de riego. Programable un único riego al día por sector.
- Riego inmediato.
- Anular un riego inmediato o programado antes de que se inicie.
- Finalizar el riego antes de que se haya cumplido el tiempo programado.
- Posibilidad de anular el riego por configuración hasta que así se desee.

3.2 ENFOQUE

Para alcanzar lo anteriormente expuesto se llevaron a cabo diferentes enfoques del proyecto:

3.2.1 ENFOQUE Nº1

- Diseño de una única fuente de alimentación cuya energía sería obtenida a través de una batería de 9V.
- Módem GSM tipo terminal.
- Dos electroválvulas tipo *latch* que serían controladas a través de algunas de las salidas del µC. Estas necesitan niveles de tensión y de corriente superiores a los que puede proporcionar el µC, por tanto fue necesario diseñar un circuito que adaptara esos niveles.
- Interfaz módem GSM ←→ µC a través del protocolo de comunicación serie RS-232 el cual nos va a permitir el envió de los comandos AT necesarios desde el µC al módem GSM para configurarlo y proporcionarle instrucciones, y conocer la respuesta del segundo ante las instrucciones y configuraciones enviadas por el primero. Esta interfaz también nos va a permitir el que el usuario tenga control e información sobre el sistema.

- Necesidad de introducir un *transceiver* que convierta los niveles de tensión de las señales del puerto serie RS-232 del módem GSM a niveles compatibles con los del µC.

3.2.2 ENFOQUE Nº2

Al estudiar las hojas de características de los módem GSM tipo terminal se comprobó que todos tenían un consumo bastante elevado además de ser caros, por lo tanto se optó por empezar a buscar módulos GSM. Se eligió el módulo Hilo de SAGEMCOM.

Para conseguir el menor consumo posible se decidió apagar el módulo GSM y encenderlo cada cierto tiempo para comprobar si se había recibido algún SMS ya que en modo *sleep* el consumo era notablemente superior además de utilizar los modos de bajo consumo presentes en el µC.

Los niveles de tensión a la salida de la interfaz serie del módulo GSM son más similares a los del µC, por lo tanto no se requiere la utilización de un *transceiver* como podría ser el conocido MAX232.

3.2.3 ENFOQUE Nº3

Para la construcción de la fuente de alimentación del sistema me decanté por un convertidor dc-dc buck ya que ofrece una alta eficiencia, algo imprescindible para un sistema alimentado a través de una batería, además de ser una de las configuraciones recomendadas en el datasheet del módulo GSM HILO. En él aparecía el circuito integrado LT1913, lo que garantizaba su buen funcionamiento. En un primer momento se escogió este integrado para alimentar a todo el sistema, pero surgió el siguiente inconveniente: la mayor parte del tiempo el módulo GSM iba a permanecer apagado teniendo que alimentar la fuente de alimentación únicamente al µC.

El sistema de riego, como cualquier dispositivo portátil, opera con cargas bajas la mayor parte del tiempo (µC en modo LPM3, unos 4µA) a la vez que requiere en momentos puntuales una potencia elevada (picos de corriente demandada por el módulo de hasta 2.2A).

Lo ideal sería conseguir una alta eficiencia para todo el rango de carga y aumentar de este modo la vida útil de la batería, pero desafortunadamente cuando un convertidor conmutado diseñado para alta eficiencia en potencias altas opera en baja carga su eficiencia se ve disminuida considerablemente.

Este hecho se debe a la necesidad de una alta frecuencia de conmutación y al consumo de los elementos internos del circuito integrado.

Mediante una frecuencia elevada podemos reducir el valor de los componentes del filtro LC, y por tanto del sistema, pero aumentamos las pérdidas de compuerta del MOSFET y las pérdidas por conmutación llegando a ser estas superiores a las pérdidas en conducción debido a la operación a altas frecuencias y carga baja.

Parte del consumo de esos elementos internos del LT1913 se puede observar mediante el dato de $I_{Quiescent=}1.2mA$ proporcionado por el fabricante (cuando el dispositivo no se encuentra conmutando), si ésta se compara con el consumo del µC tanto en bajo consumo (4µA) como en modo activo (550µA) se desprende que gran parte de la corriente se encuentra desaprovechada.

A continuación se muestra una ilustración en la que se puede observar para unos valores muy similares a los del proyecto que la eficiencia del convertidor se ve disminuida considerablemente para cargas bajas.

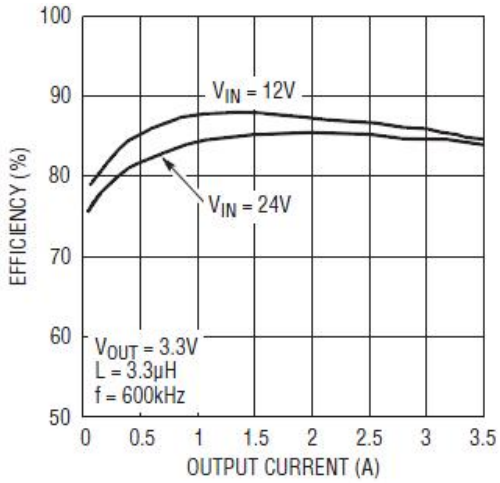


Ilustración 2: EFICIENCIA BUCK VS OUTPUT CURRENT

Para solucionar este problema de eficiencia se construyeron dos fuentes de alimentación, una mediante un convertidor buck para el módulo GSM y otra con un regulador lineal para el µC.

3.2.4 ENFOQUE Nº4

El circuito integrado LT1913 únicamente estaba disponible en encapsulado DFN lo que dificultaba la tarea de soldado, por lo que se sustituyó por el LT3680 prácticamente igual al anterior.

La configuración del módulo GSM se realizó mediante el envío de comandos AT a través del programa de Windows, HyperTerminal. La comunicación entre ambos se estableció a través de un cable conversor de USB a RS-232 ya que mi portátil no cuenta con conector DB9 y un *transceiver* para la adaptación de niveles de tensión entre el módulo GSM y el cable.

Ilustración 3: DIAGRAMA DE BLOQUES SISTEMA FINAL

3 ANÁLISIS DE SOLUCIONES Y SOLUCIÓN ADOPTADA

3.1 GSM

Este componente debía poseer las siguientes propiedades:

- Operación en las bandas de frecuencia en torno a 900MHz y 1800MHz (GSM-900, GSM-1800, EGSM 900), las cuales se utilizan en España y en Europa.
- Control vía comandos AT.
- Posibilidad de alimentarlo a 9V o menos.
- SMS en modo texto.
- Interfaz RS-232.
- Modo off.

3.1.1 OPCIÓN 1: MODEM GSM TIPO TERMINAL

TERMINAL	FABRICANTE	I EN MODO OFF	PRECIO
MC52IT	SIEMENS	480µA	111€
TC35IT	SIEMENS	550µA	107€
MTX-63I	MATRIX	500µA	108€
GT47	SONY ERICSSON	15mA	104€

Tabla 1: CONSUMOS Y PRECIOS MODEMS GSM TIPO TERMINAL

Estos tipos de modem se descartaron debido al elevado consumo de corriente y su elevado precio.

3.1.2 OPCIÓN 2: MÓDULO GSM

MÓDULO	FABRICANTE	I EN MODO OFF	PRECIO
ADH8066	SPARKFUN	100µA	37€
GSM0308	ENFORA	53µA	65€
MC55I	SIEMENS	50µA	62€
GM862	TELIT	26µA (Avg)	64€
HILO	SAGEMCOM	56µA	71€

Tabla 2: CONSUMO Y PRECIOS MÓDULOS GSM

El primer módulo que aparece en la tabla se desechó ya que la información disponible era confusa. El GSM0308 de enfora y los dos siguientes no se escogieron porque la soldadura del componente hubiera sido dificultosa debido al pitch de 0.4mm en el caso del primero y de la gran cantidad de pines en el caso de los últimos. Por esto y debido a que ofrece uno de los consumos menos elevados, muy buena información disponible en la página web y una aparente facilidad de uso se escogió el módulo HILO de SAGEMCOM. Tanto el módem GSM como el módulo GSM requieren de una tarjeta SIM.

3.2 MICROCONTROLADOR

En este caso se optó directamente por la familia de µC MSP430 de Texas Instruments porque es de muy bajo consumo y es la que se está utilizando en los proyectos de investigación de Tecnodiscap, grupo donde se ha realizado el proyecto.

Las cualidades que debía tener eran:

- Bajo consumo.
- 1 *timer* o RTC.
- 1 UART.
- Al menos 16kB de flash.

3.2.1 ALTERNATIVA 1: LA SERIE MSP430X1XX

Se encuentra en desuso y no la aconsejan para nuevos proyectos por lo que empecé a buscar en la MSP430F2xx Series.

3.2.2 ALTERNATIVA 2: LA FAMILIA MSP430AFE2XX

El más barato de esta familia que reunía todas las cualidades buscadas era el MSP430AFE251 pero finalmente se descartó por las siguientes razones:

- Los micros MSP430AFE252 y MSP430AFE253 que tenían asignadas las mismas funciones en sus pines que el posible elegido también poseían 16kB de flash, si por alguna razón se sobrepasaba esta capacidad se tenía que pasar a un µC que no las tenía lo cual era un problema.

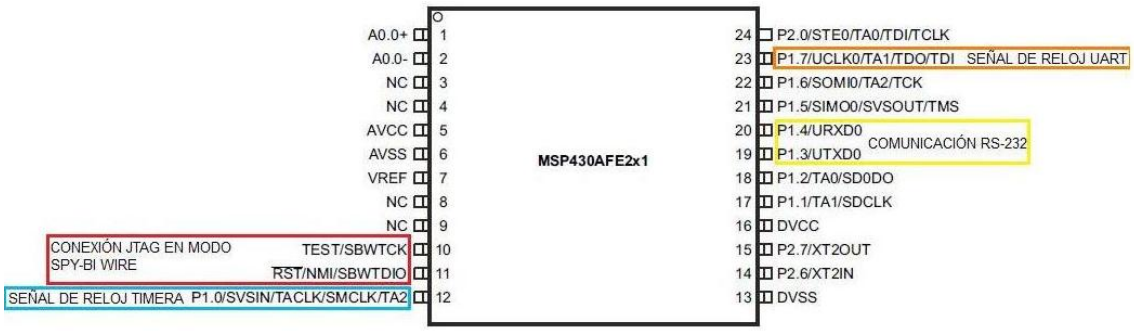


Ilustración 4: Ilustración 4: DESIGNACIÓN DE PINES EN EL MSP430AF251

- Nº de pines I/O escasas, solamente quedan siete pines libres.

3.2.3 ALTERNATIVA 3: EL MICROCONTROLADOR MSP430F2252

Se escogió porque era el más barato que cumplía con las características descritas anteriormente además de disponer de modelos compatibles pin a pin con mayor memoria para futuras ampliaciones.

3.3 FUENTE DE ALIMENTACIÓN MÓDULO GSM.

Por las razones ya expuestas anteriormente en el apartado 2.2 se escogió la estructura tipo buck y el integrado LT3680 para realizarla.

Las características buscadas eran:

- Alta eficiencia, característica de los convertidores dc-dc buck.
- Capacidad para proporcionar un consumo máximo de corriente de 2.2A. Picos de corriente de 1.8A más 0.4A de consumo de las entradas y salidas del módulo GSM.
- Posibilidad de alimentarlo a 9V.
- Tensión de salida de 3.5V.
- Bajo consumo de corriente cuando el dispositivo no se encuentre en funcionamiento.

- Rizado de la tensión de salida de 0.6V como mucho, ya que el rango de alimentación del módulo GSM es de 3.2 a 4.5V. Las fuentes conmutadas buck proporcionan un bajo rizado en la tensión de salida debido a su circuito L-C que actúa como un filtro paso bajo.

3.4 FUENTE DE ALIMENTACIÓN MICROCONTROLADOR

Para su elección se tuvo en cuenta principalmente que el regulador no desperdiciara demasiada corriente por su pin de GND, esto me permitía que la corriente suministrada por la batería fuera mínima mientras el µC se encontraba en modo de bajo consumo (la mayor parte del tiempo) y que no se incrementara en exceso cuando el µC estuviera en modo activo.

Busqué reguladores cuyo interruptor de paso fuera un transistor MOSFET, ya que en estos la corriente consumida se reduce a la requerida por otros componentes internos al ser controlados por tensión de puerta, maximizando de este modo la vida útil de la batería. En cambio en los reguladores con interruptores bipolares al formar parte la corriente de base (I_B) de la $I_{Quiescent}$ esta última aumenta con la corriente por la carga ya que $I_C = I_0 = \beta * I_B$.

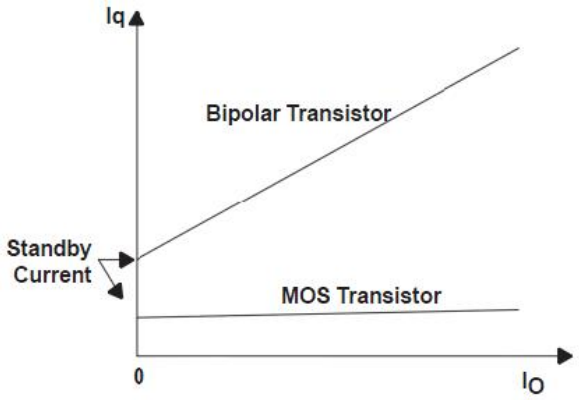


Ilustración 5: $I_{Quiescent}$ VS I_0

Además el regulador elegido debía contar con las siguientes propiedades:

- Posibilidad de alimentarlo en todo el rango de tensión de alimentación. $V_{IN} = 6 \div 10.5V$
- $I_{Lmáx} > 0{,}55mA$
- $V_{OUT} = 3.3$

Aunque una baja $V_{Dropout}$ suele ser conveniente para sistemas alimentados a través de una batería ya que permiten que esta pueda ser usada hasta el límite y que la potencia disipada en el componente sea mínima, el regulador seleccionado no tenía porque poseer esta propiedad debido a la gran diferencia entre la entrada y la salida deseada y a la mínima corriente demandada por la carga.

El regulador elegido fue el TPS71501DCKR ya que comparado con el TPS577001 poseía menor $I_{Quiescent}$ además de ser unos céntimos de euro más barato.

3.5 TARJETA Y CONECTOR SIM

3.5.1 TARJETA SIM

Tarjetas de 6 pines

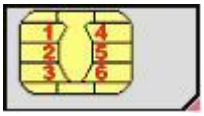


Ilustración 6: DISTRIBUCIÓN DE LOS PINES EN UNA TARJETA DE 6 CONTACTOS

Tarjetas de 8 pines:

La única diferencia entre esta y la de seis es el número de pines. Aunque tenga un número mayor de pines no aporta ninguna ventaja adicional, ya que C4 y C8 se encuentran reservados para un uso futuro. Se optó por la tarjeta SIM convencional de 6 pines.

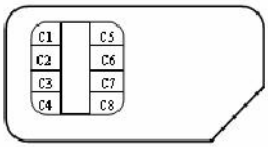


Ilustración 7: DISTRIBUCIÓN DE LOS PINES EN UNA TARJETA DE 8 CONTACTOS

3.5.2 CONECTOR SIM

Se podía utilizar tanto uno de seis como de ocho pines. En el de ocho pines se podía implementar un circuito detector de presencia de SIM, pero como a priori no encontré que esto me proporcionara ninguna ventaja me decanté por el de seis pines.

3.6 ELECTROVÁLVULA O VÁLVULAS DE SOLENOIDE

Son válvulas de accionamiento automático, aunque también cabe la posibilidad de accionarlas manualmente. Abren, cierran o gradúan el paso de un líquido o gas.

Al hacer circular corriente por el solenoide se crea un campo magnético que provoca el desplazamiento del émbolo. Al finalizar la aplicación de corriente el émbolo vuelve a su posición inicial por efecto de la gravedad, un resorte o la presión del fluido a controlar.

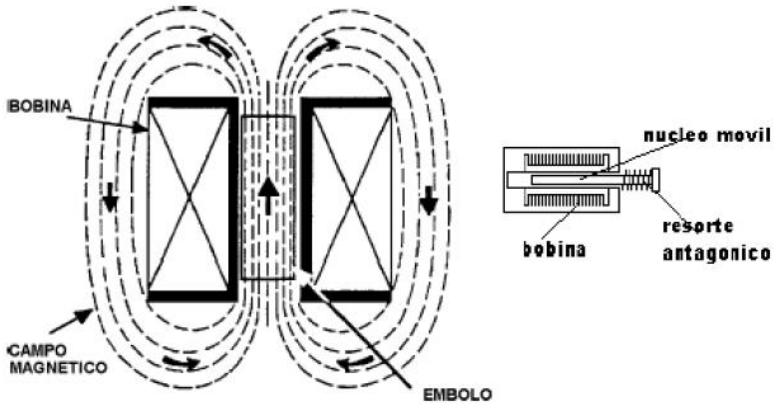


Ilustración 8: ACCIÓN DEL CAMPO MAGNÉTICO CREADO POR EL SOLENOIDE SOBRE EL ÉMBOLO

3.6.1 TIPOS DE SOLENOIDES

Solenoide estándar

Debido a la acción del campo magnético creado al circular corriente por el solenoide, el émbolo de la válvula es atraído hacia una pieza situada en la parte superior de la guía del émbolo.

Este tipo de válvulas necesitan de una aplicación permanente de energía para mantener el émbolo en la parte superior de la guía, aunque esta energía suele ser menor de la que se necesita para llevarlo hasta allí. Por esta razón se recomiendan para sistemas alimentados a través de la red eléctrica de energía.

A continuación se muestran unas tablas de diversos fabricantes con las características de solenoides de este tipo, es conveniente destacar la corriente de mantenimiento necesaria para que la válvula permanezca abierta:

- Dorot:

OPERATOR MARK	FREQUENCY (Hz)	CURRENT(AMP)	
		HOLDING	INRUSH
S-80-3-D 24VAC N.O	50	0.15	0.15
	60	0.14	0.14
S80-3-D 24VAC N.C	50	0.17	0.17
	60	0.16	0.16
S80-3 24VDC N.O		0.12	0.12
S80-3 12VDC N.O		0.28	0.28

Tabla 3: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR DOROT 3 WAY

OPERATOR MARK	FREQUENCY (Hz)	CURRENT(AMP)	
		HOLDING	INRUSH
S80-2 24VAC N.C	50	0.25	0.11
	60	0.22	0.095
S80-2 24VDC N.C		0.12	0.12
S80-2 12VDC N.C		0.28	0.28

Tabla 4: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR DOROT 2 WAY

- Bermad

ACTUATOR TYPE	CURRENT(AMP)		RESISTENCIA BOBINA (OHM)
	HOLDING	INRUSH	
S-390-2W-24VAC-R	0.125	0.25	37.5
S-390-2W-24VAC-D	0.13	0.13	
S-390-2W-12VDC	0.18	0.18	156
S-390-2W-12VDC	0.33	0.33	36

Tabla 5: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDA BERMAD 2WAY

ACTUATOR TYPE	CURRENT(AMP)		RESISTENCIA BOBINA (OHM)
	HOLDING	INRUSH	
S-390-3W-24VAC-D	0.13	0.13	37.5
S-390-3W-24VAC-D	0.20	0.20	
S-390-3W-24VAC-R	0.24	0.46	21
S-390-3W-24VDC	0.17	0.17	135
S-390-3W-12VDC	0.33	0.33	36

Tabla 6: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR BERMAD 3 WAY

Solenoide tipo *latch*

En la parte superior de la válvula hay un imán permanente, cuyo campo magnético no es suficiente para atraer hacia él por si solo al émbolo. Este imán si es capaz de retenerlo una vez que el émbolo ha sido llevado hasta él mediante la acción de un campo magnético creado a partir de la circulación de corriente a lo largo del solenoide (para esto hay que aplicar tensión durante un tiempo suficiente que suele ser de 15-20ms a 100ms). Esto permite que la válvula siga estando abierta una vez que la alimentación eléctrica cese, lo que hace que el ahorro de energía sea importante. La válvula permanecerá abierta hasta que debido a la aplicación de otro pulso de tensión que genere corriente en sentido inverso al anterior, el émbolo se separe del imán permanente al ser menor el campo magnético generado por este que el generado por el solenoide y consecuentemente la válvula se cierre.

Este tipo de solenoide es especialmente beneficioso cuando la válvula tiene que permanecer abierta durante largos periodos de tiempo o el sistema se encuentra alimentado a través de un batería, como es el caso.

Los pulsos no deben ser demasiado cortos porque la electroválvula puede no ponerse en marcha.

El voltaje aplicado no debe ser mayor que un 10%, ni tampoco menor de un 15% del voltaje nominal.

Este tipo de solenoides se pueden dividir a su vez en dos subgrupos dependiendo del número de cables utilizados para su control:

- Válvulas con solenoide tipo *latch* de tres terminales:

  Mediante la aplicación de tensión en el bobinado superior se produce la atracción del émbolo hacia la parte superior de la válvula, el cual permanecerá en esa posición aunque se retire dicha tensión. Cuando se aplique tensión en el bobinado inferior el émbolo regresará a su posición inicial. Si el solenoide puede ser alimentado con tensión alterna, a la entrada está provisto de un puente rectificador de cuatro diodos.

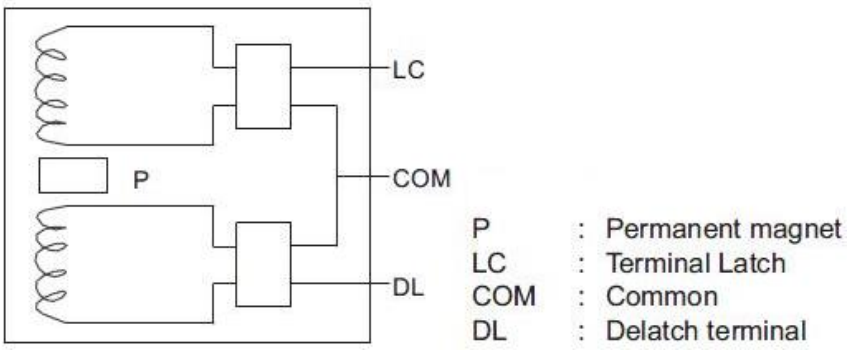


Ilustración 9: CIRCUITO SOLENOIDE LATCH DE 3 TERMINALES

A continuación se muestra una tabla que recoge los valores de las características eléctricas indicadas por fabricantes para este tipo de solenoide:

FABRICANTE	MODELO	TENSIÓN ALIMENTACIÓN (V)	R (Ω) BOBINA	L (mH) BOBINA	TIEMPO PULSO (ms)
BERMAD	S-985-2L-3W	12-50	4.2	-	20-100
DOROT	S-93-3	7.5-30	5.1	90	15-100
BACCARA	GEVA-60	12 ó 24	2.5 ó 5	-	30-100

Tabla 7: CARACTERÍSTICAS ELÉCTRICAS SOLENOIDE DE TRES TERMINALES

- Válvulas con solenoide tipo *latch* de dos terminales:

  Son solenoides con un único bobinado. La diferencia en cuanto a operación con el anterior es que para devolver al émbolo a su posición inicial se debe aplicar una tensión entre extremos del bobinado que genere una corriente de sentido inverso a la que lo llevó hasta la parte superior de la válvula.

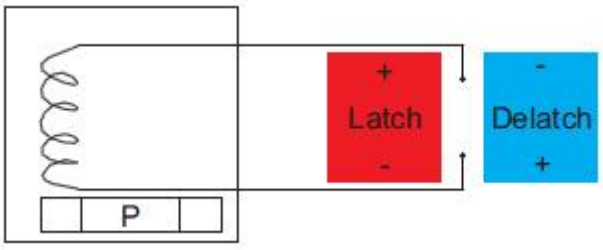


Ilustración 10: CIRCUITO SOLENOIDE LATCH DE DOS TERMINALES

A continuación se muestra una tabla que recoge los valores de las características eléctricas indicadas por fabricantes para este tipo de solenoide:

FABRICANTE	MODELO	TENSIÓN ALIMENTACIÓN (V)	R (Ω) BOBINA	L (mH) BOBINA	TIEMPO PULSO (ms)
BERMAD	S-392-2L-2W	6-20	6	90	20-100
BERMAD	S-402-2L-3W	9-40	6	90	15-100
BERMAD	S-982-2L-3W	12-50	4.2	-	30-100
DOROT	S-92-2 S-92-3	7.5-30	5.1	90	15-100
BACCARA	GEVA 60	12 ó 24	2.5 ó 5	-	30-100
BACCARA	GEVA 75	9 ó 12	4.2	-	25-50

Tabla 8: CARACTERÍSTICAS ELÉCTRICAS SOLENOIDE DE DOS TERMINALES

Se decidió no limitar el sistema en este aspecto, permitiendo la utilización tanto de solenoides de tres terminales como de dos.

Con lo cual el sistema deberá de ser capaz de operar con solenoides tipo *latch* ya sean de dos o tres terminales.

3.7 ETAPA DE ADAPTACIÓN Y CONTROL

Como el µC no es capaz de proporcionar los niveles de tensión y corriente requeridos por la electroválvula fue necesaria la implementación de un circuito que los adaptara.

Este circuito debía ser capaz de alimentar electroválvulas tanto con solenoides de tres hilos como de dos hilos, para su diseño se pensó en una estructura de puente completo. A continuación se muestran dos ilustraciones en las que se indica el conexionado de la carga a la salida del puente para las dos situaciones:

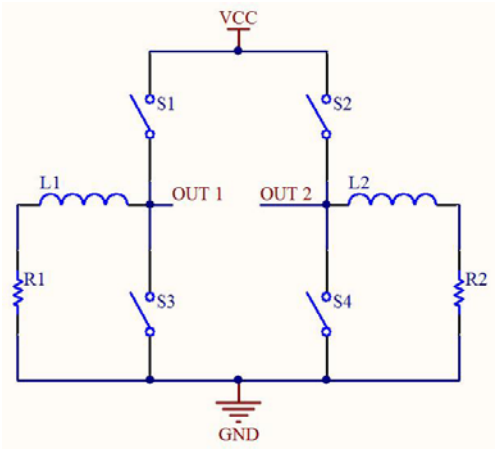


Ilustración 11: CONEXIONADO DEL SOLENOIDE DE TRES TERMINALES EN EL PUENTE EN H

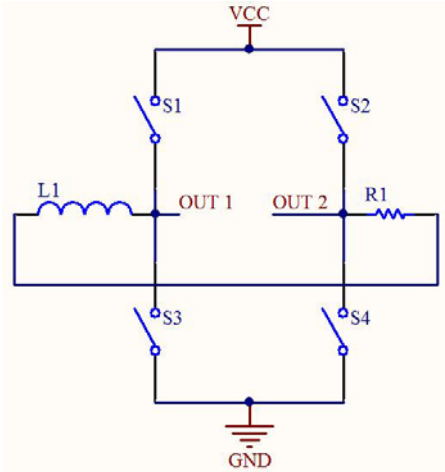


Ilustración 12: CONEXIONADO DEL SOLENOIDE DE DOS TERMINALES EN EL PUENTE EN H

El driver en puente en H debía de cumplir con los siguientes requisitos:

- Posibilidad de alimentación entre el rango de tensión de la batería: aproximadamente de 6-10.5V, tanto para la tensión de control como para la de la alimentación.
- Capacidad de suministro de picos de corriente de unos $\frac{(V_{BAT})_{máx}}{R} = \frac{10.5}{5.1} \approx 2A$. Este valor es el correspondiente a los solenoide S-92-2/S-92-3/S-93-3, ya que son estos los que tienen el mayor consumo de corriente de aquellos que se pueden alimentar a una tensión de 9V y soportan una tensión máxima de 10.5V. El valor de la inductancia no se ha tenido en cuenta ya que el driver debe ser capaz de soportar la corriente que circularía por este si por algún motivo el ancho del pulso aplicado superara 4τ.
- Buena compatibilidad con los niveles lógicos del µC. Es decir, $V_{IN(H)_{mín}} \leq 3.05$ y $V_{IN(L)_{máx}} \geq 0.25$.
- Bajo consumo en modo *standby*.

Ninguno de los *drivers* encontrados cumplía con todas las características descritas arriba. Esta tabla resume las que poseían y las que no:

	MPC17510EJ	TA7267BP	TA8428F/F6	TA7291P
TENSIÓN ALIMENTACIÓN	NO	SÍ	SÍ	SÍ
SUMINISTRO DE CORRIENTE	SÍ	SÍ	SÍ	SÍ
COMPATIBILIDAD NIVELES LÓGICOS	NO	SÍ	SÍ	NO
BAJO CONSUMO	SÍ	NO	NO	SÍ
COSTE (€)	2,76	1,86	2,62	1,29

Tabla 9: CARACTERÍSTICAS DRIVER PUENTE COMPLETO

El MCP17510EJ fue rechazado ya que no cumplía con dos de los requisitos y el TA8428F/F6 no se escogió ya que es mucho más caro en comparación con el TA726BP y el TA7291P.

La idea que se pensó para adaptar los niveles lógicos del TA7291P a los del µC fue la de colocar dos *picogates* (74HCT1GO4GW/T1 de NXP ó 74AHCT1G04SE-7 de diodes (0.39€)) entre estos:

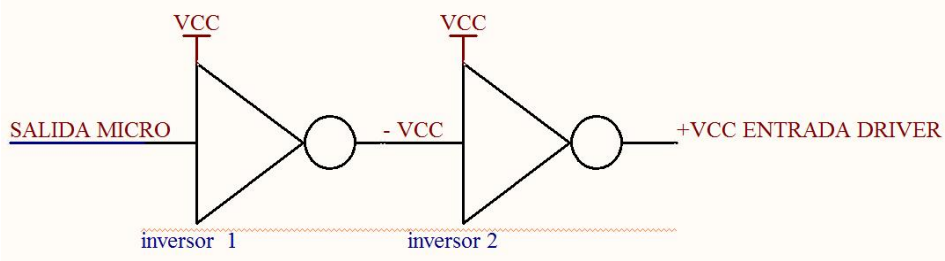


Ilustración 13: ADAPTACIÓN DE NIVELES ENTRE EL TA72919 Y EL µC

Como el nivel lógico superior para las entradas del driver era de 3.5V como mínimo, cabía la posibilidad de alimentar las *picogates* bien a través de la fuente del módulo GSM o bien subiendo la alimentación del µC de 3.3V a 3.5V. La primera opción se descartó ya que el la fuente del módulo GSM se encuentra apagada la mayor parte del tiempo.

La solución para el no bajo consumo del driver TA7267BP era la colocación de un interruptor que abriera la alimentación del driver cuando este estaba en desuso. Para llevar a cabo esta tarea se escogió el integrado QS6M3 (0.40€) en el que hay dos interruptores MOSFET uno de canal N y otro de canal P.

Finalmente me decanté por utilizar el driver TA7267BP con el integrado QS6M3, ya que esta opción es algo más barata además de reducir a la mitad el número de circuitos integrados a utilizar.

3.8 SOLENOIDE PARA LA ELECTROVÁLVULA

Las características que debe poseer el solenoide elegido son:

- Posibilidad de alimentarlo con una batería de 9V.
- Debe permitir la utilización de la batería en el más amplio rango de tensión posible.

Para estas dos primeras características hay que tener en cuenta que la tensión proporcionada por la batería variará aproximadamente desde un mínimo de 6V (cuando se encuentre descargada) hasta un máximo de 10.5V (cuando esté totalmente cargada).

- La corriente máxima en caso de la aplicación de un pulso superior o igual a 4τ no debe superar la corriente de pico máxima de 3A soportable por el driver.
- La corriente demandada por el solenoide durante la aplicación del pulso de tensión y, por tanto, la carga cedida por la batería debe ser la mínima posible.

En la siguiente ilustración se puede observar el comportamiento de la corriente en una carga RL dependiendo del tiempo de aplicación del pulso de tensión:

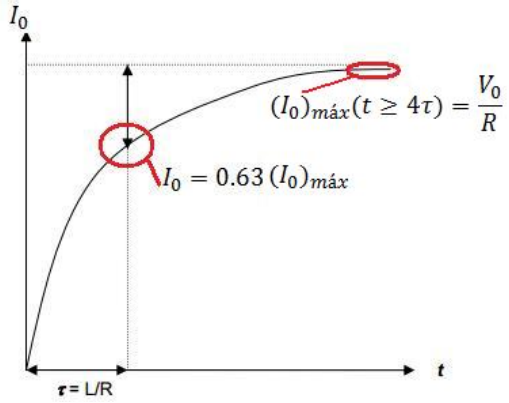


Ilustración 14: CORRIENTE VS TIEMPO CARGA RL

Por lo que según la tercera característica: $\frac{V_0}{R} \leq 3A$.

La corriente aumenta con el tiempo de aplicación del pulso de tensión, con lo cual se elige $t = 20ms$ que es el mínimo tiempo posible válido para la mayoría de los solenoides encontrados.

3.8.1 SOLENOIDES DE TRES TERMINALES

La siguiente ilustración muestra a qué hacen referencia los parámetros usados en las ecuaciones de este apartado:

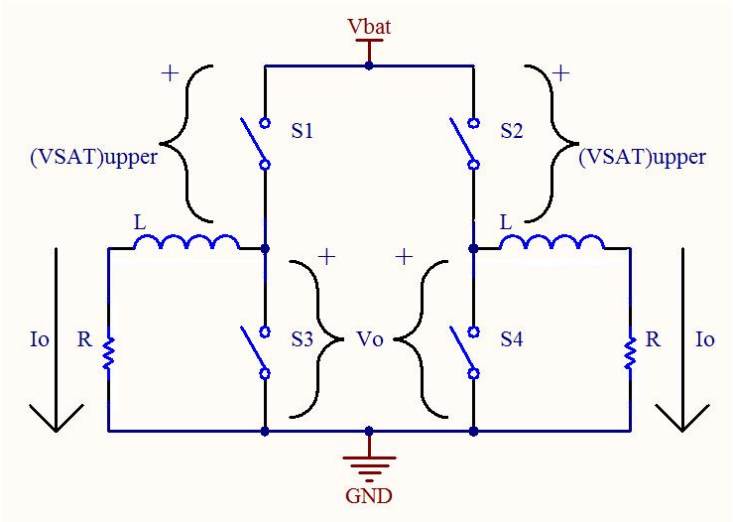


Ilustración 15: PARÁMETROS ETAPA DE SALIDA CON SOLENOIDE TRES TERMINALES

El mínimo valor de tensión necesario en la batería para la alimentación de la etapa de salida será:

$(V_{BAT})_{mín} = (V_0)_{mín} + (V_{SAT})_{upper}$, donde $(V_0)_{mín}$ es el mínimo valor posible de alimentación del solenoide y que como se ha dicho anteriormente no puede tener una variación mayor al -15% de la tensión nominal del solenoide. Es imposible conocer de forma exacta $(V_{SAT})_{upper}$ ya que el fabricante únicamente la indica para las condiciones detalladas en la siguiente tabla:

	$(V_{SAT})_{upper}$
$I_0 = 0.1A, V_{BAT} = 18V$	1.1V
$I_0 = 1A, V_{BAT} = 18V$	1.5V

Tabla 10: TENSIÓN DE SATURACIÓN TRANSISTORES LADO ALTO DRIVER

Aunque nos podemos hacer una idea de su valor aproximado si calculamos I_0 aplicando la ecuación de la corriente por una carga RL:

$$I_0(t) = \frac{V_0}{R} * \left(1 - e^{\frac{-t}{\tau}}\right)$$

La V_{SAT} de los transistores en el sistema para las corrientes de salida de 0.1 y 1A será menor que la indicada por el fabricante debido a que $V_{BAT} < 18V$.

El rango de tensión en el que podrá ser usado la batería vendrá dado por $(V_{BAT})_{mín}$ - $(V_{BAT})_{máx}$.

- S-93-3 DE DOROT: es el único de los mostrados en la tabla 7 que permite ser alimentado con una batería de 9V.

RANGO TENSIÓN ALIMENTACIÓN (V)	R-L (Ω-mH)	TIEMPO PULSO (ms)
7.5-30	5.1-90	15-100

Tabla 11: PROPIEDADES SOLENOIDE TRES TERMINALES DOROT

No va a haber ningún problema cuando V_{BAT} sea máximo ya que este solenoide admite una tensión máxima entre sus terminales de algo más de 30V.

Si por algún motivo se aplicara un pulso de tensión de duración igual o superior a 4τ, la corriente máxima demandada por este solenoide sería:

$(I_0)_{máx} = \frac{(V_0)_{máx}}{R} = \frac{(V_{BAT})_{máx} - (V_{SAT})_{upper}}{R}$, aún no teniendo en cuenta la tensión de saturación de los transistores:

$$(I_0)_{máx} = \frac{(V_{BAT})_{máx}}{R} = \frac{10.5}{5.1} = 2.05A < 3A$$

Cálculo de la corriente para conocer de forma aproximada $(V_{SAT})_{upper}$:

$$(I_O)_{mín} = \frac{(V_0)_{mín}}{R} * \left(1 - e^{\frac{-t}{\tau}}\right) = \frac{7.5 - 7.5 * 0.15}{5.1} * \left(1 - e^{\frac{-20ms}{17.64ms}}\right) = 0.84A$$

Entonces $(V_{SAT})_{upper} < 1.5V$ ya que $(I_O)_{mín} < 1A$ y $V_{BAT} < 18$, de aquí se desprende que:

$$(V_{BAT})_{mín} < (V_0)_{mín} + (V_{SAT})_{upper} = 7.5 - 7.5 * 0.15 + 1.5 = 7.875V$$

La batería se podrá utilizar desde un valor inferior a 7.875V hasta 10.5V.

3.8.2 SOLENOIDE DE DOS TERMINALES

La ilustración mostrada a continuación tiene exactamente la misma función que la anterior:

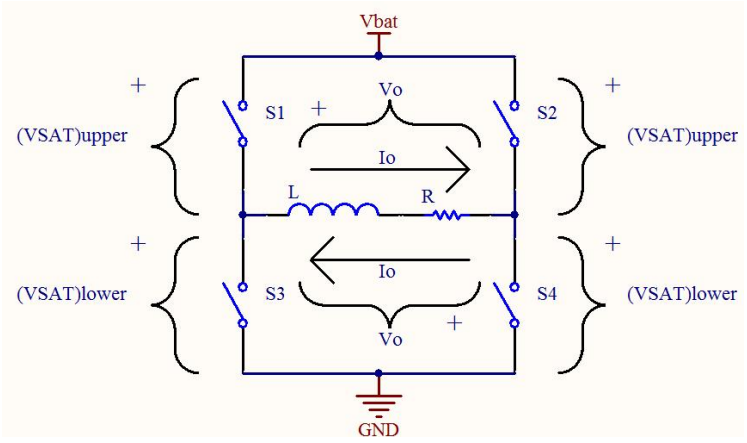


Ilustración 16: PARÁMETEROS ETAPA DE SALIDA CON SOLENOIDE DE DOS TERMINALES

Para saber la tensión mínima necesaria en la batería para alimentar a la etapa de salida se hará uso de la misma ecuación que en el caso anterior, solo que en este caso habrá que añadir un parámetro adicional $(V_{SAT})_{lower}$:

$$(V_{BAT})_{mín} = (V_0)_{mín} + (V_{SAT})_{upper} + (V_{SAT})_{lower}$$

Las tensiones de saturación $(V_{SAT})_{upper}$ y $(V_{SAT})_{lower}$ indicadas por el fabricante son las siguientes:

	$(V_{SAT})_{upper}$	$(V_{SAT})_{lower}$
$I_0 = 0.1A, V_{BAT} = 18V$	1.1V	1
$I_0 = 1A, V_{BAT} = 18V$	1.5V	1.4

Tabla 12: TENSIÓN DE SATURACIÓN TRANSISTORES DRIVER

- Solenoides de 6V de tensión nominal sin rango de tensión de alimentación: se descartaron porque no serían capaces de soportar la tensión entre sus extremos cuando la batería se encontrara totalmente cargada. Se recuerda que ésta no debe superar la tensión que corresponde a una variación de un +10% de su tensión nominal. El valor máximo de tensión de alimentación para este solenoide sería: $(V_0)_{máx} = 6 + 6 * 0.1 = 6.6V$
- S-392-2L-2W DE BERMAD

RANGO TENSIÓN ALIMENTACIÓN (V)	R-L (Ω-mH)	TIEMPO PULSO (ms)
6-20	6-90	20-100

Tabla 13: PROPIEDADES SOLENOIDE DOS TERMINALES BERMAD

No va a haber ningún problema cuando V_{BAT} sea máximo debido a que el valor superior del rango de tensión de alimentación es de 20V.

La corriente máxima demandada por este solenoide ante la aplicación de un pulso de tensión de duración $\geq 4\tau$ sería:

$(I_0)_{máx} = \frac{(V_0)_{máx}}{R} = \frac{(V_{BAT})_{máx} - (V_{SAT})_{upper}}{R}$, aún no teniendo en cuenta la tensión de saturación de los transistores:

$$(I_0)_{máx} = \frac{(V_{BAT})_{máx}}{R} = \frac{10.5}{6} = 1.75 < 3A$$

Cálculo de la corriente para conocer de forma aproximada $(V_{SAT})_{upper}$:

$$(I_O)_{mín} = \frac{(V_0)_{mín}}{R} * \left(1 - e^{\frac{-t}{\tau}}\right) = \frac{6 - 6 * 0.15}{6} * \left(1 - e^{\frac{-20ms}{15ms}}\right) = 0.62A$$

Entonces $(V_{SAT})_{upper} < 1.5V$ y $(V_{SAT})_{lower} < 1.4V$ ya que $(I_O)_{mín} < 1A$ y $V_{BAT} < 18$, por lo que se deduce que:

$$(V_{BAT})_{mín} < (V_0)_{mín} + (V_{SAT})_{upper} + (V_{SAT})_{lower} =$$
$$= 6 - 6 * 0.15 + 1.5 + 1.4 = 8V$$

El rango de tensión en el que se podrá utilizar la batería será de aproximadamente 8-10.5V.

- S-402-2L-3W DE BERMAD

RANGO TENSIÓN ALIMENTACIÓN (V)	R-L (Ω-mH)	TIEMPO PULSO (ms)
9-40	6-90	20-100

Tabla 14: PROPIEDADES SOLENOIDE DOS TERMINALES BERMAD

Si calculamos la corriente para tener una idea aproximada de las V_{SAT}:

$$(I_O)_{mín} = \frac{(V_0)_{mín}}{R} * \left(1 - e^{\frac{-t}{\tau}}\right) = \frac{9 - 9 * 0.15}{6} * \left(1 - e^{\frac{-20ms}{15ms}}\right) = 0.93A$$

Debido a este valor de corriente y a los mismos motivos que en el solenoide anterior $(V_{SAT})_{upper} < 1.5V$ y $(V_{SAT})_{lower} < 1.4V$, entonces:

$$(V_{BAT})_{mín} < (V_0)_{mín} + (V_{SAT})_{upper} + (V_{SAT})_{lower} =$$
$$= 9 - 9 * 0.15 + 1.5 + 1.4 = 10.55V$$

Aunque la alimentación de este solenoide fuera posible habría un muy bajo o nulo aprovechamiento de la batería, por lo que fue descartado.

- 9V LATCH DE RPE

Este solenoide puede ser usado en el rango que va de

$(V_O)_{mín} = 9 - 9 * 0.15 = 7.65V$ hasta $(V_O)_{máx} = 9 + 9 * 0.1 = 9.9V$.
Tiene un consumo de 500mA, y por tanto, la $(V_{SAT})_{total}$ estará entre:

$$(V_{SAT})_{total} > (V_{SAT})_{lower} + (V_{SAT})_{upper} = 1 + 1.1 = 2.1V$$

$$(V_{SAT})_{total} < (V_{SAT})_{lower} + (V_{SAT})_{upper} = 1.4 + 1.5 = 2.9V$$

De aquí se desprende un bajo rango de utilización de la batería, ya que:

$$(V_{BAT})_{mín} > (V_{SAT})_{total} + (V_O)_{mín} = 2.1 + 7.65 = 9.75V$$

3.8.3 CONCLUSIÓN

De los solenoides detallados en la tabla 7 y 8 se podrá hacer uso del S-93-3 DE DOROT y del S-392-2L-2W DE BERMAD

3.9 ETAPA DE ADAPTACIÓN DE LA TENSIÓN ENTRE LOS PINES TX DEL MICRO Y RX DEL MÓDULO

Como se puede observar en la siguiente tabla hay una desadaptación del nivel lógico alto entre ambos elementos, por tanto será necesaria una etapa que disminuya el nivel alto de tensión lógica del µC:

	V_{OUT} MICROCONTROLADOR		V_{IN} MÓDULO GSM	
	$V_{MÍN}$	$V_{MÁX}$	$V_{MÍN}$	$V_{MÁX}$
LOW LEVEL	0	0.25	0	0.4
HIGH LEVEL	3.05	3.3	2.4	3.2

Tabla 15: INCOMPATIBILIDAD DE LA TENSIÓN ENTRE TX DEL MICRO Y RX DEL MÓDULO

Además algunas de las etapas expuestas a continuación permiten proteger al módulo GSM frente a posibles reinyecciones de corriente cuando el módulo se encuentra apagado ya que pueden generar problemas durante el encendido del dispositivo.

3.9.1 SOLUCIÓN 1: DIODO ZENER

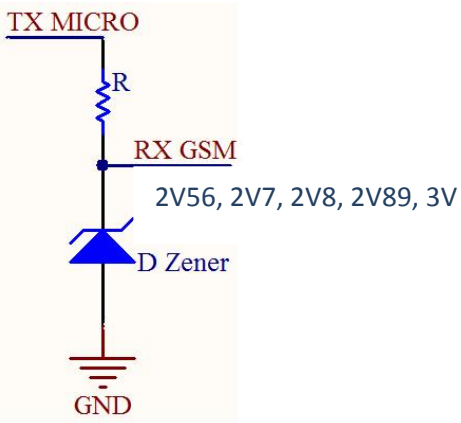


Ilustración 17: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN ZENER

Las tensiones zener han sido escogidas teniendo en cuenta que pueden sufrir una variación del ±5% respecto a su tensión nominal.

Esta opción se descartó debido a las siguientes razones:

- Habría que tener en cuenta que si se demanda una corriente mayor de 1.5mA al µC el nivel alto de tensión en un pin configurado como salida puede caer hasta Vcc-0.6, es decir, hasta 2.7V. Cosa que probablemente ocurriría ya que para tan bajas V_Z la I_z suele ser como poco de 5mA, siendo imposible la polarización de todos los diodos zener excepto la del 2V56.
- Esta estructura no ofrece protección ante reinyecciones de corriente en el caso de que el dispositivo esté apagado.

3.9.2 SOLUCIÓN 2: BUFFER

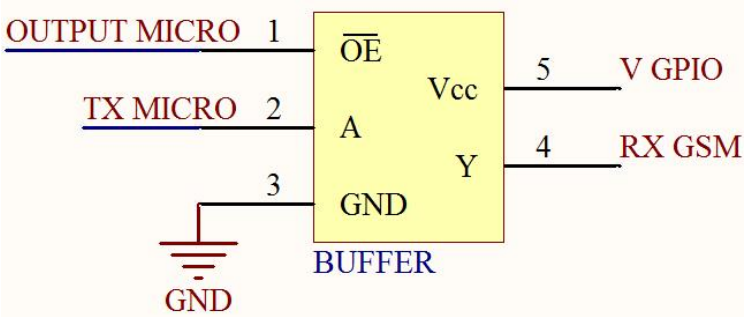


Ilustración 18: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN BUFFER

Mediante la puesta a uno de la salida del µC se evita la reinyección de corriente ya que el buffer pasa a estado de alta impedancia. Con un cero en la patilla uno el valor booleano presente en A se transmite a Y pero cambiando su nivel de tensión dependiendo de GND y Vcc.

V GPIO es una referencia de tensión generada desde el propio módulo GSM, tiene un valor típico de 2.8V aunque puede oscilar entre 2.65 y 2.95V. Como el valor de tensión del pin Y para bajas corrientes (como es el caso) ya sean entrantes o salientes de él es prácticamente igual a Vcc, entonces con esta estructura se consigue la adaptación de los niveles.

El consumo de corriente es reducido ya que ronda los 10µA cuando se encuentra en estado de alta impedancia y decenas de µA cuando se encuentra en funcionamiento.

Esta estructura presenta una única desventaja: la utilización de una salida adicional del µC.

3.9.3 SOLUCIÓN 3: DIODO SCHOTTKY

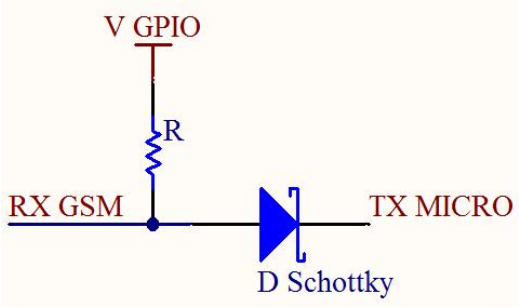


Ilustración 19: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN SCHOTTKY

El módulo queda protegido ante la reinyección de corriente ya que si no está encendido $V_{GPIO} = 0V$, por lo que, $V_{RX\ GSM} = 0$. Al ser la tensión en el ánodo del diodo de cero voltios nunca va a estar en directa y consecuentemente no va a permitir la salida de corriente del módulo GSM. No hay corriente que pueda ir del µC al módulo ya que el diodo no permite ese sentido de circulación de corriente.

Si el nivel presente en TX MICRO es alto el diodo se encuentra en inversa y $V_{RX\ GSM} = V_{GPIO} - V_R$, de donde se deduce que $V_{RX\ GSM} < V_{GPIO}$. Como el valor máximo de V_{GPIO} es de 2.95V la adaptación de tensión entre ambos elementos es correcta.

Si por el contrario el nivel lógico presente en TX MICRO es bajo el diodo se encuentra en directa y $V_{RX\ GSM} = V_{TX\ MICRO} + V_{FORWARD}$,

Esta fue la solución adoptada.

4 HARDWARE

Los esquemas completos del circuito así como los planos de pistas se pueden encontrar en los anexos número 1 y 2 respectivamente.

4.1 DIAGRAMA DE BLOQUES

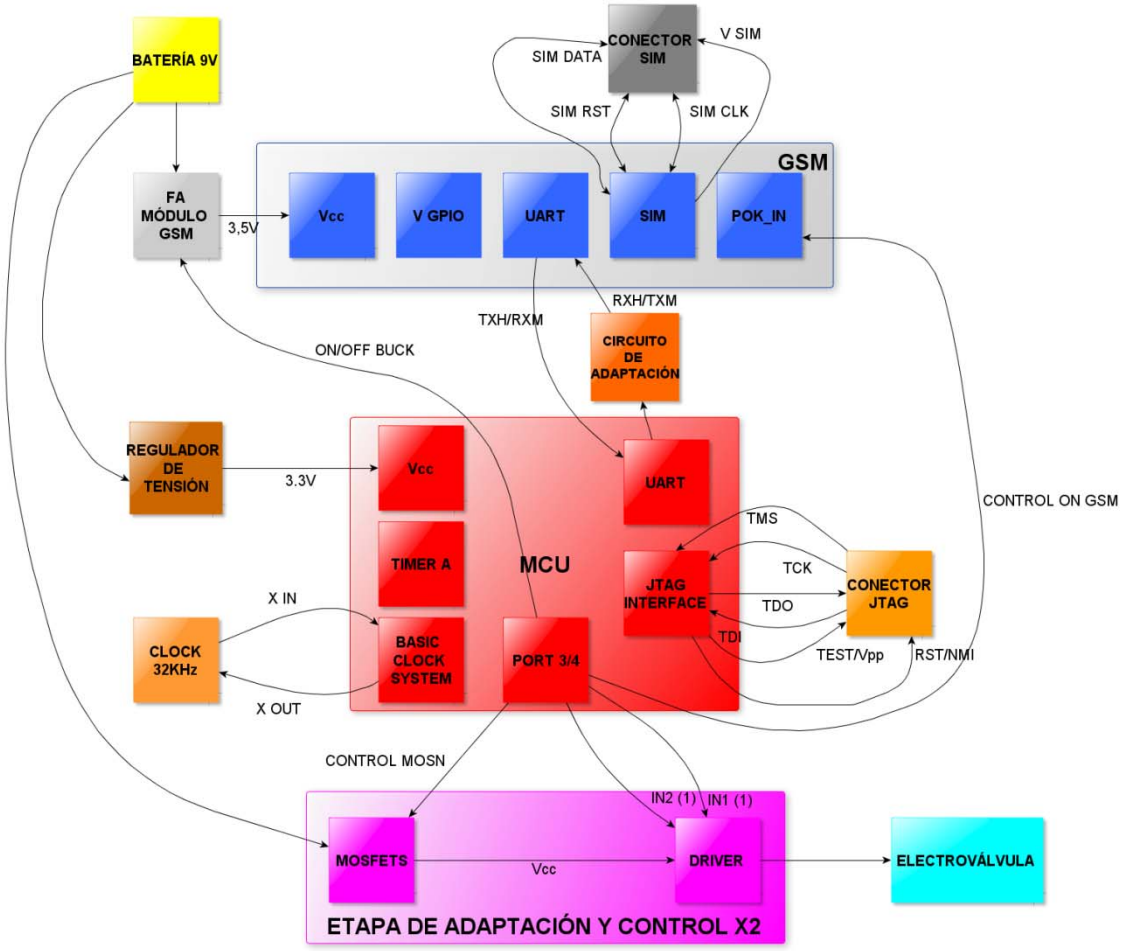


Ilustración 20: DIAGRAMA DE BLOQUES DEL CIRCUITO

4.2 FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

Se ha construido una fuente de alimentación conmutada de tipo buck basada en el integrado LT3680 de Linear Technology.

4.2.1 CARACTERÍSTICAS DEL LT3680

- V_{IN} puede ir de 3.6 a 36V.
- 3.5A de corriente máxima de salida.
- Frecuencia de conmutación ajustable de 200KHz a 2.4MHz.
- Tensión de salida de 0.79 a 30V.
- Se encenderá y apagará mediante el µC a través del pin RUN/SS, lo que permitirá reducir el consumo de corriente del componente hasta un máximo de 0.5µA.
- Control de la tensión de salida mediante el modo corriente.
- *Power good pin*. Se produce su puesta a uno cuando la tensión a la salida del dispositivo ha alcanzado el 91% de su valor final.
- Protección térmica.

4.2.2 DIAGRAMA DE BLOQUES DEL LT3680

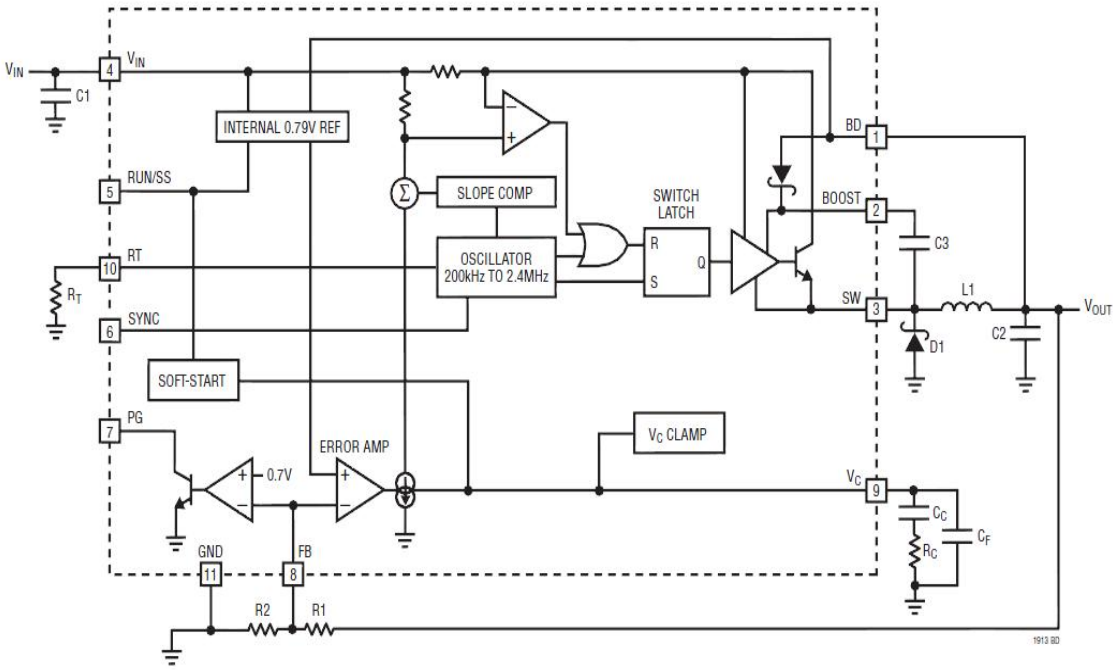


Ilustración 21: DIAGRAMA DE BLOQUES DEL INTEGRADO LT3680

4.2.3 CIRCUITO

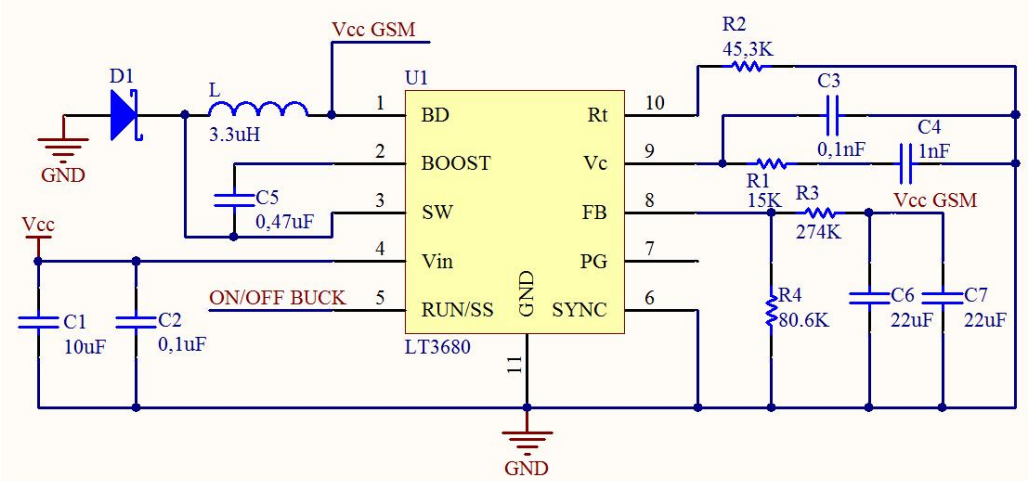


Ilustración 22: CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

4.2.4 COMPONENTES: VALOR Y FUNCIÓN

En este apartado se detallan los cálculos realizados para la asignación de valores a los componentes, su función y las características que se han tenido en cuenta para su elección en caso de ser necesario:

R_3 y R_4

Su función es fijar el valor de la tensión de salida, la ecuación que relaciona el valor de ésta con el de las resistencias es la siguiente:

$$R_3 = R_4 * \left(\frac{(V_{CC})_{GSM}}{0.79} - 1\right) => \frac{R_3}{R_4} = \frac{(V_{CC})_{GSM}}{0.79} - 1 = \frac{3.5}{0.79} - 1 = \frac{271}{79}$$

Ajustando los valores de resistencias a los disponibles en el mercado se establecieron los siguientes:

$R_3 = 274K\Omega$

$R_4 = 80.6K\Omega$

Bobina (L)

Para su elección fueron tenidos en cuenta los siguientes parámetros:

- La corriente eficaz máxima que soporta la bobina debía de ser mayor que la corriente demandada por la carga.
- La corriente de saturación de la bobina un 30% mayor que la corriente demandada por la carga. Mediante este porcentaje se asegura que la bobina funcione correctamente ante el pico de corriente que circula por ella:

$$I_{L_{Peak}} = \overline{I_{OUT(MÁX)}} + \frac{\Delta I_L}{2}$$

En las hojas características del LT3680 el fabricante indica que se parta de la siguiente ecuación para el cálculo del rizado:

$$\Delta I_l \leq 0.4\left(\overline{I_{OUT\,(MÁX)}}\right)$$

Sustituyendo esta expresión en la ecuación inmediatamente superior:

$$I_{L_{Peak}} = \overline{I_{OUT(MÁX)}} + \frac{0.4\overline{I_{OUT(MÁX)}}}{2} = \overline{I_{OUT(MÁX)}}(1 + 0.2) =$$

$$= 2.2(1 + 0.2) = 2.64A$$

- La DCR no debe ser mayor de 0.1Ω con el objetivo de mejorar el rendimiento del sistema.
- La frecuencia de resonancia debe estar por encima de 800KHz ya que será la frecuencia a la que conmutará el interruptor.
- El núcleo de la bobina debe ser adecuado para trabajar con altas frecuencias.
- La frecuencia para la que se ha testeado la bobina debe ser mayor que la frecuencia de conmutación.

Tiene como objetivo, junto a los condensadores de salida, filtrar la onda cuadrada generada por el LT3680.

Partiendo de la ecuación indicada por el fabricante para el cálculo del rizado de corriente:

$$\Delta I_l \leq 0.4\left(I_{OUT\,(MÁX)}\right) = 0.4 * 2.2A = 0.88A$$

Conociendo que en un convertidor dc-dc buck, teniendo en cuenta la caída de tensión en el diodo de recirculación de corriente, el rizado de corriente por la bobina viene dado por la ecuación:

$$\Delta I_l = \left(\frac{(V_{CC})_{GSM} + V_D}{F_{SW} * L}\right) * (1 - D)$$

Donde V_D corresponde a la caída de tensión del diodo, F_{SW} a la frecuencia de conmutación del interruptor y D al ciclo de trabajo.

Despejando el valor de L:

$$L = \left(\frac{(V_{CC})_{GSM} + V_D}{F_{SW} \times \Delta I_L}\right) \times \left(1 - \frac{(V_{CC})_{GSM} + V_D}{V_{IN(MÁX)}}\right)$$

Sustituyendo valores en la anterior ecuación, eligiendo una F_{SW} de 800KHz y siendo el valor de V_D aproximadamente 0.4V:

$$L = \left(\frac{3.5 + 0.4}{800KHz \times 0.88A}\right) \times \left(1 - \frac{3.5 + 0.4}{9}\right) = 3.14uH$$

Como para cumplir la condición de rizado el valor de la bobina puede ser mayor al calculado, entonces, L=3.3µH.

Condensador de bypass (C_1)

El convertidor requiere de elevadas corrientes pulsantes con rápidos tiempos de subida y de bajada, estas corrientes producen caídas de tensión en la línea de alimentación debido a su elevado valor y su alto contenido de componentes armónicas de alta frecuencia. Mediante el condensador se reducirá el rizado causado por este motivo ya que:

- Almacenará la carga y la liberará cuando se produzcan demandas instantáneas de ésta, evitando que esta corriente tenga que recorrer la inductancia de la línea de alimentación. De este modo el condensador desacopla la fuente de alimentación principal de la del circuito integrado.
- Proporcionará un camino de baja impedancia para las componentes armónicas de alta frecuencia.

Para conocer cómo proporcionar un camino de baja impedancia a través de un condensador de bypass para el ruido de alta frecuencia en la línea de alimentación es necesario el estudio del circuito equivalente del condensador y de una pista de circuito impreso:

- Circuito equivalente del condensador:

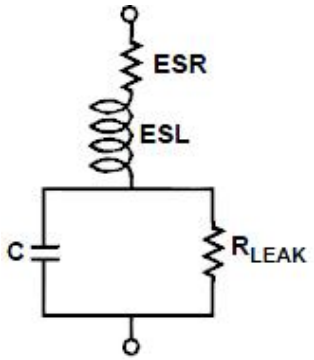


Ilustración 23: CIRCUITO EQUIVALENTE DEL CONDENSADOR

Este circuito tiene la dependencia impedancia-frecuencia mostrada a continuación, donde el valor de la frecuencia de resonancia viene determinada por $\frac{1}{2\times\Pi} \times \sqrt{\frac{1}{ESL\times C}}$:

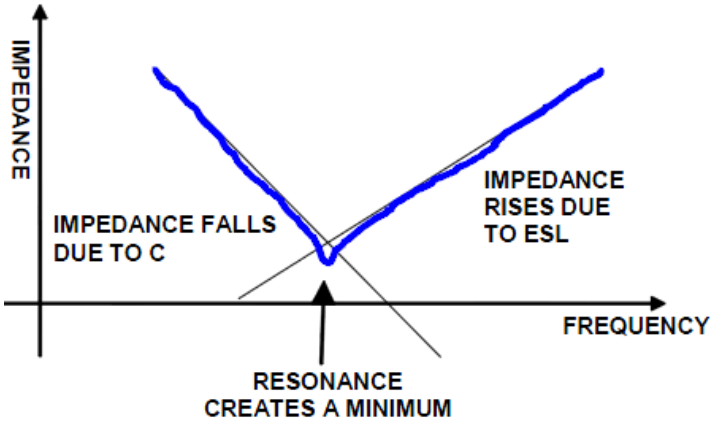


FIGURE 8. IMPEDANCE OF AN ACTUAL CAPACITOR (NON-IDEAL)

Ilustración 24: IMPEDANCIA DE UN CONDENSADOR VS FRECUENCIA

A menor valor de ESL el mínimo creado por la frecuencia de resonancia se encontrará más hacia la derecha, por tanto, se obtendrá un mejor filtrado del ruido producido por las componentes de alta frecuencia. Consecuentemente con esto se escogió un condensador cerámico ya que poseen un valor de ESL reducido.

Dentro de los condensadores cerámicos me decanté por el de tipo X5R ya que sufren un cambio menor en la capacidad con la variación de temperatura (±15%) que los Y5V. La siguiente gráfica muestra esta variación para estos últimos.

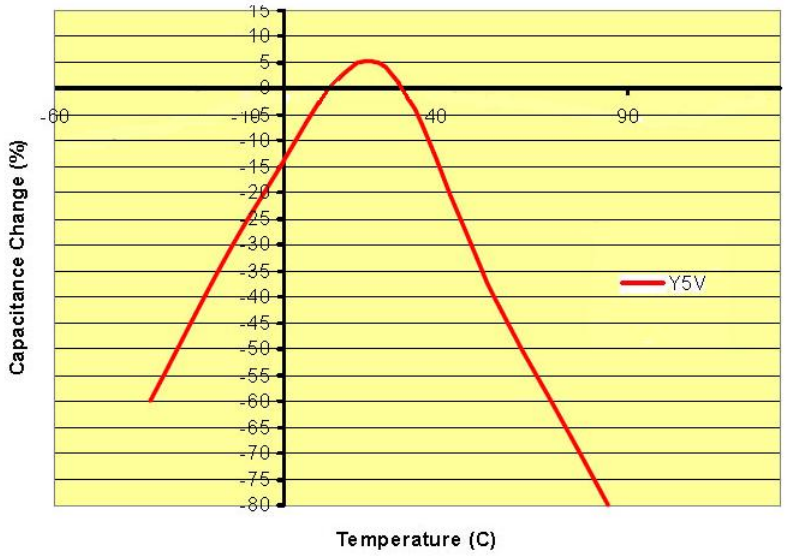


Ilustración 25: VARIACIÓN DE LA CAPACIDAD DE UN CONDENSADOR YV5 VS TEMPERATURA

- Circuito equivalente de una pista de circuito impreso:

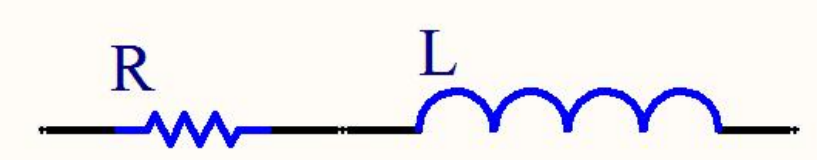


Ilustración 26: CIRCUITO EQUIVALENTE DE UNA PISTA DE CIRCUITO IMPRESO

Cuanto más alejado esté el condensador del circuito integrado más inductancia se añade al bucle de corriente formado por el condensador y el circuito integrado ejerciendo un efecto contrario al que se consigue con un condensador de baja ESL, es decir, es como si le añadiéramos ESL al condensador y su frecuencia de resonancia disminuyera.

Para reducir este efecto se debe colocar el condensador lo más cerca posible del circuito integrado reduciendo al máximo posible el tamaño del bucle de corriente, construir las pistas de masa por las que circulan las componentes de alta frecuencia anchas y cortas y unir el pin de GND directamente a el plano de masa mediante una vía.

Este condensador también reducirá las perturbaciones que pueden ser introducidas desde la fuente de alimentación o circuitos integrados cercanos.

Condensador de bypass (C_2)

Al ser C_1 cerámico ya filtra bastante bien las componentes de alta frecuencia, pero el filtrado puede ser mejorado tal y como se demuestra a continuación si en paralelo con C_1 se coloca un condensador de menor capacidad y tamaño:

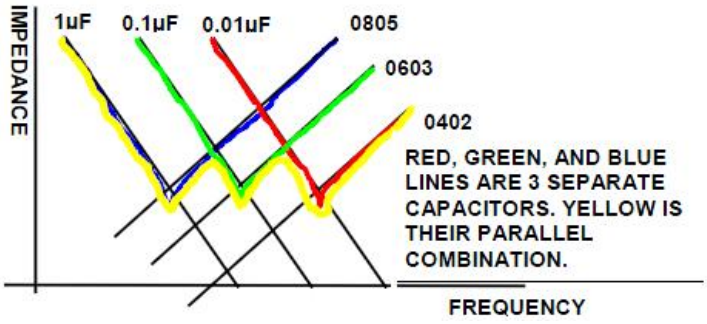


Ilustración 27: IMPEDANCIA DE TRES CONDENSADORES EN PARALELO CON ENCAPSULADO SMD ESCALADO VS FRECUENCIA

Este condensador se colocará más cercano al circuito integrado que C_1 ya que por él circularánlas señales AC de mayor frecuencia.

Condensadores de salida (C_6 y C_7)

Junto con la bobina se encarga de filtrar la onda de tensión cuadrada generada por el LT3680 además de satisfacer demandas de corriente transitorias mediante la carga almacenada en ellos y proporcionar estabilidad al bucle de control.

Para que haya un bajo nivel de rizado en la salida la impedancia del condensador a la frecuencia de conmutación debe ser reducida además de poseer una baja ESR. Estas dos cualidades las presentan los condensadores cerámicos y los electrolíticos y de tántalo de alto rendimiento pero estos últimos son más caros por lo que los cerámicos fueron los escogidos.

Utilizando la fórmula proporcionada por el fabricante para el cálculo del condensador de salida, su valor queda:

$$C_{OUT} = \frac{100}{(V_{CC})_{GSM} * F_{SW}} = \frac{100}{3.5 * 800KHz} = 35.71uF$$

Como a mayor condensador mejor filtrado de la tensión de salida $\left(\Delta V_0 = \frac{I \times T}{C}\right)$ y mejor respuesta transitoria, se colocaron dos condensadores de 22µF en paralelo (eran más baratos que uno de 44µF). Con lo que la capacidad equivalente a la salida asciende a 44µF.

R_2:

Se encarga de programar la frecuencia de conmutación del oscilador, el cual forma parte del circuito que genera la onda PWM que excita al elemento de conmutación.

Se escoge mediante una tabla proporcionada por el fabricante en la que se asigna para cada valor de frecuencia de conmutación un valor de resistencia. Para una frecuencia de 800KHz el valor de resistencia fijado por el fabricante es de 45.3KΩ.

Elementos unidos a V_C:

R_1 y C_4 forman parte del bucle de compensación de frecuencia. Estos componentes son difíciles de calcular, en la hoja de características del LT3680 se recomienda partir de los valores asignados a estos componentes en el circuito ejemplo que mejor se adapte a tu situación e ir variándolos hasta conseguir el funcionamiento óptimo. Para evitar esta situación se escogió la misma frecuencia que en el diseño presente en las hojas de características del módulo GSM, y por tanto los mismos valores de R_1 y C_4.

C_3 sirve para filtrar el ruido a la frecuencia de conmutación. El valor de este condensador se escogió conforme a las hojas características del módulo GSM.

C_5

Se recomienda un valor de 0,47µF para la tensión de salida seleccionada.

Diodo de recirculación de corriente (D_1)

Este componente debía ser capaz de soportar tensiones inversas mayores que la tensión de alimentación del LT3680.

Teniendo en cuenta que por el diodo únicamente circula corriente durante t_{off}, la corriente media por éste vendrá dada por la siguiente ecuación:

$$\overline{I_D} = \overline{I_{OUT}} - \overline{I_{t_{on}}} = \overline{I_{OUT}} - \overline{I_{OUT}} * D = \overline{I_{OUT}} * (1 - D) =$$

$$= \overline{I_{OUT}} * \left(1 - \frac{(V_{CC})_{GSM}}{V_{IN}}\right) = \overline{I_{OUT}} * \left(\frac{V_{IN} - (V_{CC})_{GSM}}{V_{IN}}\right)$$

Sustituyendo valores la corriente media por el diodo queda:

$$\overline{I_D} = 2.2 \times \frac{(9 - 3.5)}{9} = 1.34A$$

La única razón para considerar la elección de un diodo capaz de soportar corrientes mayores que la calculada es una posible situación de cortocircuito. La corriente media que circularía por el diodo en esa situación sería:

$$\overline{I_D} = \overline{I_{OUT}} * \left(\frac{V_{IN} - (V_{CC})_{GSM}}{V_{IN}}\right) = \left(I_{LIM} - \frac{\Delta I_L}{2}\right) * \left(\frac{V_{IN} - (V_{CC})_{GSM}}{V_{IN}}\right) =$$

$$= (5.5 - 0.44) * \frac{(9 - 3.5)}{9} = 3.09A$$

I_{LIM} es la máxima corriente entregable por el convertidor.

4.2.5 ONDA DE TENSIÓN DE SALIDA OBTENIDA EN EL LABORATORIO

La onda de tensión fue obtenida para una $R_L \approx 2\Omega$ y con lo cual para una corriente media de salida de:

$$\overline{I_{OUT}} = \frac{\overline{(V_{CC})_{GSM}}}{R_L} = \frac{[(V_{CC})_{GSM}]_{\text{máx}} - \frac{\Delta V_0}{2}}{2} = \frac{3.44\text{V} - 30mV}{2} = 1.7A$$

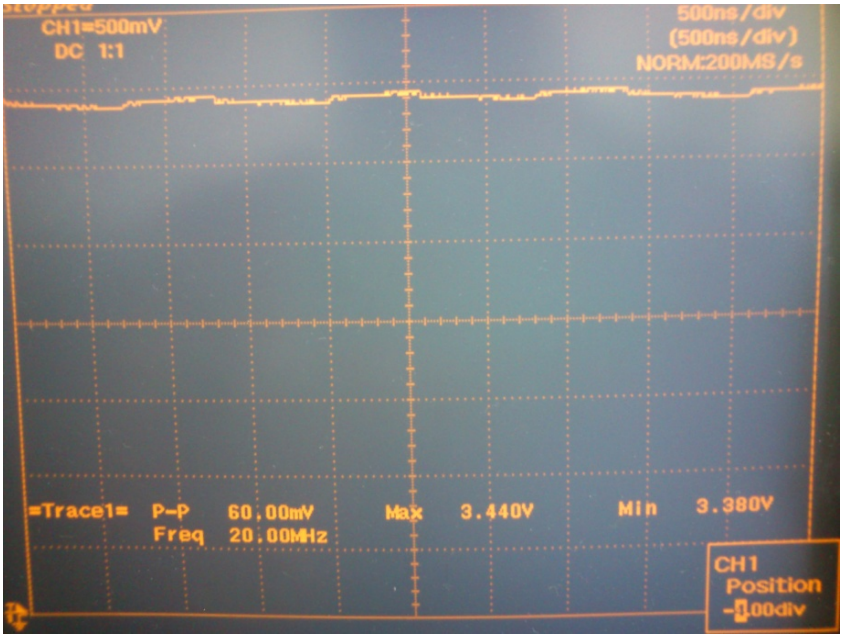


Ilustración 28: ONDA DE TENSIÓN OBTENIDA EXPERIMENTALMENTE A LA SALIDA DEL CONVERTIDOR

Como se puede observar la onda obtenida oscila entre los 3.38 y 3.44V. El módulo GSM admite una tensión de alimentación comprendida entre los 3.2 y 4.5V, por lo que se cumple con el requerimiento.

4.3 MÓDULO GSM. SAGEM HILO MODULE

4.3.1 CARACTERÍSTICAS

- Alimentación de 3.2 a 4.5V. La tensión nominal es de 3.7V.
- Consumo de corriente típico de 35µA en modo off.
- El mínimo valor de consumo de corriente en *standby* es el que se produce en modo de recepción discontinua DRX9, un valor típico de 1.3mA para temperaturas de 25°C. Ya que este consumo es relativamente alto se decidió apagar el módulo y encenderlo cada cierto tiempo para conocer si se había producido la recepción de algún SMS.
- Soporta las bandas de frecuencia GSM 850, EGSM 900, DCS 1800 y PCS 1900.
- Cuando el módulo se encuentra comunicando en la banda de frecuencia GSM 900 el consumo se encuentra entre 220 y 230mA, pero hay picos de corriente de 1.8A y 550us de duración cada 4.6ms
- Las tarjetas SIM que soporta son las de 3 ó 1.8V. Las señales para su manejo se encuentran disponibles en el conector de 40 pines.
- UART: posibilidad de control de flujo tanto por software como por hardware, velocidad de transmisión de hasta 115.2kbps, *auto-bauding*.
- *Power on pin*. Se controlará el encendido y apagado del modulo mediante el µC a través de este pin.
- Entradas y salidas de propósito general: 5 GPIO + 1 ADC. Estas entradas y salidas pueden ser usadas por ejemplo para detectar la presencia de antena y/o de SIM, para hacerlas conmutar una vez se ha alcanzado una temperatura programada, para indicar mediante un diodo LED el estado del dispositivo…
- Generador de ondas PWM.
- Posibilidad de conexión de altavoz y micrófono.
- RTC. Se mantiene en funcionamiento aunque el módulo se encuentre desconectado gracias a su alimentación a través de VBACKUP.

4.3.2 DIAGRAMA DE BLOQUES DEL MÓDULO GSM

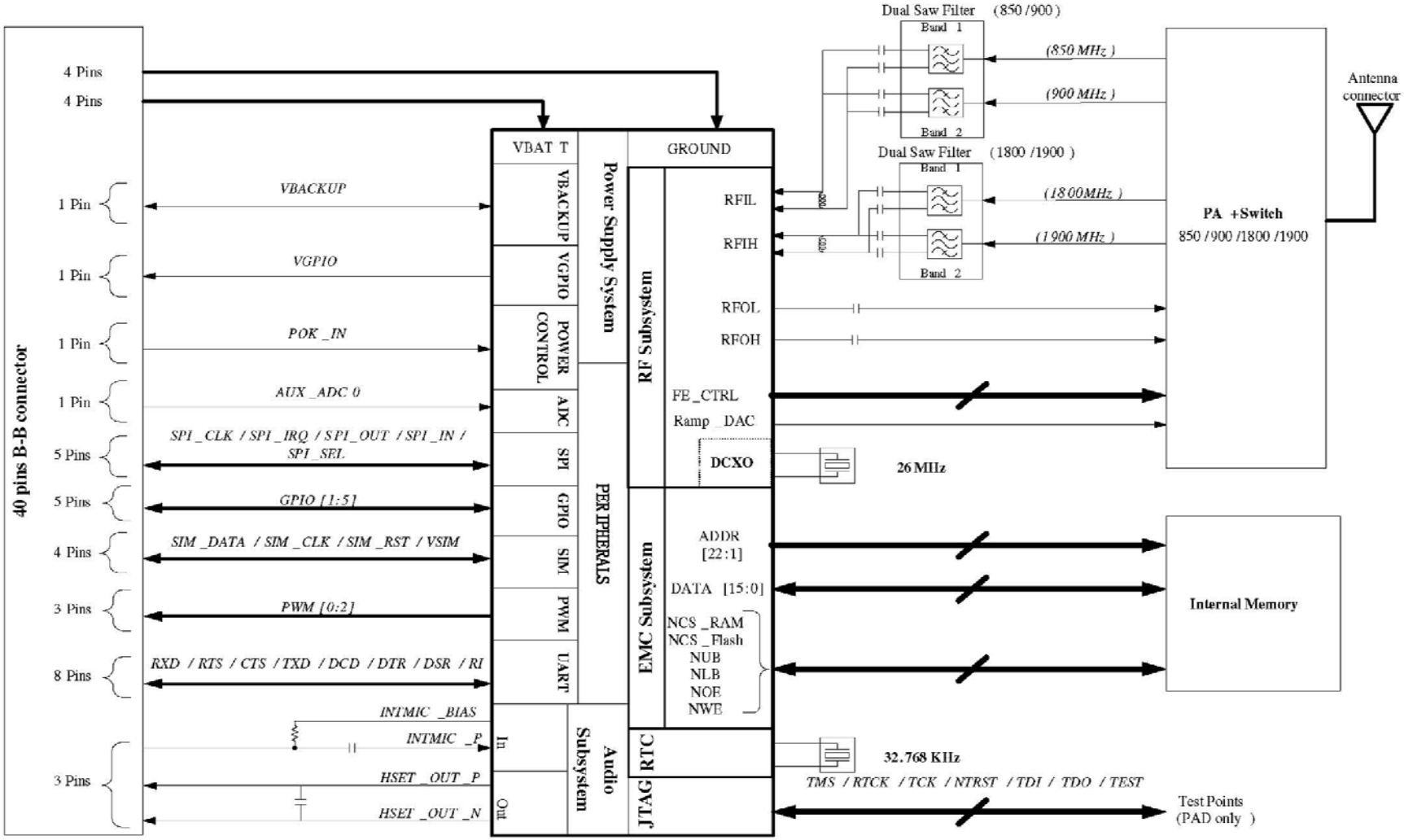


Ilustración 29: DIAGRAMA DE BLOQUES DEL MÓDULO GSM

4.3.3 CIRUITO DEL CONECTOR DEL MÓDULO GSM

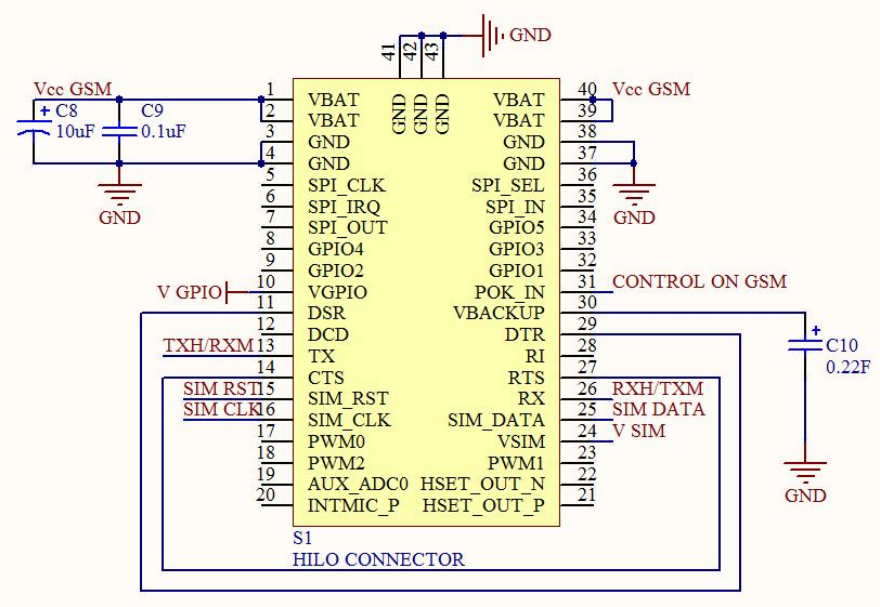


Ilustración 30: CIRCUITO DEL CONECTOR DEL MODULO GSM

- Los pines 41,42 y 43 representan la conexión de la carcasa metálica del módulo a masa.
- C_8/ C_9: condensadores de bypass.
- C_{10}: colocado entre VBACKUP y masa permite mantener el RTC activo aunque se retire la alimentación al módulo. Si $(V_{CC})_{GSM}$ es mayor que VBACKUP el RTC es alimentado a través de la primera, si sucede lo contrario es VBACKUP la que se va a encargar de alimentar el RTC. De este modo se evita la implementación de un RTC en el µC.

4.4 FUENTE DE ALIMENTACIÓN DEL µC

Para construirla se utilizó el regulador lineal de bajo consumo TPS71501DCKR de Texas Instruments.

4.4.1 CARACTERÍSTICAS DEL TPS71501DCKR

- Voltaje de entrada máximo de 24V.
- Voltaje de salida de 1.2 a 15 voltios.
- Corriente máxima de salida de 50mA.
- Corriente de reposo típica de 3.2µA, manteniéndose estable para todo el rango de corriente por la carga.
- Regulador con baja caída de tensión de entrada a salida.

- Estable con un condensador de capacidad mayor o igual a 0.47µF.

4.4.2 DIAGRAMA DE BLOQUES DEL TPS7150DCKR

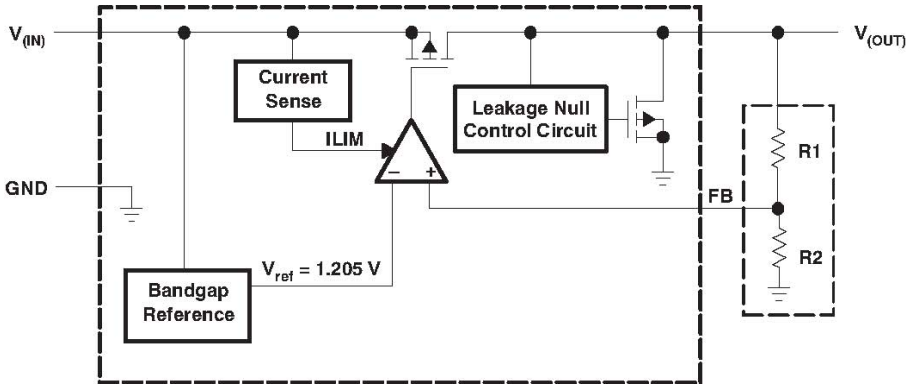


Ilustración 31: DIAGRAMA DE BLOQUES DEL REGULADOR LINEAL

4.4.3 CIRCUITO

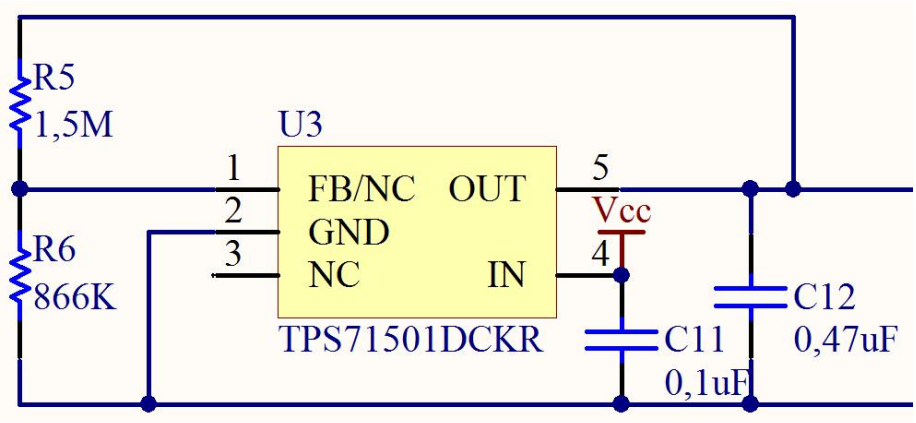


Ilustración 32: CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL µC

4.4.4 COMPONENTES: VALOR Y FUNCIÓN

R_5 y R_6

Mediante este divisor de tensión se programa la tensión deseada en la salida del regulador lineal. La siguiente ecuación, común para este tipo de reguladores, relaciona la tensión de salida del regulador ($(V_{CC})_{micro}$) con el valor de estas resistencias:

$$(V_{CC})_{micro} = V_{REF} * \left(1 + \frac{R_5}{R_6}\right) => \frac{R_5}{R_6} = \frac{V_{OUT}}{V_{REF}} - 1 => R_5 = \left(\frac{V_{OUT}}{V_{REF}} - 1\right) * R_6$$

Dónde V_{REF} es la referencia interna de tensión, la cual tiene un valor de 1.205V y para V_{OUT} nos interesa un valor de 3.3V.

El fabricante aconseja que la corriente por el divisor resistivo sea de 1.5µA, por tanto:

$$R_6 = \frac{V_{REF}}{1.5\mu A} = 803.33K\Omega$$

Sustituyendo este valor en la primera ecuación:

$$R_5 = \left(\frac{3.3}{1.205} - 1\right) * 803.33 = 1.39M\Omega$$

Ajustando los valores de resistencias a los disponibles en el mercado, se estableció $R_6 = 866K\Omega$ y $R_5 = 1.5M\Omega$.

C_{11}

Condensador de bypass.

C_{12}

Este condensador es necesario para estabilizar el bucle de control interno del regulador.

4.5 MICROCONTROLADOR MSP430F2252

4.5.1 CARACTERÍSTICAS

- Tensión de alimentación entre 1.8 y 3.6V.
- Bajo consumo.
- 2 *timers*.
- UART con posibilidad de detección automática de la velocidad de transmisión.
- Conversor A/D.
- Dos amplificadores operacionales configurables.
- Cinco modos de bajo consumo. Ordenados de mayor a menor consumo son LPM0, LPM1, LPM2, LPM3 y LPM4.
- Cuatro fuentes de reloj a partir de las que se obtienen tres señales de reloj para la operación de la CPU y periféricos. Esto permite elegir la mínima frecuencia de trabajo posible para cada periférico haciendo que el consumo sea mínimo.
- *Brownout reset*. Señal generada que resetea el µC cuando la tensión de alimentación del dispositivo es aplicada o retirada.
- 16KB de flash y 512 bytes de RAM.

4.5.2 DIAGRAMA DE BLOQUES

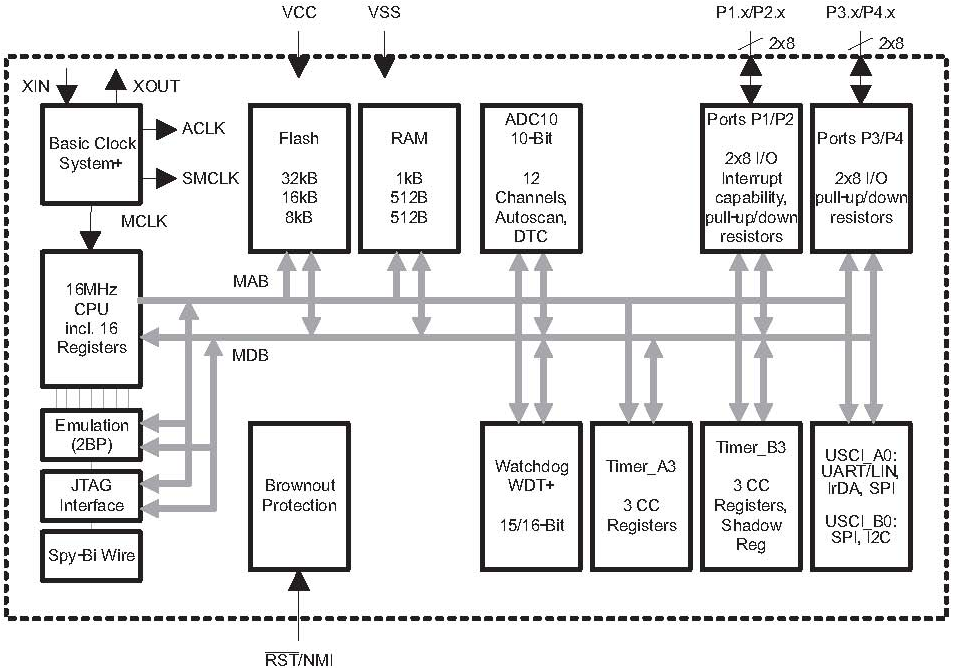


Ilustración 33: DIAGRAMA DE BLOQUES DEL µC

4.5.3 CIRCUITO

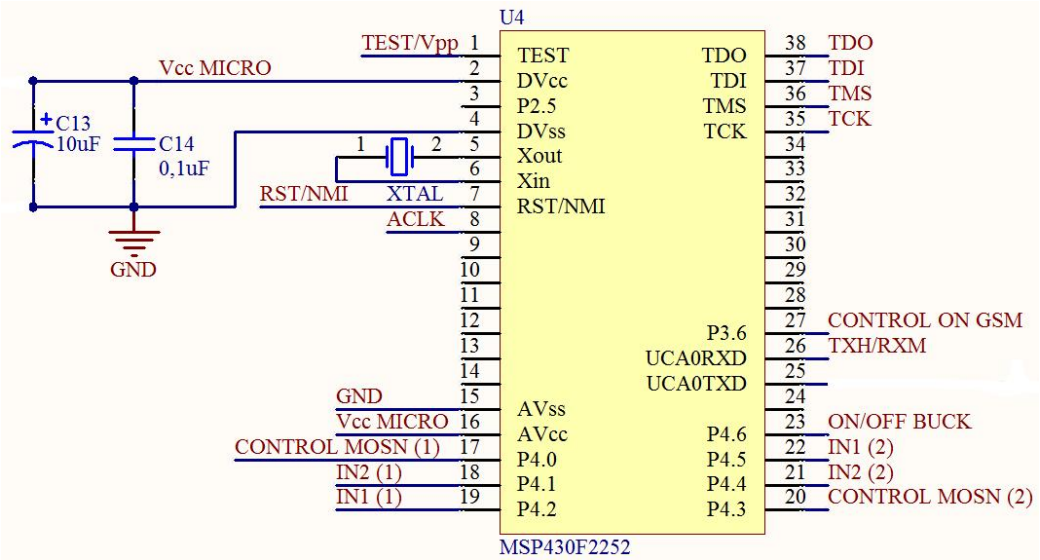


Ilustración 34: CIRCUITO DEL µC

4.5.4 COMPONENTES: VALOR Y FUNCIÓN

C_{13} y C_{14}

Condensadores de bypass.

Cristal de frecuencia 32768Hz (XTAL)

Las señales de reloj que controlan los diferentes periféricos pueden ser obtenidas a partir de diferentes fuentes de reloj. A continuación se detallan las diferentes fuentes de reloj y sus características:

- VLOCLK: Este oscilador es interno. Su frecuencia típica es de 12KHz, aunque oscila bastante debido a que tiene una variación de un 0.5%/°C y de un 4%/V por lo que no se debe usar con periféricos en los que se requiera precisión.
- LFXT1CLK: oscilador de baja y alta frecuencia. En baja frecuencia se puede usar con un cristal estándar de reloj o con una fuente externa de reloj de 32768Hz. En alta frecuencia (rango de 400KHz a 16MHz) puede ser utilizado con cristales, fuentes externas de reloj o resonadores.
- DCOCLK: es interno. Ofrece diferentes opciones para escoger su frecuencia de trabajo. Una de ellas es seleccionar mediante software uno de los 18 valores de frecuencia posibles, estos valores están en el rango que va de 60KHz a 26MHz por lo que se deduce que la diferencia entre sus posibles valores máximos y mínimos es alta. Otra opción es escoger una frecuencia calibrada (1, 8, 12 ó 16MHz). Por ejemplo en el caso de escoger una frecuencia calibrada de 1MHz su tolerancia máxima para el rango de temperaturas de 0 a 85°C es de ±2.5%, para una Vcc de 1.8V a 3.6 es de ±3% y la tolerancia máxima combinando ambas es del $\pm$5%. La frecuencia de este oscilador también puede ser programada mediante la colocación de una resistencia externa, mediante este método se produce una desviación de un $\pm$0.1%/°C y de un 10%/V respecto a la frecuencia elegida.

A continuación se muestran las diferentes posibilidades que había para obtener la señal de reloj que gobernara a los *timers* y a la UART:

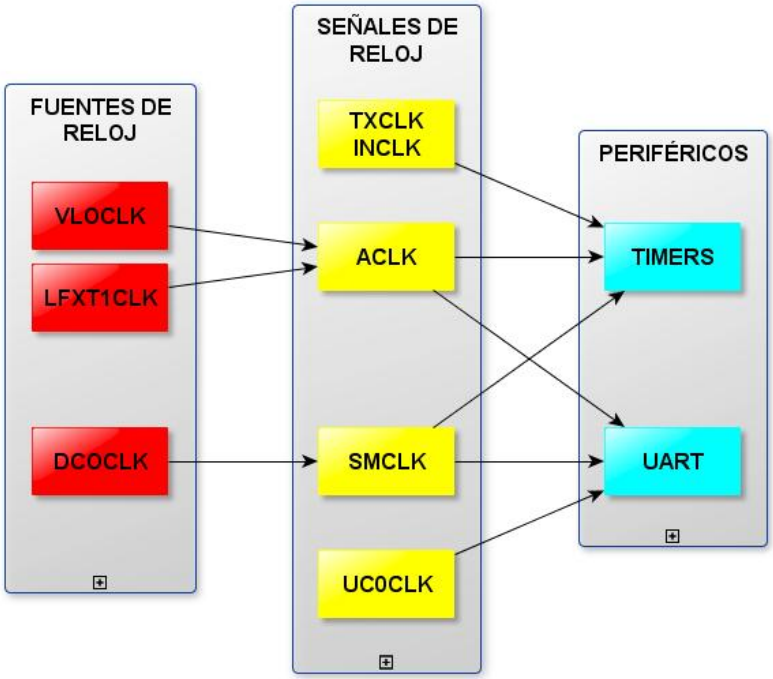


Ilustración 35: FORMAS DE OBTENER LAS SEÑALES DE RELOJ QUE GOBIERNEN A LOS PERIFÉRICOS

TXCLK, INCLK y UC0CLK se obtienen introduciendo directamente la señal de reloj por el pin correspondiente siendo necesaria la construcción completa del sistema oscilante, esta opción se descartó.

ACLK es la única señal de reloj que se mantiene activa en LPM3 y, por tanto, con la que se va a poder minimizar el consumo al máximo. La elección de ACLK como señal de reloj me lleva a escoger entre VLOCLK o LFXT1CLK. VLOCLK se descartó como fuente de reloj ya que esta no es lo suficientemente estable ni como para fijar la velocidad de transmisión de los datos entre el µC y el módulo GSM ni como para contar tiempo. La fuente de reloj elegida fue LFXT1CLK.

El valor del cristal utilizado fue de 32768Hz por dos razones:

- Interesaba generar el menor número de interrupciones posibles para que el µC permaneciera el mayor tiempo posible en modo de bajo consumo. A menor frecuencia de cristal el *timer* va a tardar más tiempo en alcanzar el valor introducido en TACCR0 y, por tanto, la interrupción se generará transcurrido un espacio de tiempo mayor.
- A menor frecuencia menor consumo.

No se han introducido condensadores con el cristal, ya que estos los proporciona el µC. Se pueden seleccionar hasta cuatro valores mediante software.

4.6 ETAPA DE ADAPTACIÓN DE TENSIÓN ENTRE TX DEL µC y RX DEL MÓDULO GSM

La función de este circuito fue detallada con anterioridad en apartado 3.9 del presente documento.

4.6.1 CIRCUITO

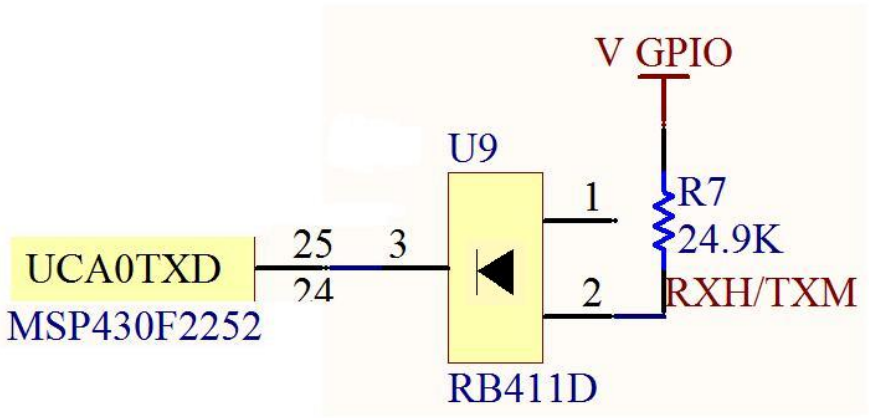


Ilustración 36: ETAPA DE ADAPTACIÓN DE TENSIÓN ENTRE TX DEL µC Y RX DEL MÓDULO GSM

4.6.2 COMPONENTES: VALOR Y FUNCIÓN

Resistencia

Teniendo en cuenta que la tensión presente en RXH/TXM debe tomar un valor de al menos 2.4V para que un nivel alto sea entendido como tal:

$$V_{GPIO} - \Delta V_R \geq 2.4V$$

Tomando como V_{GPIO} el menor valor posible:

$$\Delta V_R \leq V_{GPIO} - 2.4V, \Delta V_R \leq 2.650 - 2.4 = 0.25V$$

Como la $I_{MÁX}$ que puede absorber el pin RX del módulo es de 10µA apliqué la siguiente ecuación para el cálculo de la resistencia:

$$R \leq \frac{0.25}{I_{MÁX}} = \frac{0.25}{10uA} = 25K\Omega$$

Diodo

El valor máximo del nivel lógico bajo del µC es de 0.25V, como 0.4V es el máximo valor admisible para que un nivel lógico bajo sea entendido como tal por el módulo GSM se debe elegir un diodo que polarizado directamente su tensión no supere los 0.15V. Como la corriente máxima que va a circular por el diodo es de más o menos $\frac{(V_{GPIO})_{máx}}{R} = \frac{2.95V}{25K\Omega} = 0.12mA$ esta condición es fácil de cumplir.

4.7 ETAPA DE ADAPTACIÓN Y CONTROL DE LAS ELECTROVÁLVULAS

4.7.1 CIRCUITO

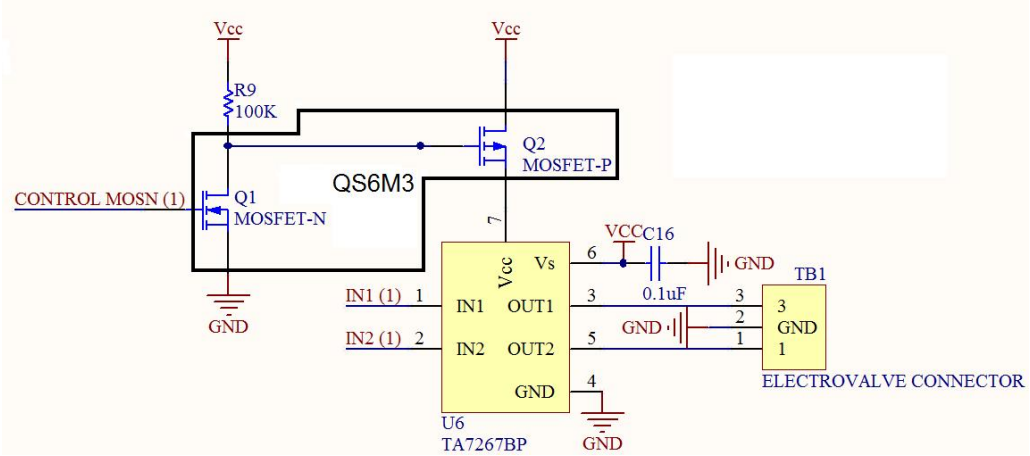


Ilustración 37: CIRCUITO DE LA ETAPA DE ADAPTACIÓN Y CONTROL

4.7.2 COMPONENTES: CÁLCULO Y FUNCIÓN

Circuito integrado QS6M3 y driver TA7267BP

La función de ambos fue explicada en el apartado 3.7 de este documento.

C_{16}

Condensador de bypass.

R_9

El MOSFET de canal N tiene una corriente de drenaje de 1µA para una $V_{GS} = 0V$, con lo que el valor de esta resistencia tendrá que ser lo suficientemente bajo como para que no se produzca la puesta en ON del MOSFET de canal P cuando estamos aplicando un cero en la puerta del MOSFET de canal N. Sabiendo que la mínima tensión umbral del MOSFET de canal P es de -0.7V se realiza el siguiente cálculo para la elección del valor de la resistencia:

$$(V_{GATE})_{pmos} - (V_{SOURCE})_{pmos} \geq -0.7V \Rightarrow V_{CC} - I \times R_9 - V_{CC} \geq -0.7$$

Despejando R_9 y sustituyendo valores:

$$R_9 \leq \frac{0.7}{I} = \frac{0.7}{1\mu A} = 700K\Omega$$

Finalmente se escoge un valor de 100KΩ ya que de este modo se reduce $(V_{GS})_{pmos}$ y, por tanto, $(I_{DS})_{pmos}$. Se escogió esta opción frente a la de reducir el consumo del NMOS cuando este está en funcionamiento ya que es la situación que se da durante la mayor parte del tiempo.

4.8 PRUEBAS DEL HARDWARE

En una primera versión del diseño hardware se construyó una sola placa, en la que iban montados todos los componentes. En esta placa la tarea de comprobación del funcionamiento de cada módulo era compleja. Al estropearse el integrado del µC y debido a que desoldarlo era complicado, se realizó un diseño modular de la PCB para facilitar las tareas de comprobación. El hardware se encuentra formado por cuatro módulos:

- Fuente de alimentación del módulo GSM.
- Módulo GSM. En este se encuentra el conector del módulo y la SIM.
- µC. Esta parte se encuentra formada por: el conector JTAG, la fuente de alimentación del µC y la etapa de adaptación de tensión entre los niveles Rx del módulo GSM y Tx de µC.
- Etapa de adaptación y control de las electroválvulas.

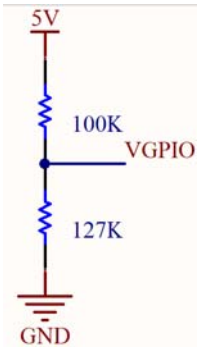
Los cuatro módulos se encuentran conectados entre sí a través de conectores.

Al testear cada parte por separado y comprobar su funcionamiento se detectó que el módulo GSM no detectaba la presencia de SIM, ésta era su repuesta al introducir el comando AT+CPIN=”4923” desde el µC:

- + CME ERROR: 10, el cual significa que la SIM no se encuentra insertada o está dañada.

Por lo que se utilizo la placa proporcionada por Libelium (su esquema circuital se encuentra el anexo número uno). Para adaptar esta placa con la del µC se hizo lo siguiente:

- Se extrajo las resistencias que adaptaban los niveles de tensión en RX ya que esta etapa ya estaba incluida en la placa del µC (apartado 4.6 del presente capítulo). Además servían para adaptar niveles de tensión mayores.
- El pin de VGPIO no estaba accesible por lo que la etapa de adaptación se tuvo que alimentar de la forma indicada a continuación:



Donde:

$$V_{GPIO} = 5 * \frac{127K\Omega}{100K\Omega + 127K\Omega} = 2.79$$

Esta placa de Libelium no tiene el pin de POK_IN disponible por lo que el módulo GSM está permanentemente encendido.

5 CIRCUITO DE CONFIGURACIÓN DEL MÓDULO GSM

5.1 FUNCIÓN

Este circuito fue utilizado para:

- Conocer los parámetros configurados por defecto en el módulo GSM así como para establecer aquellos que interesaban para llevar a cabo el proyecto.
- Guardar los SMS con los que responde el sistema de riego al usuario en el módulo GSM.

Si no se hubiera realizado este montaje se hubiera tenido que introducir los comandos AT de configuración y cada uno de los SMS en la flash del µC, por lo que se ganó en memoria y comodidad.

5.2 DIAGRAMA DE BLOQUES

Para la realización de este montaje me aproveché de la placa proporcionada por Libelium (http://www.cooking-hacks.com/index.php/arduino-gprs-module.html) con el módulo GSM, cuyo esquema del circuito se encuentra en el anexo número uno.

Mediante el programa de Windows HyperTerminal se envían los comandos AT al módulo GSM, visualizando la respuesta de éste en la pantalla del ordenador.

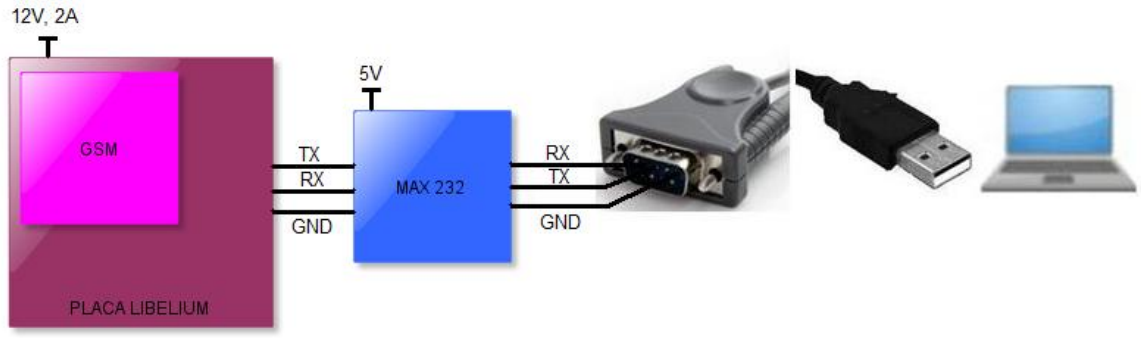


Ilustración 38: CIRCUITO PARA LA CONFIGURACIÓN DE PARÁMETROS DEL MÓDULO GSM

6 SOFTWARE

6.1 INTERFAZ USUARIO-SISTEMA

El usuario va a comunicarse únicamente mediante el envío de SMS con el sistema de riego, permitiéndole de este modo actuar sobre él.

La estructura elegida para el SMS fue la siguiente: /CLAVE/COMANDO/CONFIGURACIÓN./.

Las posibles respuestas del sistema comunes a todos los SMS son las siguientes:

- CLAVE INCORRECTA, si la clave es introducida de forma errónea.
- COMANDO INCORRECTO, esta respuesta se recibe siempre que la sintaxis del comando no sea correcta excepto para el comando SU (SIGN UP), que se recibirá COMANDO INCORRECTO O USUARIO NO REGISTRADO.
- CONFIGURACION INCORRECTA, la configuración no se ha introducido de la forma indicada o algunos de los parámetros introducidos no son lógicos (por ejemplo día de la semana mayor que 7).

A continuación se va a dar una explicación detallada de cómo el usuario tiene que estructurar los SMS para darse de alta o de baja del sistema, configurarlo, modificarle la clave, activar o desactivar el riego por configuración e iniciar o finalizar el riego de forma inmediata.

6.1.1 SIGN UP

SMS: /RIXSMS/SU/

Este es el primer paso que hay que dar ya que sin él no podremos introducir ningún otro comando.

Las respuestas particulares de este comando son:

- COMANDO INCORRECTO O USUARIO NO REGISTRADO, si el comando no se escribe correctamente o si se intenta introducir uno que no sea SU antes de darse de alta.
- USUARIOS COMPLETOS, se admiten, como máximo, dos usuarios.

6.1.2 MODIFICAR LA CLAVE

SMS: /RIXSMS/P/XXXXXX./

La clave introducida debe tener exactamente seis caracteres y estos pueden ser cualesquiera. Una vez modificada, en lugar de RIXSMS se introducirá la elegida por el usuario.

6.1.3 CONFIGURAR RIEGO

SMS: /RIXSMS/C/AespacioBespacioCespacioDespacioE./

A: día de la semana. Se asignará un valor del 1 al 7.

B: sector de riego a configurar. Si B=1 el comando actuará sobre el sector 1, si B=2 sobre el sector 2, en cambio si B=3 el comando actuará sobre ambos.

C: días de riego de la semana. Formado por siete caracteres, el primero corresponde al lunes y el último al domingo. Cada carácter puede tomar el valor 0 si ese día no se desea regar o el número del día de la semana en el que estemos si por el contrario deseamos regar ese día de la semana. Por ejemplo si el usuario desea regar martes y jueves C=0204000.

D: hora de riego. Su formato es HH:MM.

E: tiempo de riego en minutos desde 000 hasta 999min. Se deben completar tres cifras, si se quiere regar durante hora y media E=090.

6.1.4 RIEGO INMEDIATO

SMS: /RIXSMS/1/FespacioG./

F: indica el sector en el que se va a iniciar el riego. Funciona de la misma forma que B.

G: tiempo de riego en minutos.

El riego por configuración será omitido si coincide con un riego inmediato. Si se recibe un SMS de riego inmediato cuando se encuentra activo el riego por configuración se continuará con el riego inmediato.

6.1.5 FINALIZAR RIEGO, DESACTIVAR RIEGO POR CONFIGURACIÓN

SMS: /RIXSMS/2/H./

H: sector de riego a desactivar y/o parar. Funciona exactamente igual que B.

Si se está regando en el momento de recepción de este mensaje se finalizará el riego y se anulará cualquier riego futuro ya sea programado o inmediato.

6.1.6 RESTABLECER RIEGO POR CONFIGURACIÓN

SMS: /RIXSMS/3/I./

I: indica el sector de riego en el que se va a restablecer el riego por configuración. Se programa de la misma forma que B.

6.1.7 REMOVE ACCOUNT

Para darse de baja del sistema de riego se enviará el siguiente SMS: /RIXSMS/RA/

6.2 COMANDOS AT

Para una explicación detallada de qué son los comandos AT, para qué sirven cada uno de los comandos utilizados en el proyecto, sus posibles parámetros y las respuestas del módulo ante estos ir al anexo número 3.

6.2.1 COMANDOS DE CONFIGURACIÓN

Todos los comandos AT que aparecen a continuación fueron introducidos mediante el programa HyperTerminal de Windows para la configuración del módulo GSM. En este apartado se hace referencia a cuáles fueron los valores de los parámetros seleccionados para cada comando y por qué, también se introducen las respuestas que se obtuvieron para cada comando pero omitiendo los CR y LF con el fin de ganar en claridad.

AT+IPR?, AT+IPR=<rate>

AT+IPR?

+IPR: 19200

OK

AT+IPR=9600

OK

El *baud rate* se configura a 9600 ya que es la máxima velocidad posible para el µC utilizando para la UART una señal de reloj de 32768Hz generada a partir del oscilador LFXT1CLK.

AT+CREG?

AT+CREG?

+CREG: 0,5

OK

El cero es el valor que me interesa ya que no quiero recibir ningún *unsolicited result code* cuando haya un cambio en el registro. El 5 nos dice que estamos registrados en modo *roaming*, esto se debe a que mi operador SIMYO tiene por defecto activada esta opción.

AT+CMGF?, AT+CMGF=<mode>

AT+CMGF?

+CMGF: 0

OK

AT+CMGF=1

OK

Las diferencias entre el modo texto y el modo PDU son las siguientes:

- El SMS en modo PDU se encuentra formado por una cadena de caracteres representados por octetos hexadecimales o semioctetos decimales, mientras que el modo texto es una codificación del PDU.
- En modo texto la aplicación está limitada a la opción de codificación preestablecida, en modo PDU cualquier opción de codificación de 7, 8 ó 16 bits puede ser utilizada. Hasta puedes construir la tuya propia de 8 bits.
- La sintaxis y la respuesta de algunos comandos destinados a gestionar actividades relacionadas con los SMS pueden variar dependiendo si se elige un modo u otro.
- En modo PDU el dispositivo envía y recibe SMS codificados en el formato utilizado por la red. Si el modo texto es el utilizado el dispositivo será el encargado de convertir el SMS de modo PDU a modo texto y viceversa.
- La cadena PDU contiene datos adicionales a los que presenta la cadena de caracteres en modo texto.

A continuación se detallan las opciones de codificación de caracteres en modo PDU, para una explicación detallada de cada tipo de codificación ir al anexo número 4.

Las opciones de codificación de caracteres en modo PDU son las siguientes:

- *GSM 7 bit default alphabet*. Cuando usamos este tipo de alfabeto la norma nos requiere hacer grupos de ocho bits a partir de palabras de siete, pudiendo transmitir de esta forma hasta 160 caracteres al no desperdiciar un bit por cada palabra.

  A continuación se detalla cómo se realizaría esta transformación de caracteres de siete bits a ocho utilizando de ejemplo la palabra "HOLA":

  Primero se codifica cada letra con el número hexadecimal que le corresponde.

	H	O	L	A
Hex	48	4F	4C	41
Bin	1001000	1001111	1001100	1000001

Tabla 16: CODIFICACIÓN DE "HOLA" SEGÚN 7 BIT GSM DEFAULT ALPHABET

  Luego se completa cada palabra binaria con los bit/s menos significativos de la siguiente hasta que cada una de ellas se encuentre formada por ocho bits.

Bin	1 1001000	00 100111 ~~1~~	001 10011 ~~00~~	1000 ~~001~~
Hex	C8	27	33	08

Tabla 17: TRANSFORMAR CARÁCTERES "HOLA" DE 7 A 8 BITS

  Esta operación requiere del desarrollo de una función en el programa introducido en el µC que realice la codificación a 8 bits y la descodificación a 7 bits. Además de esto este tipo de codificación difiere en algunos caracteres con la ASCII (utilizada por el µC), por lo que se debería de tener cuidado en no utilizarlos.

- 8 bit. Esta opción suele ser utilizada para el envío de datos por ejemplo en MMS, tonos de llamada y WAP push; pero también se puede emplear para transmitir datos en una aplicación propia.

  Aunque puede ser aplicada también para el envío de SMS de texto está contraindicado debido a que no todos los dispositivos la soportan y a que no hay forma de indicar que sistema de representación de caracteres se está utilizando. Por ejemplo, si se envía un SMS utilizando una codificación a ocho bits y el receptor tiene en ese momento la misma opción activada en su dispositivo el SMS podrá ser leído correctamente, si no es así puede ocurrir que el texto no se pueda representar o que dentro del mismo SMS haya caracteres que si se hayan traducido de forma correcta al tener la misma representación binaria en los dos tipos de códigos y otros no.

- UCS2. Aunque el valor hexadecimal de los caracteres codificados mediante este tipo de código es el mismo que el respectivo ASCII utilizado por el µC, estos se encuentran formados por 16 bits lo que obligaría a implementar alguna función para la codificación de ASCII a UCS2 y la descodificación de UCS2 a ASCII. Este tipo de codificación tiene otro inconveniente y es el número máximo de caracteres por SMS que queda reducido a 70.

Para enviar un SMS con la palabra "HOLA" en modo PDU y a través de la codificación *7 bit GSM default alphabet* deberíamos proceder la siguiente forma:

- Introducir AT+CMGS=longitud en octetos de la trama PDU (a partir de 11 inclusive).
- Esperar a recibir > desde el módulo
- Introducir 07914356060013F111000981260643 21F50000A706C8273308 seguido de control+z.

Donde (todos los parámetros son octetos hexadecimales, a no ser que se indique lo contrario):

- 07: número de octetos que forman la información acerca de SMSC. Si es cero el número de SMSC guardado en el teléfono será el usado.
- 91: tipo de dirección del SMSC. 91 significa formato internacional.
- 4356060013F1: número del SMSC, 34656000311. Cada par de semioctetos decimales se intercambian de posición y como el número es impar se añade una F por completar una secuencia de octetos.
- 11: primer octeto del SMS-SUBMIT PDU.
- 00: valor que adjudica el número de referencia para el SMS. Con 00 es el propio dispositivo el que lo selecciona.
- 09: número de semioctetos decimales del teléfono de envío.
- 81: tipo de dirección del teléfono de envío.
- 26064321F5: número del teléfono de envío. Cabe hacer las mismas observaciones que en el número del SMSC.
- 00: Identificador del protocolo.
- 00: indicamos el tipo de alfabeto utilizado. 00 para 7 bit GSM default alphabet, 01 para 8 bit y 10 para UCS2. El valor 11 se encuentra reservado.
- A7: período de vigencia del SMS.
- 04: septetos que forman el SMS. Serían número de octetos si se hubiera elegido cualquier otro tipo de codificación.
- C8273308: texto del SMS.

En cambio para enviar el mismo SMS pero en modo texto solamente tendríamos que:

- Enviar al módulo GSM AT+CMGS="nº de teléfono de envío".
- Esperar a recibir >.
- Introducir HOLA<control+z>.

Al recibir un SMS en modo PDU con la palabra "HOLA" la trama de datos que recibiríamos sería: 07914356060018F2040B914356554985F200002050825142950004C8273308, donde (todos los parámetros son octetos hexadecimales, a no ser que se indique lo contrario):

- 07: número de octetos que forman la información acerca del SMSC.
- 91: tipo de dirección del SMSC. 91 significa formato internacional.
- 4356060018F2: número del SMSC. 34656000812. Cada par de semioctetos decimales se intercambian de posición y como el número es impar se añade una F por completar una secuencia de octetos.
- 04: se indica el tipo de SMS, si hay más SMS para enviar, si hay que enviar un informe de estado a la entidad desde la que se ha recibido el SMS…
- 0B: número de semioctetos decimales del teléfono de envío.
- 91: tipo de dirección del teléfono de envío.
- 4356554985F2: número del teléfono de envío. Cabe hacer las mismas observaciones que en el número del SMSC.
- 00: Identificador del protocolo.
- 00: se indica el tipo de alfabeto utilizado en el SMS. 00 para 7 bit GSM default alphabet, 01 para 8 bit y 10 para UCS2. El valor 11 se encuentra reservado.
- 20508251429500: Nos indica cuándo se produjo la llegada del SMS en semioctetos decimales. En este caso el 28/05/02 a las 15:24:59. Las dos últimas cifras indican la desviación de la hora a la que se recibió el SMS respecto a la marcada por GMT en ese momento en cuartos de hora. Si hubiera un 04 nos indicaría que la hora reflejada en el SMS es igual a GMT+1.
- 04: septetos que forman el SMS. Serían número de octetos si se hubiera elegido cualquier otro tipo de codificación.
- C8273308: texto del SMS.

En cambio el mismo SMS recibido en modo texto quedaría de esta forma:

"REC UNREAD","+34655594582","","02/05/28,15:24:59+00"

HOLA

Por los problemas expuestos en la elección del tipo de código en modo PDU y una mayor sencillez al enviar y recibir SMS en modo texto, se eligió este último.

AT+CSCS?

AT+CSCS?

+CSCS: "IRA"

OK

IRA es el tipo de alfabeto preestablecido, por tanto deberá ser este el que se utilice en modo texto. Para conocer más acerca de este tipo de codificación ir al anexo número 4.

AT+CPMS?, AT+CPMS=?, AT+CPMS="ME,"SM",ME"

AT+CPMS?

+CPMS: "SM",0,50,"SM",0,50,"SM",0,50

OK

AT+CPMS=?

+CPMS: ("SM","ME"),("SM","ME"),("SM","ME")

OK

AT+CPMS="ME","SM","ME"

+CPMS: 0,100,0,50,0,100

OK

El módulo te permite mediante este comando elegir la memoria desde la que se van a leer y borrar los SMS, la memoria que se va utilizar para enviar SMS guardados y escribirlos o la memoria utilizada para guardar los SMS recibidos.

La elección correcta de memorias en este paso me va a permitir no tener que implementar este comando en el µC ya que si por ejemplo eliges una memoria para leer y borrar y otra diferente para guardar los SMS recibidos cuando quiera leer o borrar un SMS recibido tendré que ejecutar AT+CPMS.

Como memoria en la cual se van a escribir y desde la cual se van a enviar los SMS se eligió la SIM. En cambio la memoria seleccionada para leer, borrar y guardar SMS recibidos fue la memoria del módulo GSM. Esta elección se debió a que los SMS que van a ser enviados desde el GSM no tienen que ser leídos ni borrados, en cambio los SMS recibidos sí que tienen que ser leídos (para procesar la información) y borrados (para que la memoria nunca llegue a estar completa).

AT+CSDH?

AT+CSDH?

+CSDH: 0

OK

Evito que cuando lea un SMS me aparezcan parámetros innecesarios.

AT+CNMI?

AT+CNMI?

+CNMI: 0,0,0,0,0

OK

Me sirven los parámetros configurados por defecto, ya que no quiero que el módulo GSM avise al µC cuando se reciba un SMS, ni recibir ningún tipo de indicación acerca de CBM ni tampoco un informe de estado acerca de si el SMS ha llegado con éxito a su destinatario.

AT+CMEE?, AT+CMEE=<n>

AT+CMEE?

+CMEE: 1

OK

AT+CMEE=0

OK

Se desactiva la recepción de errores de forma numérica o detallada, ya que hay hasta 85 tipos de errores y sería complicado manejarlos con el µC.

AT&W

AT&W

OK

Por último se guardan todos los cambios realizados en la configuración del módulo GSM.

6.2.2 COMANDOS AT PARA EL MANEJO DE SMS

El comando AT para guardar SMS en la memoria del módulo fue introducido únicamente mediante HyperTeminal, en cambio los otros tres se implementaron también en el programa del µC.

Guardar SMS en el módulo

AT+CMGW=”+346XXXXXXXX”

>CLAVE INCORRECTA<control+z>

+CMGW: 1

OK

AT+CMGW=”+346XXXXXXXX”

>COMANDO INCORRECTO O USUARIO NO REGISTRADO<control+z>

+CMGW: 2

OK

AT+CMGW=”+346XXXXXXXX”

>COMANDO INCORRECTO<control+z>

+CMGW: 3

OK

AT+CMGW=”+346XXXXXXXX”

>USUARIOS COMPLETOS<control+z>

+CMGW: 4

OK

AT+CMGW=”+346XXXXXXXX”

>CONFIGURACION INCORRECTA<control+z>

+CMGW: 5

OK

AT+CMGW=”+346XXXXXXXX”

>CARACTERES MAXIMOS SUPERADOS <control+z>

+CMGW: 6

OK

Para cada uno de los SMS el módulo respondió con +CMGW:<nº entero positivo>. Este número se utiliza para ordenar al módulo GSM que envíe el SMS guardado en esa posición. Por ejemplo para enviar el SMS de CONFIGURACIÓN INCORRECTA tendremos que hacer llegar vía UART al módulo GSM el comando AT+CMSS=4.

Envío de SMS

Existen dos opciones disponibles AT+CMGS y AT+CMSS. Como el texto de los SMS no cambia y cada uno de ellos se enviará tantas veces como sea necesario se eligió el comando AT+CMSS frente a AT+CMGS ya que de este modo no se tiene que introducir el cuerpo del SMS cada vez que se envía.

AT+CMSS=1[,”nº de teléfono del usuario”]

+CMSS: 10

OK

Leer SMS

AT+CMGR=1

+CMGR: "REC UNREAD","+34XXXXXXXXX","","12/04/27,15:39:56+08"

Hola gsm!

OK

Si en esa posición no hay ningún SMS el módulo responde con +CMS ERROR: 321.

Borrar SMS recibidos

AT+CMGD=0,1

OK

De este modo borro todos los SMS leídos.

6.2.3 COMANDO AT PARA LA OBTENCIÓN DE LA HORA Y FECHA

AT+CCLK?

+CCLK: "12/06/01,11:57:57+04"

OK

El módulo GSM obtiene la hora y la fecha después de registrarse en la red.

6.2.4 COMANDO AT PARA INTRODUCIR EL PIN

AT+CPIN=”4923”

OK

6.3 PROGRAMA PRINCIPAL

El programa se ha desarrollado pensando en la idea inicial de encender el módulo GSM cada media hora para ver si se ha producido la recepción de algún SMS y luego desconectarlo, aunque estas acciones no puedan llevarse a cabo al no poder acceder en la placa de Libelium al pin POK_IN del módulo GSM.

Al iniciar el programa se leen los posibles datos que haya podido introducir el usuario antes de que se produjera un fallo en la tensión de alimentación o se agotara la batería. Estos datos se almacenan en flash en unas secciones denominadas segmentos. Hay cuatro segmentos disponibles (A, B, C y D), cada uno de ellos tiene un espacio disponible de 40 bytes. No es aconsejable realizar ninguna operación en el segmento A ya que en él el fabricante introduce datos relativos a la calibración de DCO. En el segmento B se guarda usuario1, en el C usuario2 y en el D la clave.

Una vez hecho esto el programa seguirá por el main2 y 3 al no haber transcurrido aún media hora. Al no encontrarse aún nada activo se entra en bajo consumo. El µC saldrá de este modo cada treinta segundos a través de la interrupción generada mediante el *timer* A. Volverá al modo activo siempre y cuando haya transcurrido media hora y/o una o ambas estaciones estén activas o pendientes de activación.

Cuando se hayan cumplido los treinta minutos se enciende el GSM, se comprueba que esté disponible y se introduce el pin. Una vez se ha hecho esto se comprueba si se ha recibido algún SMS mediante AT+CMGR. Si la respuesta es positiva se separa el texto del SMS en tres partes CLAVE, COMANDO y CONFIGURACIÓN.

Mediante la función procesar SMS se realiza lo siguiente: comprobar si el usuario ya está registrado; verificar que la clave, el comando y la configuración no se hayan introducido de forma errónea. Si es así se guardarán los datos introducidos por el usuario y se realizarán las acciones pertinentes. A través de esta función también se envían los SMS de respuesta al usuario siempre y cuando la clave sea la correcta si el usuario no está registrado o de todas formas si sí que lo está.

Una vez leídos los SMS se borran mediante AT+CMGD para que no terminen por ocupar todo el espacio de memoria disponible.

Antes de desconectar al módulo GSM mediante el comando AT*PSCPOF se le pregunta la fecha y la hora con el fin de conocer si se tiene que iniciar el riego por configuración.

En el main4 se actualiza el día de la semana y se comprueba si se tiene que iniciar el riego por configuración. El día de la semana sólo se actualiza en el caso de que al sistema le haya llegado un SMS de riego por configuración. La comprobación se omitirá si ese día ya se ha regado, si la estación ya está activa o si llega un SMS de riego inmediato. Únicamente se realizarán acciones si la HORA DE RIEGO–HORA ACTUAL=0 (se iniciará el riego) o si la HORA DE RIEGO-HORA ACTUAL<30’ (el riego se marcará como pendiente).

El inicio del riego cuando HORA DE RIEGO-HORA ACTUAL=0 o se recibe un SMS de riego inmediato se lleva a cabo en el main5, en esta parte del programa también se finaliza el riego o se anula el riego por configuración si se recibe un SMS de stop. En caso de recibir un SMS de riego inmediato mientras la estación se encuentra activa, se regará durante el tiempo indicado en el último mensaje de texto.

La función de la parte del programa principal representado por el main2 es activar el riego si este se encuentra pendiente de activación.

Y por último el main3 se encarga de finalizar el riego una vez cumplido el tiempo de riego.

A continuación se exponen únicamente los diagramas correspondientes al programa principal. Para ver los restantes ir al anexo número cinco.

En los diagramas las flechas verdes corresponden a un “sí”, en cambio las rojas lo hacen a un “no”.

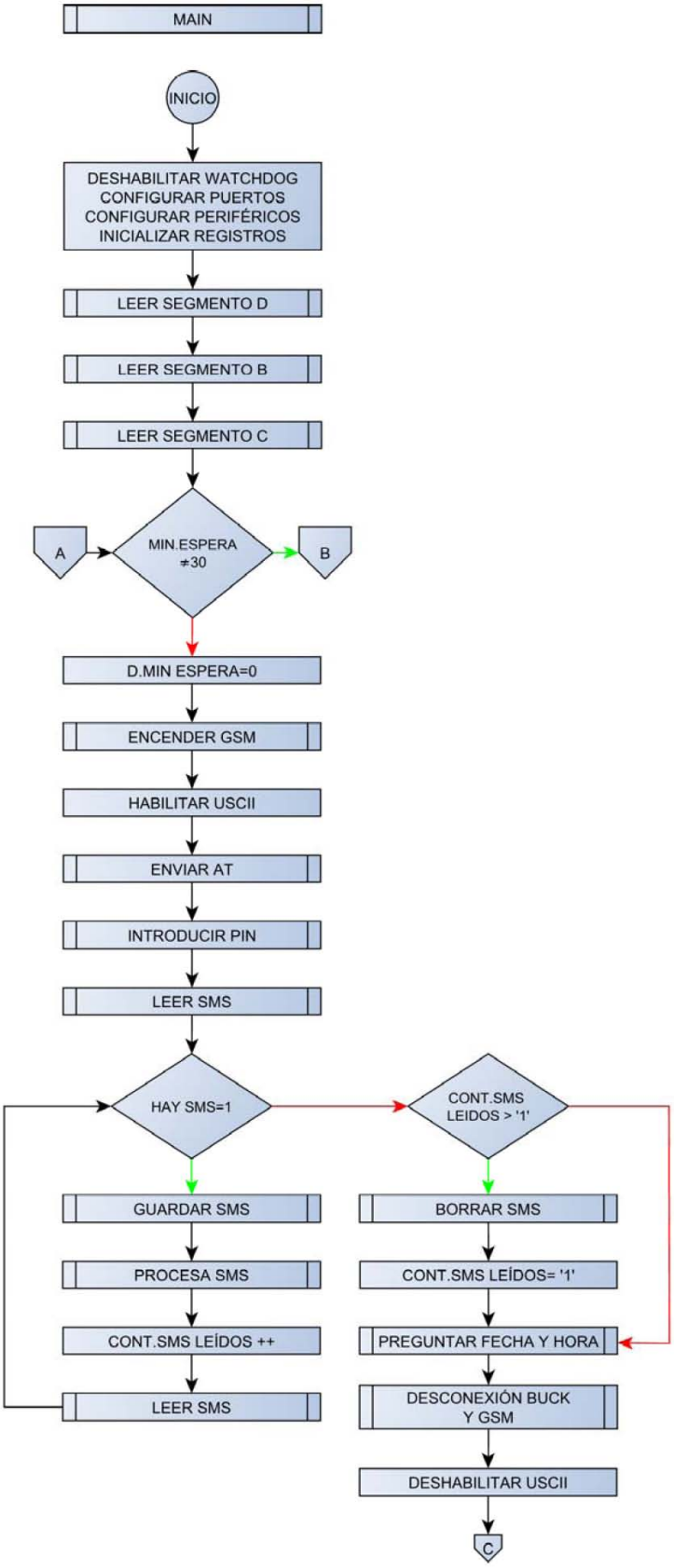


Ilustración 39: DIAGRAMA DE BLOQUES MAIN1

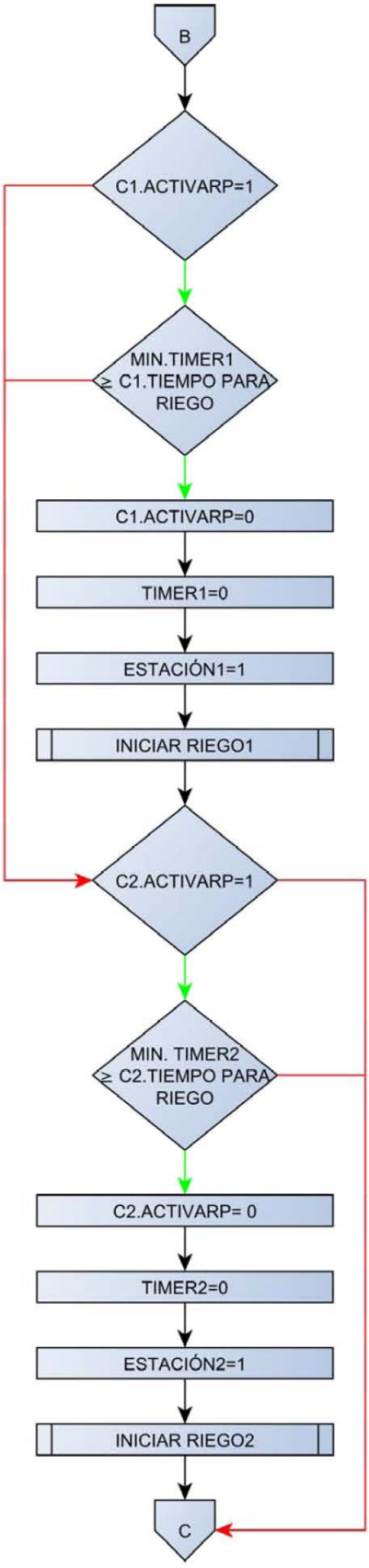


Ilustración 40: DIAGRAMA DE BLOQUES MAIN2

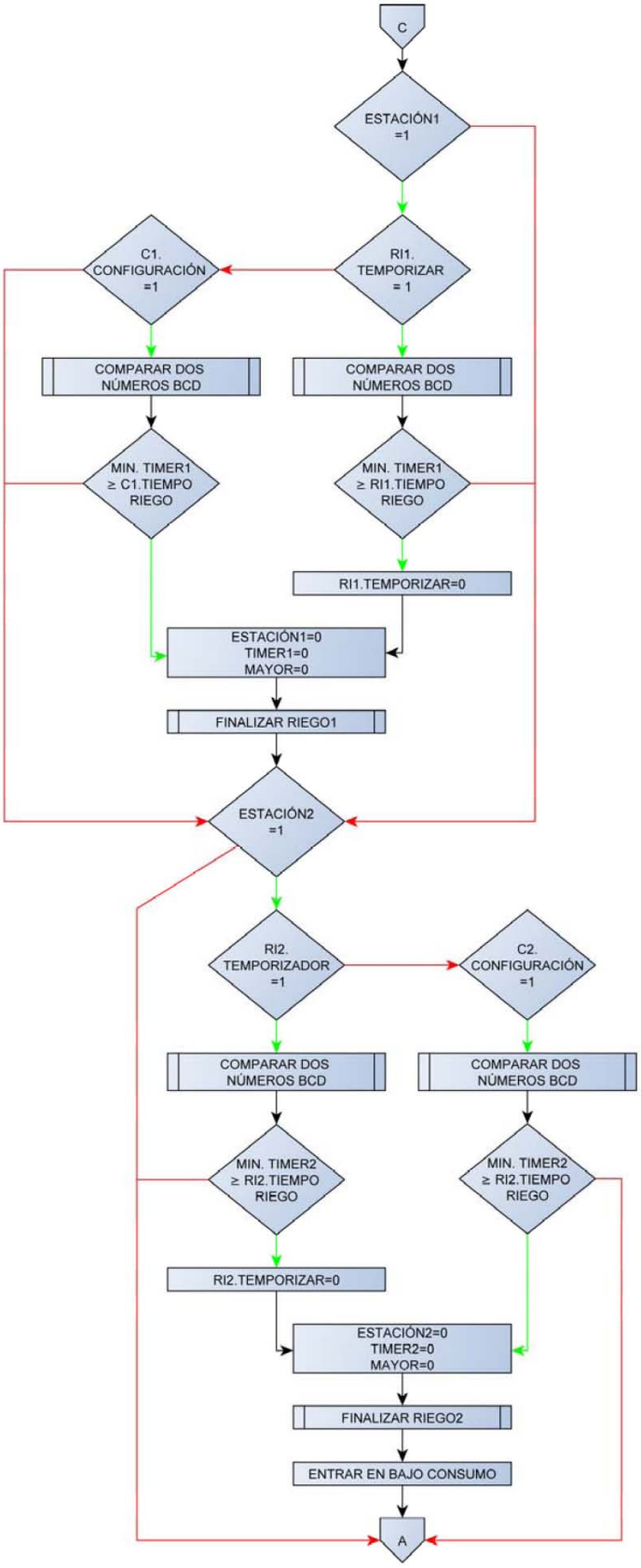


Ilustración 41: DIAGRAMA DE BLOQUES MAIN3

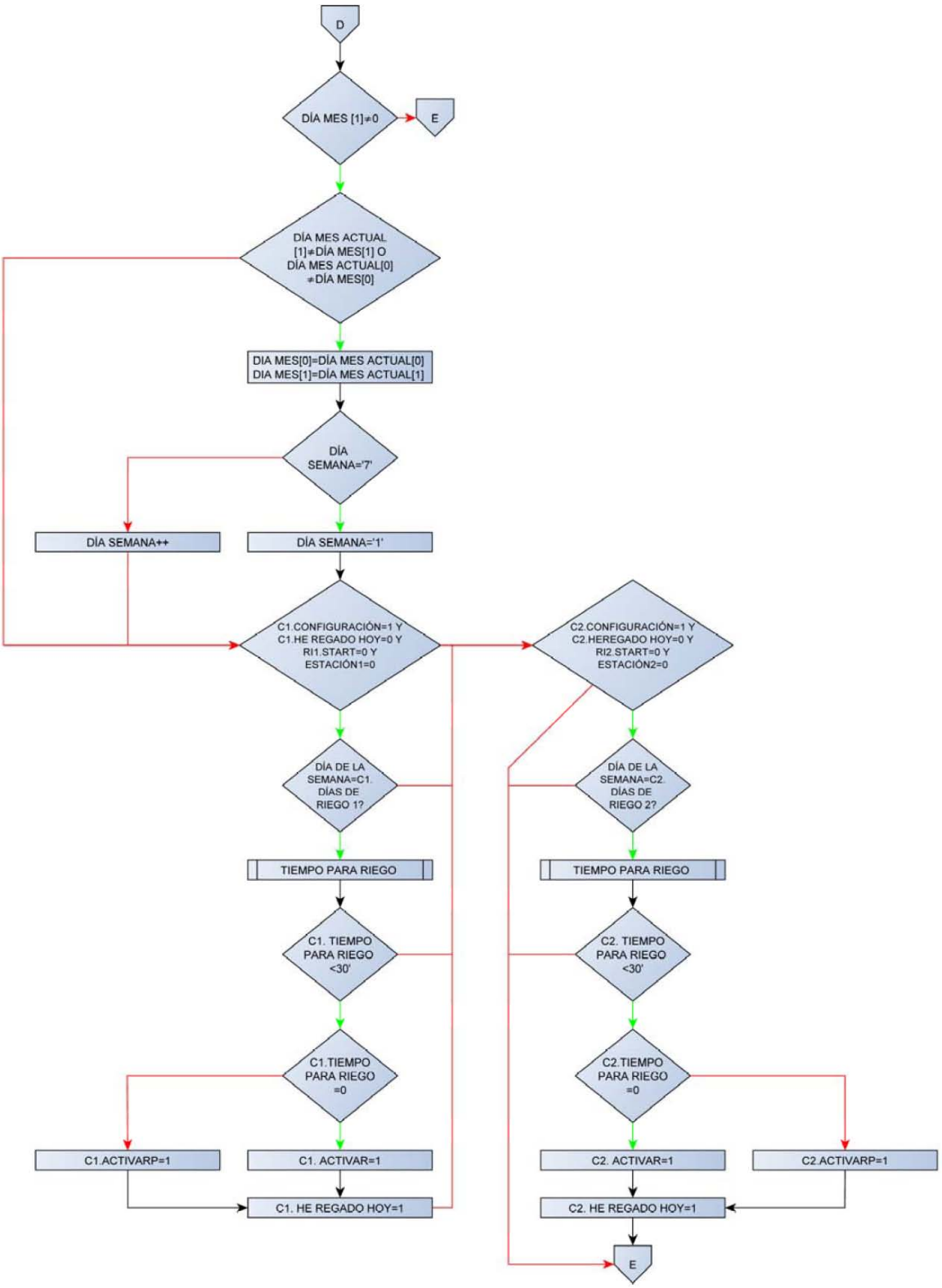


Ilustración 42: DIAGRAMA DE BLOQUES MAIN4

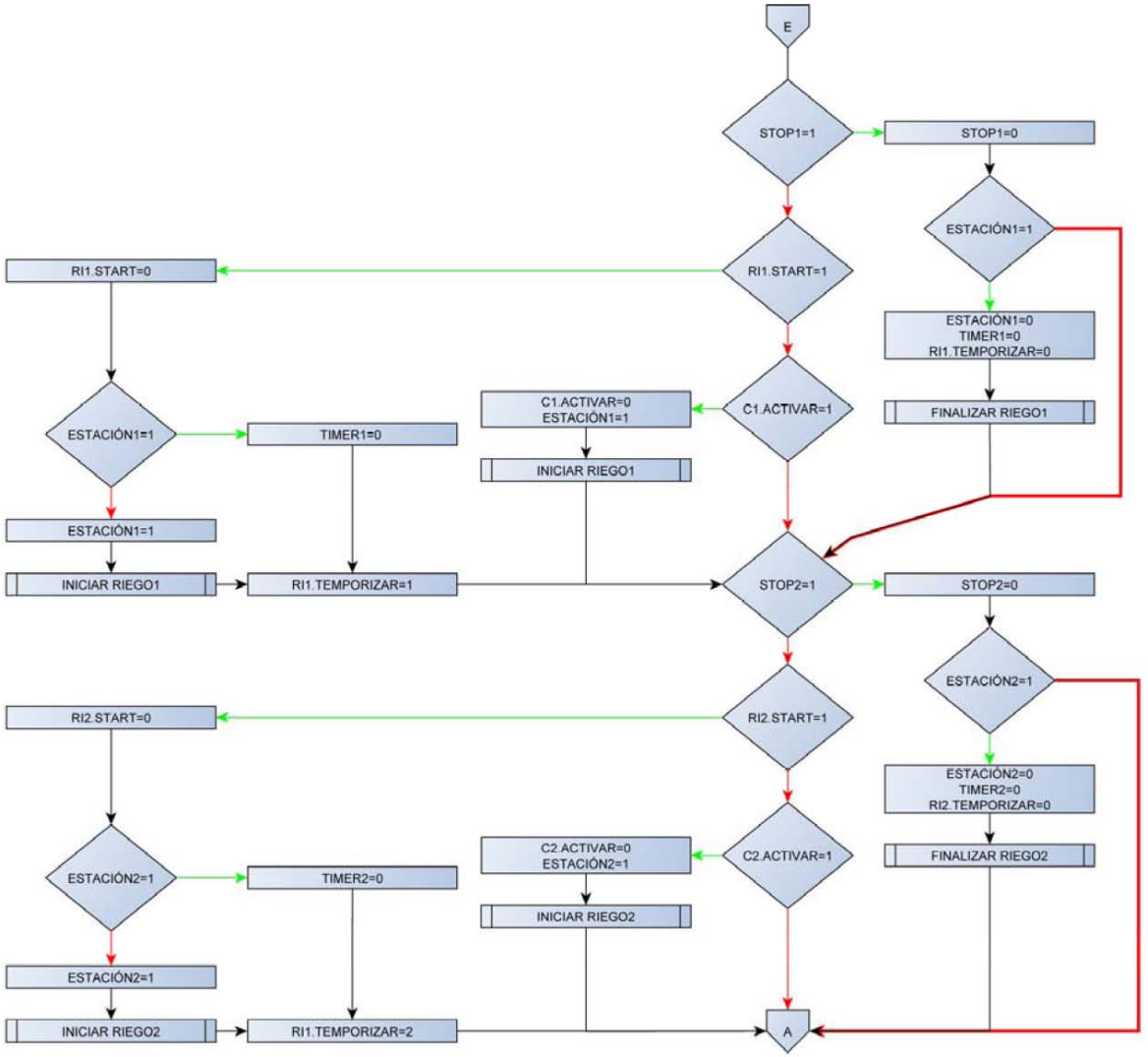


Ilustración 43: DIAGRAMA DE BLOQUES MAIN5

7 CONCLUSIONES Y FUTUROS DESARROLLOS

7.1 CONCLUSIONES

El proyecto me ha sido útil para:

- Reforzar mis conocimientos de electrónica de potencia, ya que se ha hecho uso de estructuras típicas de esta disciplina como el convertidor buck y los *drivers.*
- Ampliar mis conocimientos de lenguaje C.
- Saber manejar un µC y un sistema de desarrollo distinto al que utilizamos en la carrera.
- Conocer y manipular casi todos los comandos AT correspondientes al manejo de los SMS.
- Aplicar los conocimientos acerca del programa ALTIUM.
- Adquirir conocimientos acerca del diseño de PCB's.

7.2 FUTUROS DESARROLLOS

Sería conveniente desarrollar un circuito que detecte el nivel de la batería y cuando éste esté por debajo de un determinado nivel avisar al usuario mediante un SMS.

El usuario únicamente puede modificar los parámetros del sistema mediante el envío de SMS. La colocación de una pantalla LCD y el desarrollo de otra interfaz usuario-sistema podría ser bastante útil.

La introducción de un sensor de lluvia para no iniciar o finalizar el riego sería apropiado con el fin de minimizar el consumo de agua. Hay diversos tipos de sensores:

- La mayoría usan discos higroscópicos que se hinchan con la presencia de agua y se encogen cuando deja de llover, accionando a su vez un interruptor.
- Otros miden la cantidad de agua almacenada en un recipiente.
- Algunos cuentan con un detector de congelamiento para evitar que el sistema trabaje con temperaturas extremadamente frías.

Normalmente están conectados a los terminales del control de irrigación o están instalados en serie con el circuito de la válvula solenoide de forma que evitan que se abran las válvulas si se ha detectado lluvia.

Una opción mejor que la anterior es la colocación de un sensor de humedad ya que estos únicamente van a desactivar el sistema cuando la humedad del suelo se encuentre por encima del límite indicado por el usuario.

Si el sistema de riego está concebido para su uso en instalaciones agropecuarias la implementación de estas dos herramientas podría ser interesante:

- Programación de la cantidad de fertilizante que va a ser disuelto con el agua del riego.
- Lisímetro: su utilización permite extraer muestras de la solución nutritiva del suelo, mediante éstas se determina la disponibilidad de nutrientes minerales con lo cual se puede razonar si se cumplen los requerimientos nutritivos del cultivo y en consecuencia dimensionar los requerimientos de fertilizantes.

8 REFERENCIAS

8.1 CONVERTIDOR BUCK

- http://cds.linear.com/docs/Datasheet/3680fb.pdf
- http://www.polyscope.ch/dlCenter/ps/2010_15/15_10.07.pdf

8.2 MODULO GSM

- http://support.sagemcom.com/site/specifications/URD1_OTL_5635.1_007_70230_ed_06_31July09_-_HiLo_application_note.pdf
- http://www.cooking-hacks.com/index.php/documentation/tutorials/arduino-gprs-quadband
- http://support.sagemcom.com/site/specifications/URD1_5635.1_005_70086_ed_06_03July09_-_HiLo_technical_specification.pdf

8.3 REGULADOR LINEAL DE VOLTAJE

- http://www.torex-europe.com/clientfiles/File/app_notes/Basic%20Knowledge%20of%20LDO%20Voltage%20Regulators.pdf
- http://www.ti.com/lit/an/slva072/slva072.pdf
- http://www.farnell.com/datasheets/1447536.pdf

8.4 MICROCOTROLADOR

- http://www.ti.com/lit/ds/symlink/msp430f2252.pdf
- http://www.ti.com/lit/ug/slau144i/slau144i.pdf

8.5 TARJETA SIM

- http://www.gsmspain.com/foros/h825245_General_Sim-pins-ventajas.html
- http://pinouts.ru/Memory/SmartCardIso_pinout.shtml

8.6 ELECTROVÁLVULAS

- http://es.wikipedia.org/wiki/Electrov%C3%A1lvula
- http://www.solenoidsolutionsinc.com/custom-solutions/latching-solenoid-valves--low-energy/
- http://profesores.elo.utfsm.cl/~jgb/CARVALLOVARGASc.pdf

8.7 MINMIZACIÓN DE EMI

- http://diec.unizar.es/~tpollan/libro/Apuntes/digT3.pdf
- http://www.intersil.com/content/dam/Intersil/documents/an13/an1325.pdf

8.8 COMANDOS AT

- http://bluehack.elhacker.net/proyectos/comandosat/comandosat.html
- http://www.developershome.com/sms/
- http://www.blogelectronica.com/sms-pdu/
- http://www.dreamfabric.com/sms/
- http://www.libelium.com/squidbee/upload/2/2f/AT_Command_Set_for_SAGEM_HiLo_HiLoNC_Modules_-_5635.1_008_70248_ED07_24june09.pdf
- http://www.etsi.org/deliver/etsi_ts/127000_127099/127005/05.00.00_60/ts_127005v050000p.pdf
- https://github.com/bitcoder/ruby_ucp/wiki/SMS-Alphabets

8.9 CÓDIGOS DE CODIFICACIÓN

- http://www.itu.int/rec/T-REC-T.50-199209-I/es
- http://www.zytrax.com/tech/characters/
- http://www.commzgate.com/blog/2007/07/the-gsm-default-alphabet-set-gsm-0338-7-bit/

ANEXO Nº1: ESQUEMAS DEL CIRUCITO

FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

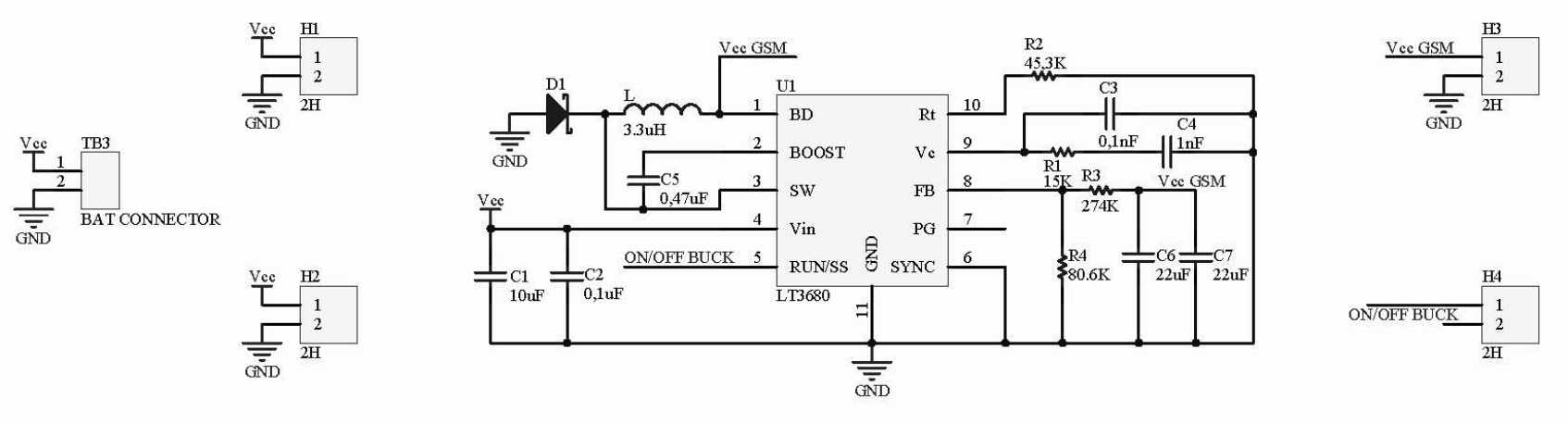


Ilustración 44: ESQUEMA COMPLETO DEL CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

CONECTOR DEL MÓDULO GSM Y SIM

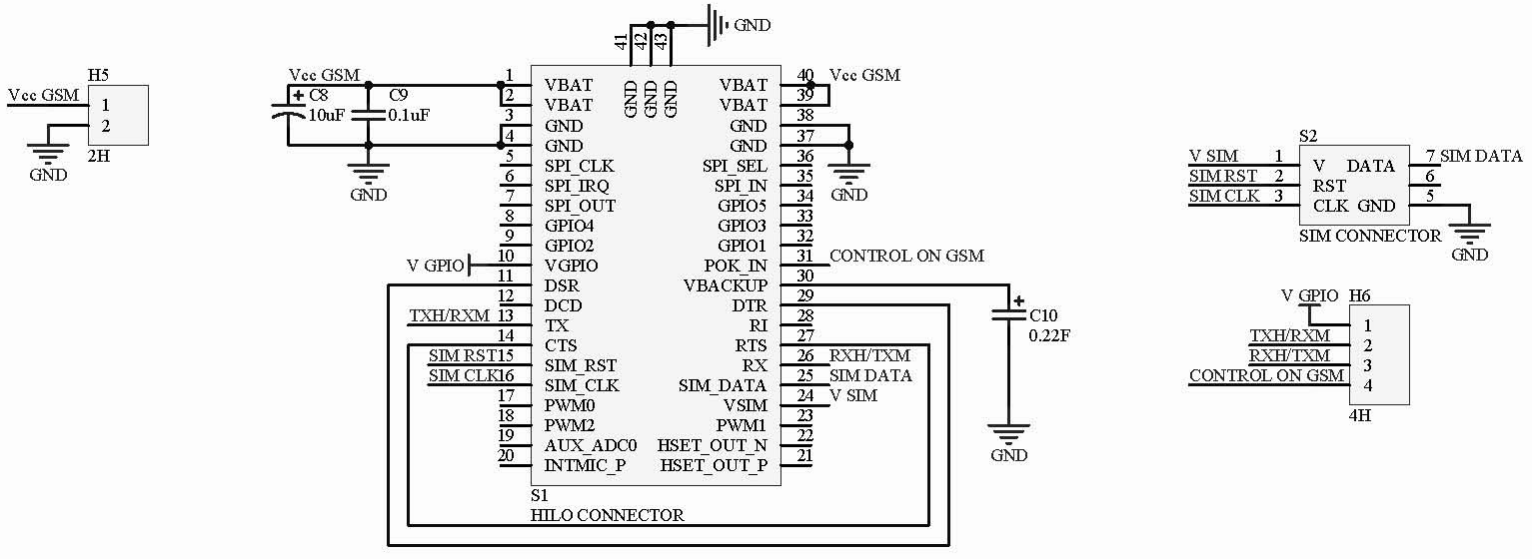


Ilustración 45: ESQUEMA COMPLETO DEL CIRCUITO DEL CONECTOR DEL MÓDULO GSM

µC

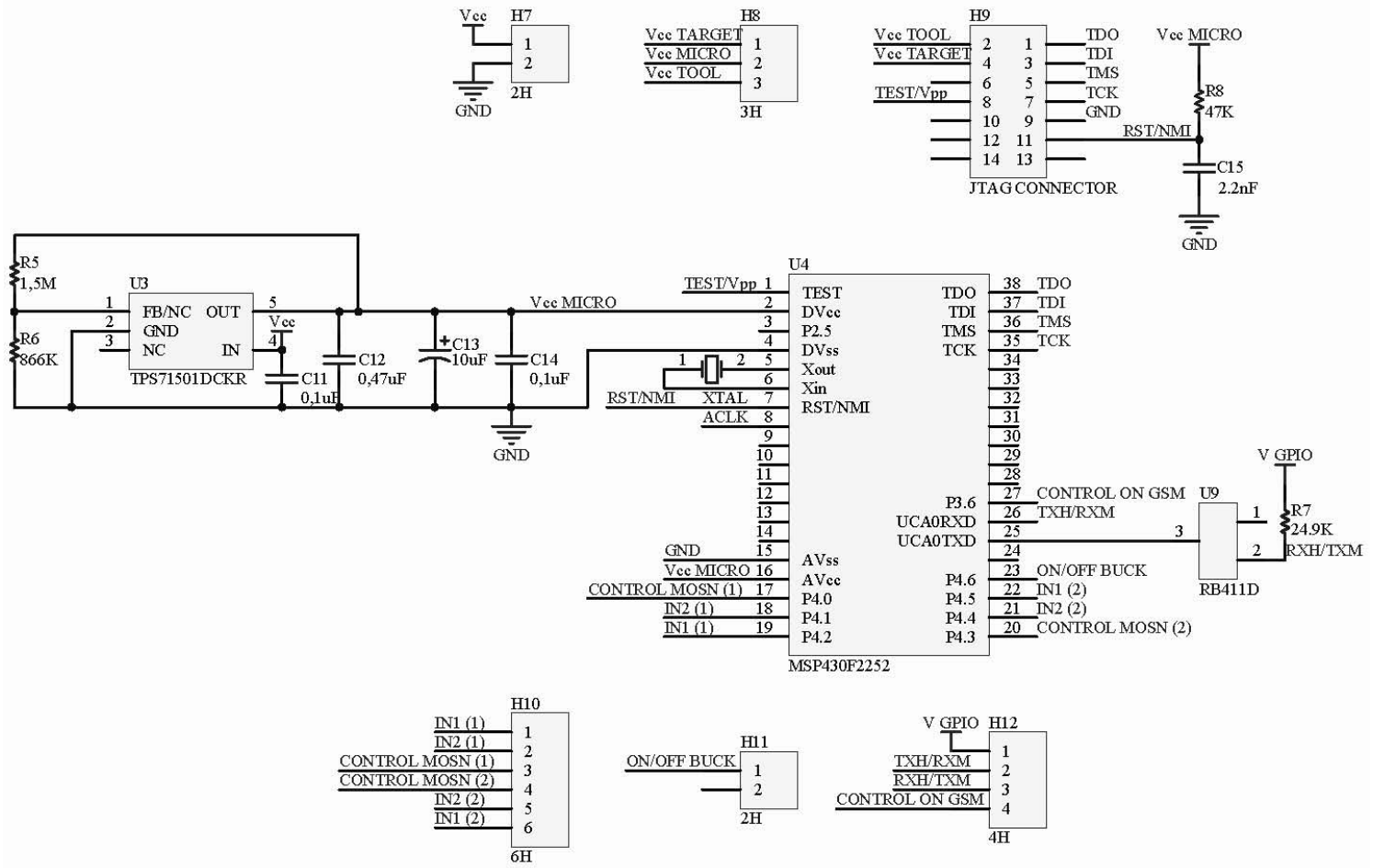


Ilustración 46: ESQUEMA COMPLETO DEL CIRCUITO DEL µC

ETAPA DE SALIDA

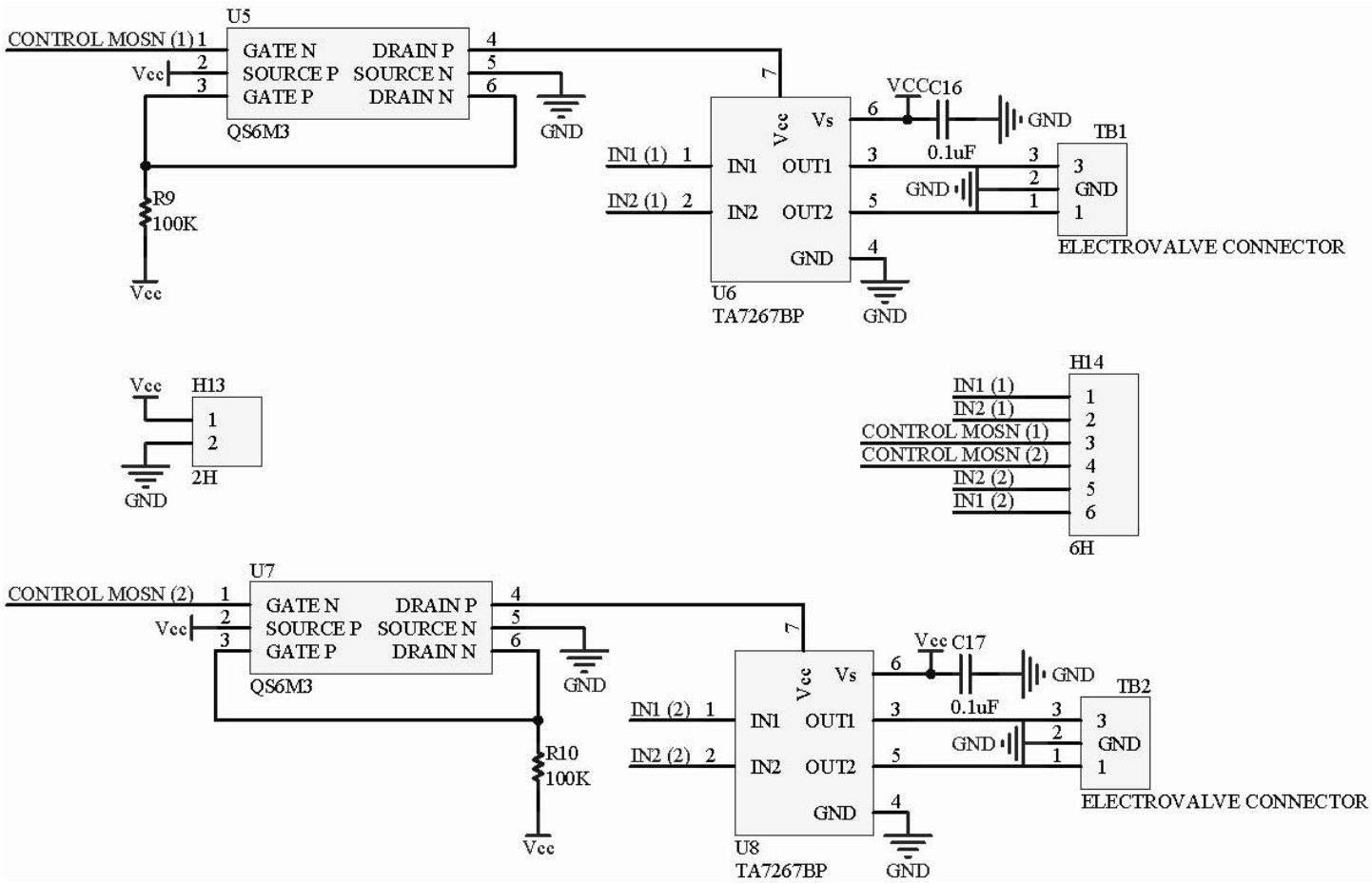


Ilustración 47: ESQUEMA COMPLETO DEL CIRCUITO DE LA ETAPA DE SALIDA

PLACA PROPORCIONADA POR LIBELIUM

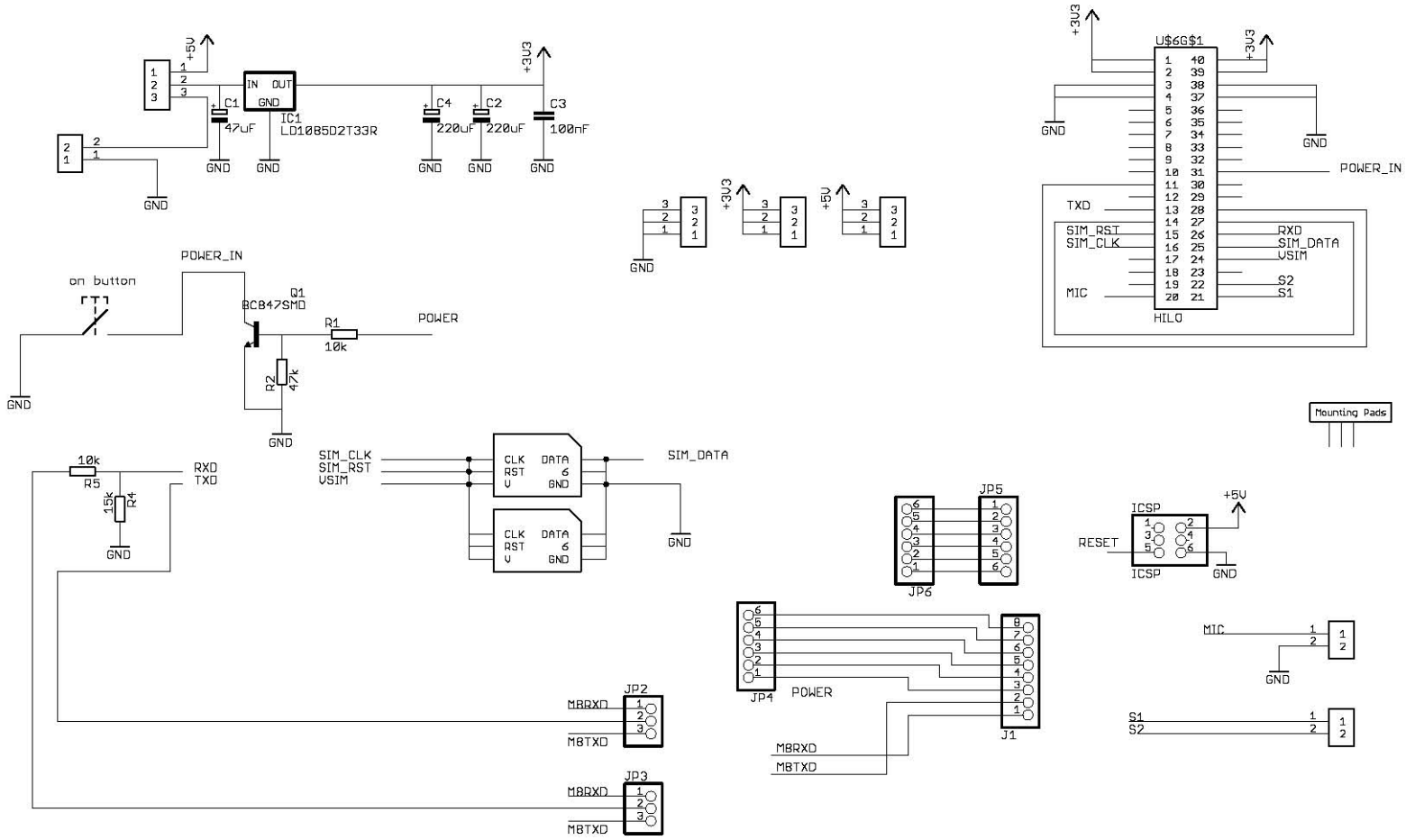


Ilustración 48: ESQUEMA DEL CIRUITO DE LA PLACA DE LIBELIUM

ANEXO Nº2: PLANOS DE PISTAS DE LAS PLACAS

Para el ruteo de las PCB'S, en general, se siguieron las siguientes normas con el fin de reducir las EMI:

- Evitar las pistas largas ya que actúan como antenas. Trazarlas lo más cortas posibles, de este modo se va a conseguir que capte/irradie una menor cantidad de ondas electromagnéticas del/al entorno.
- Con pistas cortas también conseguimos reducir el valor de la inductancia parásita, y con lo cual también la caída de tensión generada por estas:

$$\Delta V = L * \frac{di}{dt}$$

- Evitar los ángulos de 90º ya que de este modo también se reduce la longitud total de la pista.
- Evitar las pistas paralelas y largas ya que introducen una capacidad. Esta capacidad hace que una variación de tensión en una de las pistas sea transmitida a la otra.
- Reducir los bucles de corriente ya que estos actúan como espiras. Al circular corriente por una espira esta genera un campo magnético que produce corrientes inducidas sobre otros bucles próximos.

Según lo especificado en estos puntos cabe hacer las siguientes consideraciones para las líneas de alimentación y masa:

- Las líneas de alimentación se trazarán cortas y lo más juntas posibles.
- Se introducirán planos de masa para permitir caminos de retorno con la menor impedancia posible. Un plano de masa habilita líneas de retorno directas para las señales de baja frecuencia, minimiza el área del bucle de corriente para el retorno de señales de alta frecuencia y la impedancia común a varias señales.

Las últimas dos recomendaciones seguidas son:

- Se seguirá una distribución de la línea de alimentación y de masa a modo de estrella. De este modo se evita que las diferentes partes del circuito compartan impedancias y que, por tanto, las variaciones de tensión generadas por la corriente de retorno de un circuito afecten lo menos posible a los restantes.
- En cuanto a la disposición y valor de los condensadores de bypass se tuvo en cuenta lo indicado en el apartado 4.2.4 de la memoria.

FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

Para trazar las pistas, distribuir los componentes y el plano de masa se siguieron en la medida de lo posible las recomendaciones descritas en las hojas de características del LT3680.

PLANO DE PISTAS CARA TOP

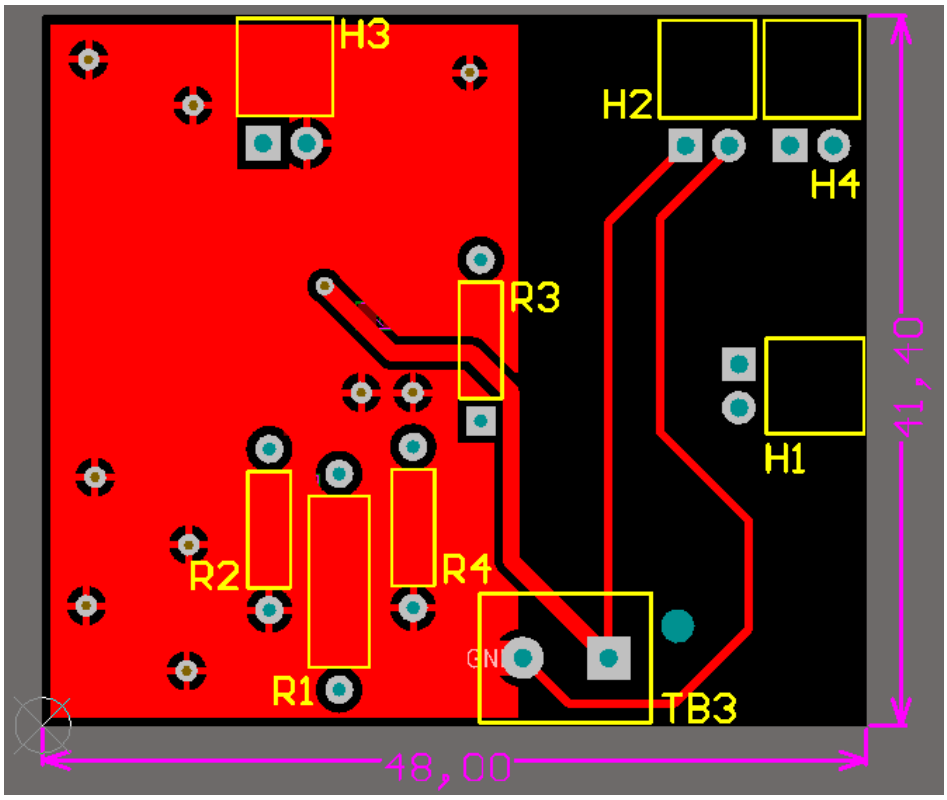


Ilustración 49: PLANO DE PISTAS CARA TOP DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

En esta cara de la PCB se colocaron los componentes THD para que al menos un plano de masa estuviera lo menos interrumpido posible.

PLANO DE PISTAS CARA BOTTOM

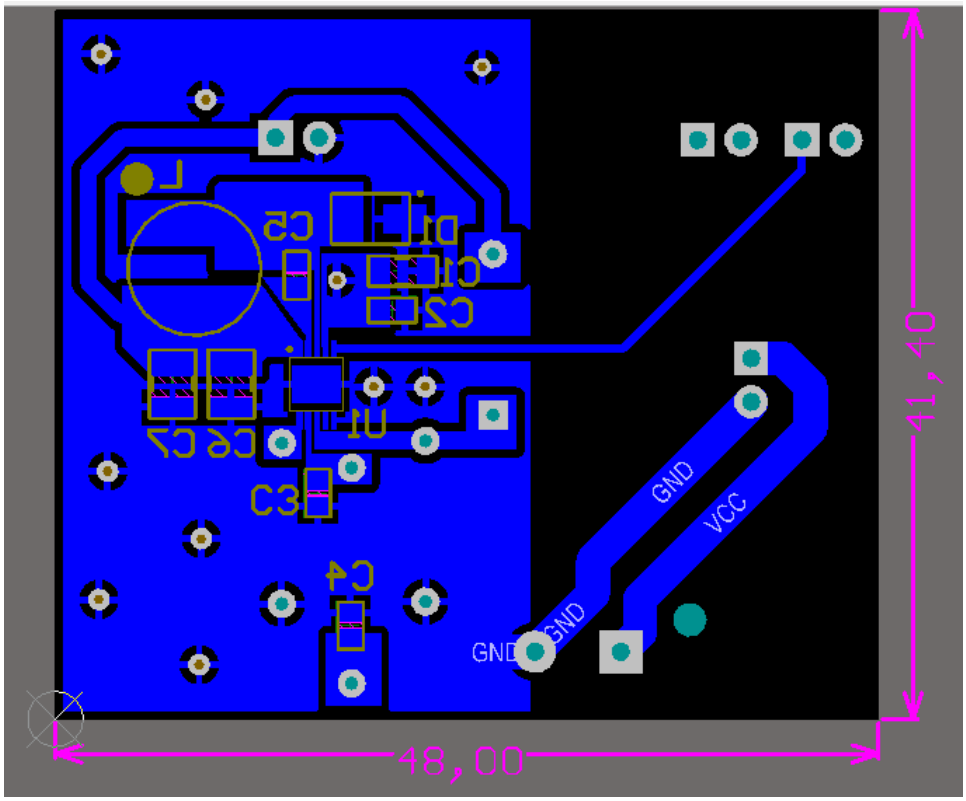


Ilustración 50: PLANO DE PISTAS CARA BOTTOM DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM

Las indicaciones descritas a continuación fueron las seguidas para la realización de esta cara de la PCB.

- Elevadas corrientes pulsantes fluyen por el condensador de entrada y por el diodo, por este motivo el bucle formado por estos componentes debe ser lo más pequeño posible.

- Los condensadores de entrada, los diodos, la bobina y los condensadores de salida deben colocarse en la misma cara y realizar las conexiones entre ellos en esa misma cara.

- Los nodos de los pines 2 (BOOST) y 3(SW) deben ser lo más pequeños posibles.

- Mantener los nodos 8 (FB) y 9 (Vc) pequeños para que puedan ser apantallados de SW y BOOST mediante el plano de masa.

- El pin de debajo del LT3680 actúa como un disipador de calor. Colocar vías adyacentes al LT3680 y extender al máximo posible el plano de masa para mantener la resistencia térmica baja.

MÓDULO GSM

PLANO DE PISTAS CARA TOP

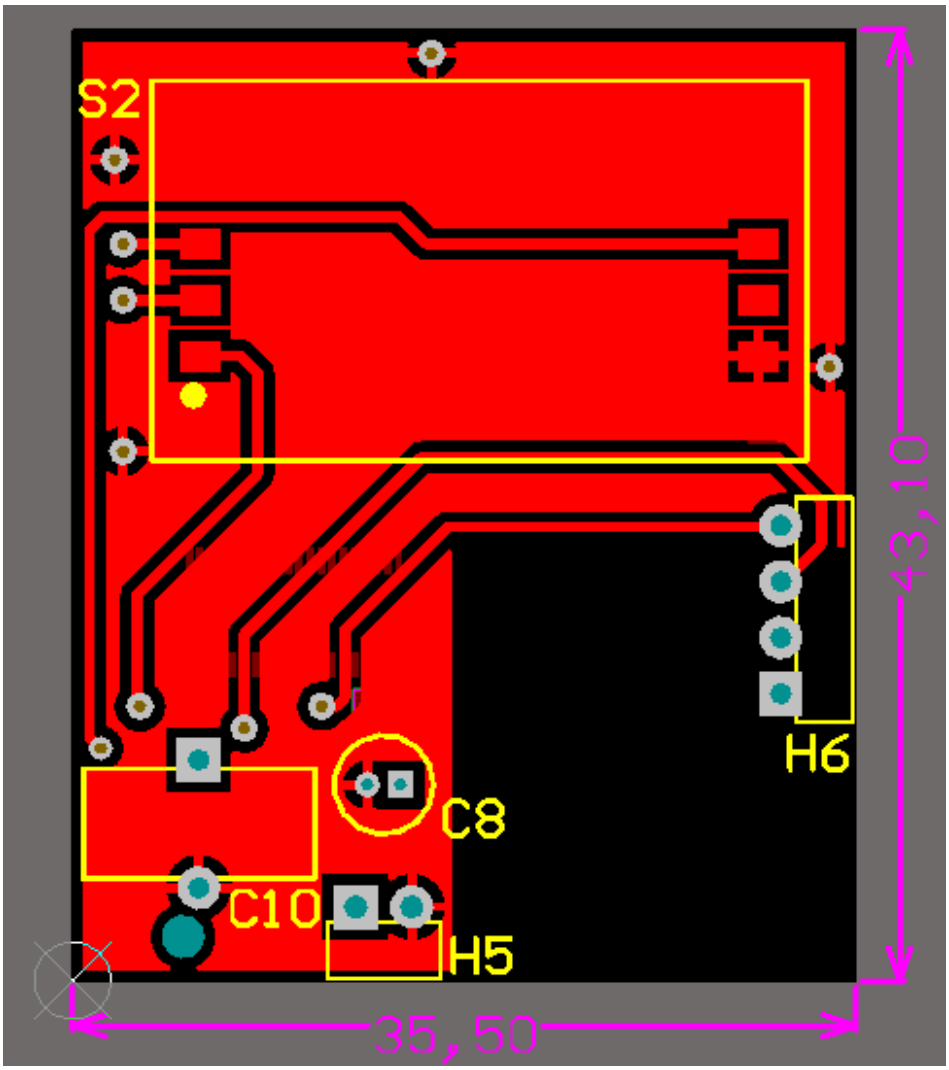


Ilustración 51: PLANO DE PISTAS CARA TOP DEL MÓDULO GSM

PLANO DE PISTAS CARA BOTTOM

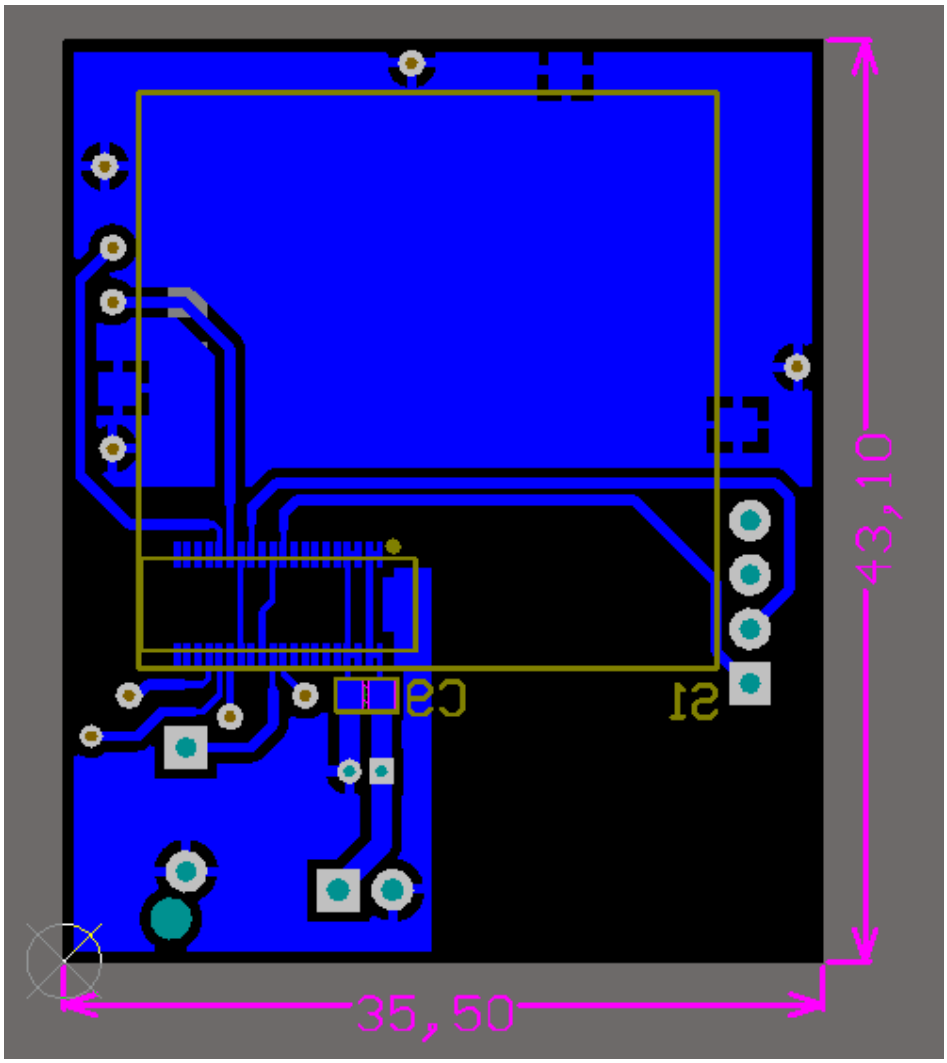


Ilustración 52: PLANO DE PISTAS CARA BOTTOM DEL MÓDULO GSM

µC

PLANO DE PISTAS CARA TOP

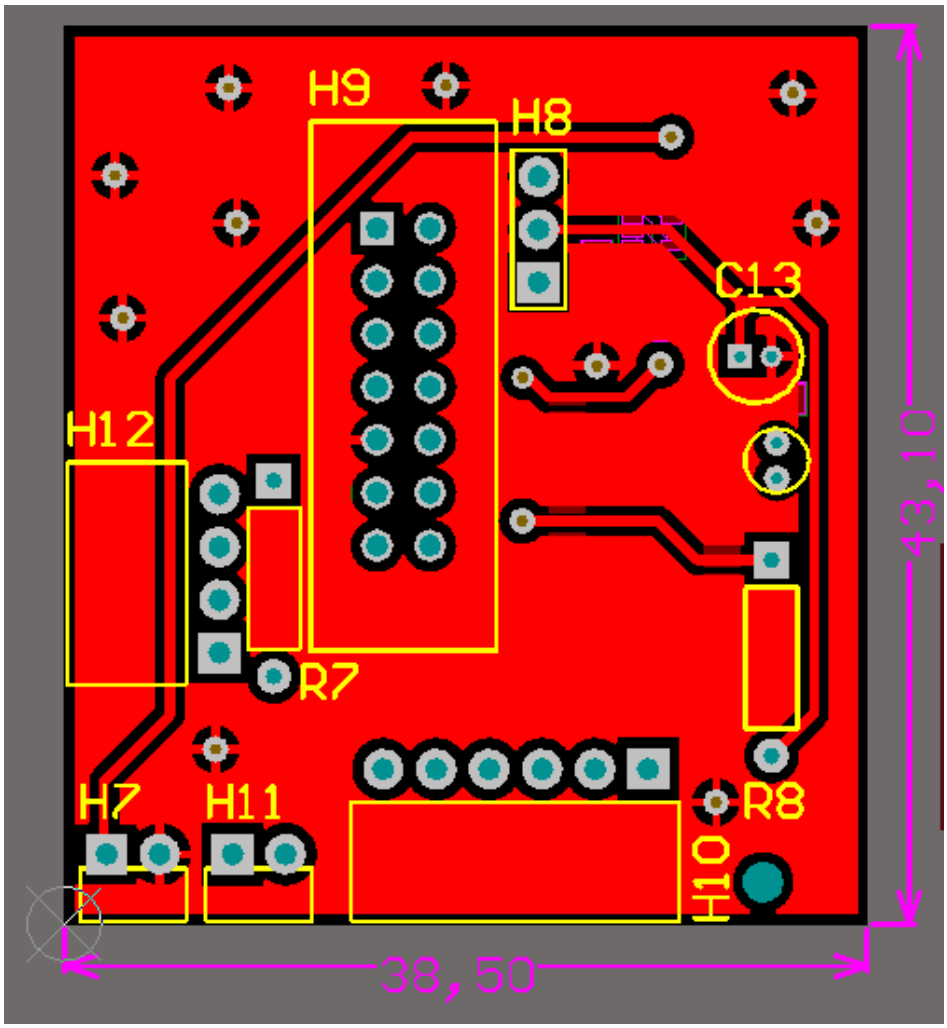


Ilustración 53: PLANO DE PISTAS CARA TOP DEL µC

PLANO DE PISTAS CARA BOTTOM

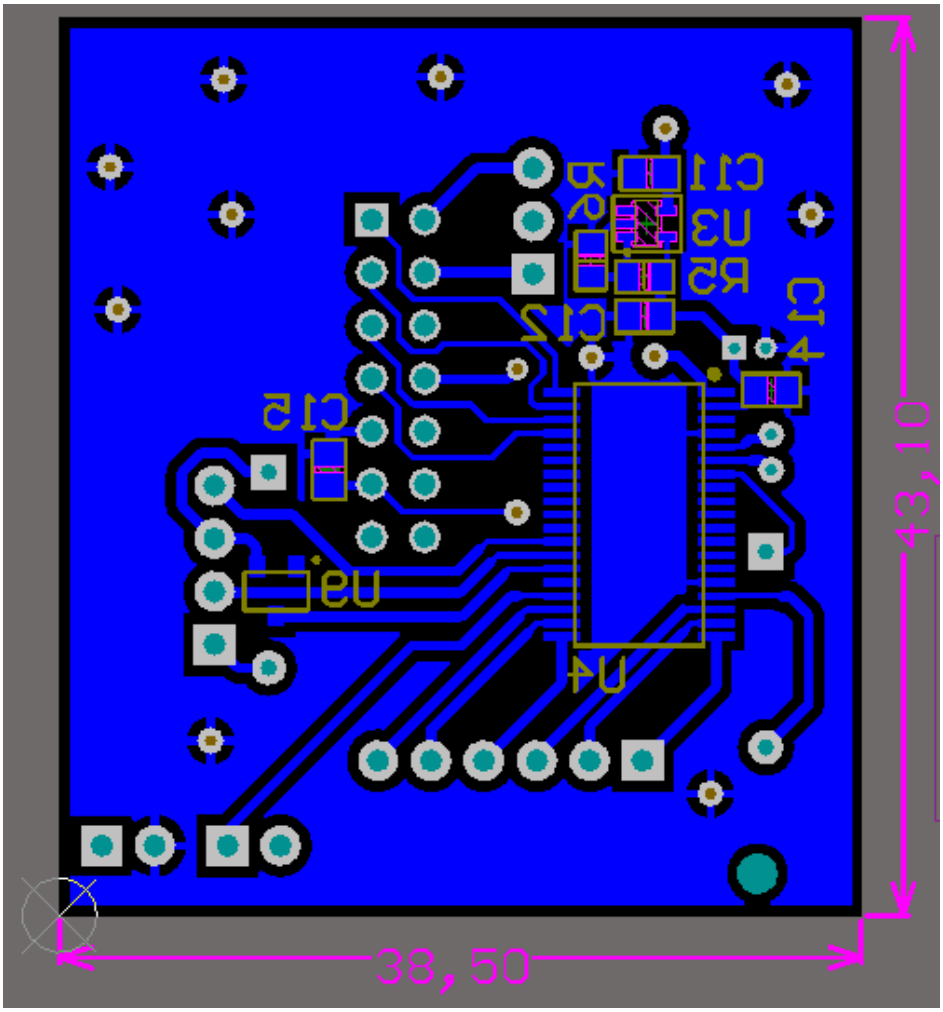


Ilustración 54: PLANO DE PISTAS CARA BOTTOM DEL µC

ETAPA DE SALIDA

PLANO DE PISTAS CARA TOP

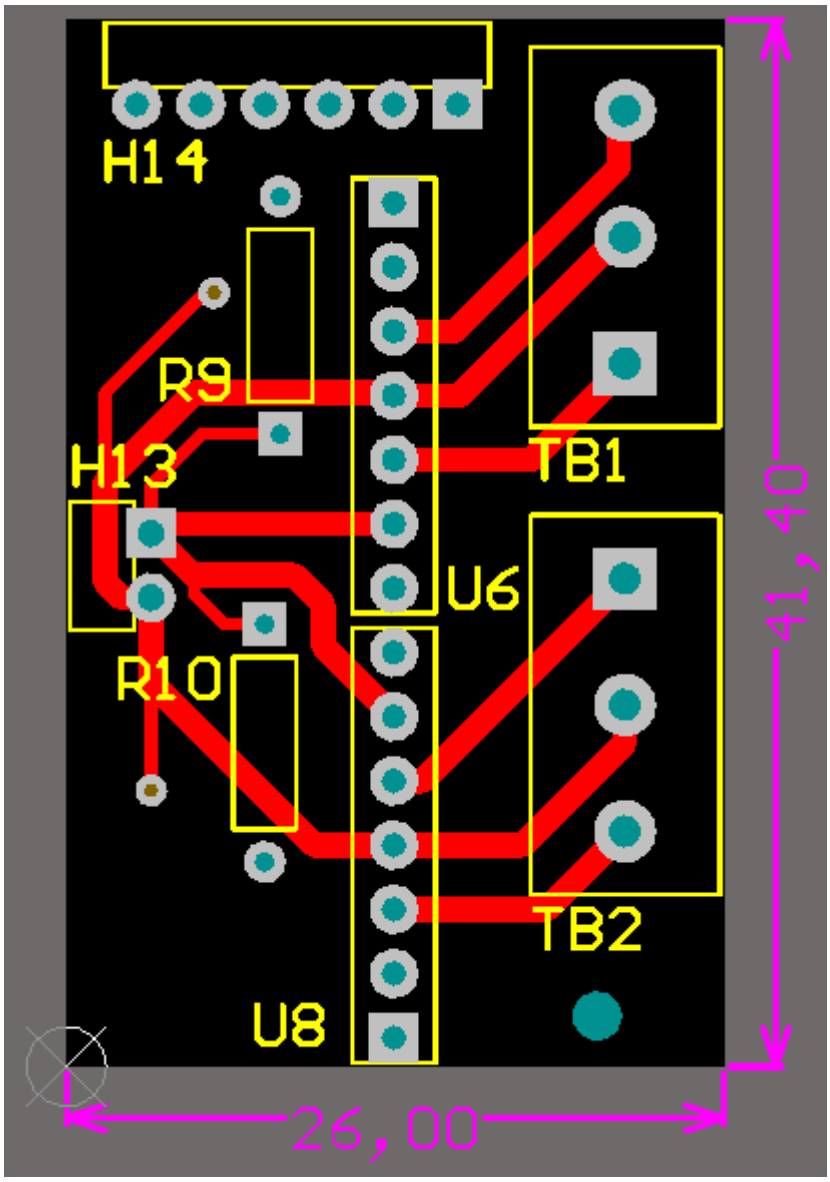


Ilustración 55: PLANO DE PISTAS CARA TOP DE LA ETAPA DE SALIDA

PLANO DE PISTAS CARA BOTTOM

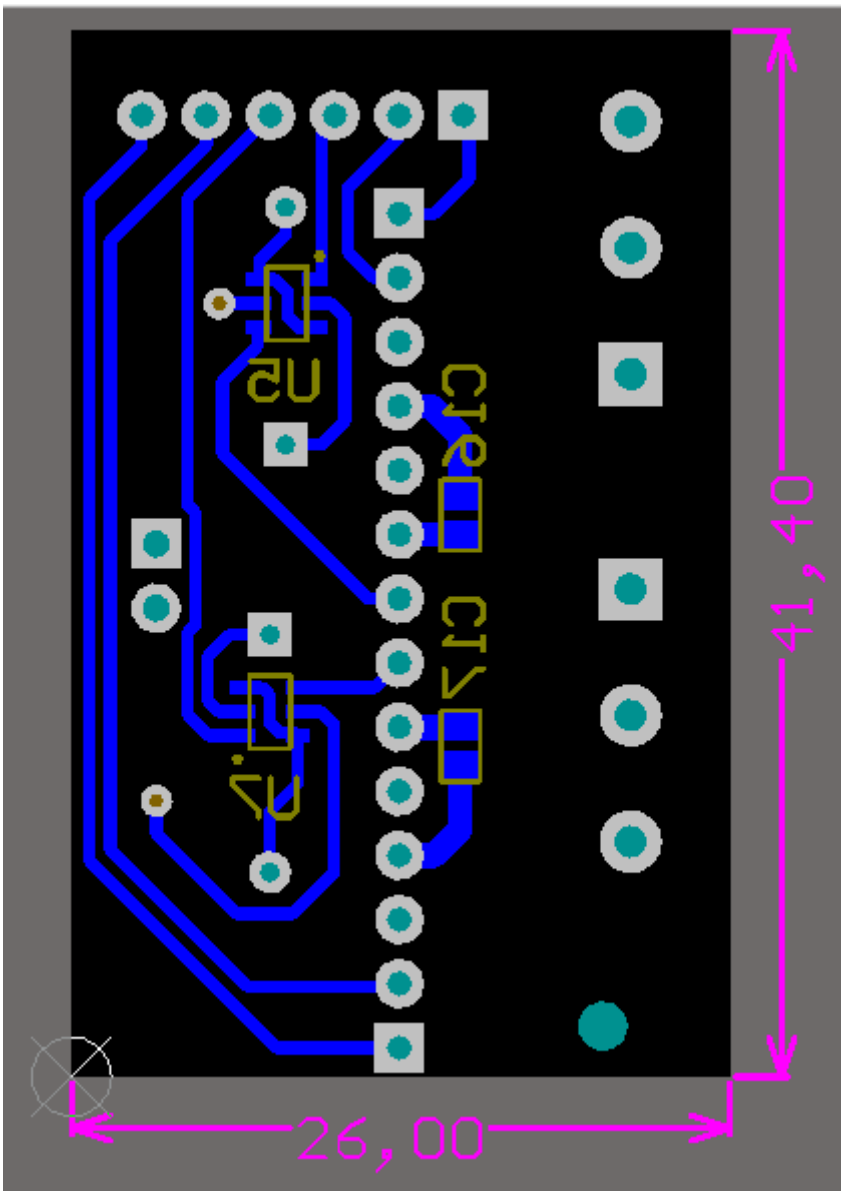


Ilustración 56: PLANO DE PISTAS CARA BOTTOM DE LA ETAPA DE SALIDA

ANEXO Nº3: COMANDOS AT

INTRODUCCIÓN

Fueron desarrollados en 1977 por Dennis Hayes como una interfaz de comunicación con un módem para poder configurarlo y proporcionarle instrucciones tales como marcar un número de teléfono. Más adelante fueron las compañías Micromm y US Robotics las que siguieron desarrollando y expandiendo el juego de comandos hasta universalizarlo.

Se denominan así por la abreviatura de la palabra inglesa attention.

Aunque fueron creados para posibilitar la comunicación entre un módem y el hombre, la telefonía móvil también ha adoptado como estándar este lenguaje. Todos los teléfonos móviles GSM y modem GSM/GPRS poseen un juego de comandos AT específico que sirve de interfaz para configurarlos y proporcionarles instrucciones tales como: realizar llamadas de datos o de voz, leer y escribir en la agenda de contactos, enviar mensajes de texto o multimedia…

Hay que tener en cuenta que todos los teléfonos móviles, módems GPRS/GSM no tienen porque tener implementados todos los comandos AT posibles, ni cada uno de los parámetros que forman el comando, ni aceptar todos los valores que pueden tomar dichos parámetros ni tampoco responder de la misma forma a un comando.

SINTAXIS DE LOS COMANDOS AT

Según su sintaxis existen dos tipos de comandos AT, básicos y extendidos. Como todos los utilizados en este proyecto son extendidos excepto uno (AT&W), a continuación se va a proceder a realizar una explicación detallada únicamente de la sintaxis de los extendidos.

Las normas son las siguientes:

- Todo comando debe empezar con AT y terminar con CR (retorno de carro).

  Por ejemplo AT+CPMS="ME","SM","ME"<CR>. El retorno de carro se introduce mediante la tecla enter y permite que el cursor regrese a la izquierda de la pantalla en HyperTerminal, nunca se va a visualizar en la pantalla del ordenador. En cambio para hacerle llegar un CR al módulo GSM desde el µC se debe enviar vía UART el carácter ASCII CR cuyo valor en hexadecimal es 0x13.

- En una línea se puede introducir más de un comando.

  Sólo el primer comando debe llevar la abreviatura AT y ambos comandos deben estar separados por punto y coma.

  Un ejemplo sería AT+CMGD=1;+CPMS?<CR>

- Si alguno de los parámetros que forman parte del comando es una cadena de caracteres está debe ir entre comillas.

  AT+CPMS="ME","SM","ME"<CR>

POSIBLES RESPUESTAS

Si el comando se ha ejecutado satisfactoriamente la respuesta será:

- <CR><LF>OK<CR><LF> (*final result code*).

  AT+CMGD=1<CR>

  <CR><LF>OK<CR><LF>. LF significa línea completa y hace que el cursor se desplace a la misma posición de la siguiente línea en HyperTerminal, al igual que CR LF es un carácter de los denominados no imprimibles por lo que no se va a visualizar en la pantalla del ordenador. LF será recibido por el µC vía UART codificado en ASCII, 0x10 en hexadecimal.

- <CR><LF>Una línea de información <CR><LF>(information response) junto a un <CR><LF> OK<CR><LF>.

  AT+CPMS="SM","SM","SM" <CR>

  <CR><LF>+CPMS: 2,50,2,50,2,50 <CR><LF>

  <CR><LF>OK<CR><LF>

Si el comando ha sido introducido incorrectamente o ha habido algún error en su ejecución los posibles *final result codes* serán:

- <CR><LF>+CMS ERROR:<nº de error><CR><LF>, si el comando introducido corresponde a alguno de los utilizados para el manejo de las prestaciones relacionadas con los SMS.

  AT+CMGD=0<CR>

  +CMS ERROR: 321

- <CR><LF>+CME ERROR:<nº de error><CR><LF>, indica errores relacionados con la SIM, la red, el propio módulo…

  AT+CPIN=”4923”<CR>

  <CR><LF>+CME ERROR: 3<CR><LF>, indica que el PIN ya se ha insertado.

- <CR><LF>ERROR<CR><LF>, se recibe cuando la sintaxis del comando es incorrecta, los parámetros introducidos en el comando no son válidos o el comando introducido no es soportado. Puede ser recibido en lugar del anterior si así se desea.

  AT_CMGD=1<CR>

  <CR><LF>ERROR<CR><LF>

Si en una misma línea se escriben varios comandos:

- Tantas líneas de *information response* como comandos se hayan introducido y un OK si todos los comandos se ejecutan de forma satisfactoria.
- Si el primer comando introducido no se ejecuta correctamente se recibirá como única respuesta alguno de los 3 *final result codes* de error.
- Si se introducen tres comandos en una misma línea y es el segundo el que se ejecuta de forma errónea se recibirá una línea de *information response* correspondiente al primer comando seguida de un final *result code* de error. En cambio si es el tercero el que no se ejecuta de forma correcta se recibirán dos líneas de *information response* seguidas por un *final result code* de error.

También los llamados *unsolicited result codes* son recibidos desde el módulo GSM para proporcionar información acerca de la ocurrencia de un evento, como por ejemplo la recepción de un SMS. En mi caso no he activado ningún aviso por tanto no voy a recibir ningún *unsolicited result codes*.

Las respuestas anteriores han sido detalladas de forma verbal. Si mediante el comando ATV[<value>] se configura la respuesta de forma que esta sea recibida numéricamente el <CR><LF>OK<CR><LF> se remplazaría por 0<CR> y el <CR><LF>ERROR<CR><LF> por 4<CR>.

Los CR y LF recibidos desde el módulo no son importantes para trabajar con un programa como HyperTerminal pero si lo son cuando trabajamos con el µC ya que son caracteres formados por bits de datos que se van a recibir a través de la UART.

Por simplificación a partir de ahora se omitirán los CR y LF.

TIPOS DE COMANDOS AT

Hay cuatro tipos según la operación que se lleve a cabo con ellos:

- *Test command*: utilizado para conocer si un comando AT es soportado por el teléfono móvil o módem GPRS/GSM. Si es soportado se recibirá una línea de *information response* en la que podremos observar los diferentes parámetros y los valores que podemos asignar o pueden tomar dichos parámetros (si el comando AT está formado por algún parámetro) seguida de un OK. Si no es soportado se recibirá ERROR.

  AT+CPMS=?

  +CPMS: ("SM","ME"),("SM","ME"),("SM","ME")

  OK

- *Write command*: son aplicados para cambiar la configuración del teléfono móvil o módem GPRS/GSM. Los parámetros que están entre corchetes son opcionales, si no se introducen seguirán adoptando su valor por defecto.

  AT+CPMS=<mem1>[,<mem2>[,<mem3>]]

  AT+CPMS="ME","SM","ME"

  +CPMS: 0,100,0,50,0,100

  OK

- *Read command*: se emplea para leer la configuración actual del teléfono móvil o módem GPRS/GSM.

  AT+CPMS?

  +CPMS: "SM",1,50,"SM",1,50,"SM",1,50

  OK

- *Execute command*: se usa para realizar una acción o para obtener información acerca del estado del teléfono móvil o módem GPRS/GSM.

  AT+CMGD=0,4

  OK

DETALLE DE LOS COMANDOS AT UTILIZADOS

PR COMMAND: SET FIXED LOCAL RATE

Sintaxis	Respuesta
AT+IPR?	**+IPR: <rate>**
	OK

Se obtiene el baud rate al que está configurado el módulo GSM.

Sintaxis	Respuesta
AT+IPR=<rate>	**OK**

Sustituir <rate> por el valor compatible deseado.

+CREG COMMAND: NETWORK REGISTRATION

Sintaxis	Respuesta
AT+CREG?	**+CREG: <n>,<stat>[,<lac>,<ci>]**
	OK

Los parámetros que me interesan de este comando son <n> y <stat>:

- **<n>:**

  0: deshabilita la recepción de un *unsolicited result code* cuando hay un cambio en el registro, es decir, cuando hay un cambio en <stat>.

  1: habilita la recepción del *unsolicited result code* +CREG: <stat>

  2: habilita la recepción del *unsolicited result code* +CREG: <stat>[,<lac>,<ci>]

- **<stat>:**

  0: no registrado, el dispositivo en este momento no está buscando un nuevo operador para registrarse.

. 1: registrado, *home network.*

  2: no registrado, el dispositivo está buscando un nuevo operador para registrarse.

  3: registro denegado.

  4: estado desconocido.

  5: registrado, *roaming.*

+CMGF COMMAND: SELECT SMS MESSAGE FORMAT

Sintaxis	Respuesta
AT+CMGF?	**+CMGF: <mode>** **OK**

- **<mode>:**

  0: modo PDU

  1: modo texto

Sintaxis	Respuesta
AT+CMGF=<mode>	**OK**

+CSCS COMMAND: SET TE CHARACTER SET

Selecciona el tipo de carácter al que va a ser codificado toda cadena de caracteres como agenda, SMS…

Sintaxis	Respuesta
AT+CSCS=<chset>	**OK**

- **<chset>:**

  "GSM": *GSM 7-bit default alphabet*. Debe aparecer como opción en todo módulo.

  "UCS2": *16 bit universal multiple-octet coded character set*.

  "IRA": valor por defecto.

+CPMS COMMAND: PREFERRED MESSAGE STORAGE

Sintaxis	Respuesta
AT+CPMS=?	**+CPMS:**(list of supported<**mem1**>), (list of supported<**mem2**>), (list of supported<**mem3**>) **OK**

Este comando nos permite conocer qué tipos de memoria tiene disponible nuestro módulo GSM, en el caso del HILO se puede elegir entre las mismas dos opciones para <**mem1**>, <**mem2**> y <**mem3**>**.** Estas son:

- SM: área de almacenamiento localizada en la tarjeta SIM.
- ME: área de almacenamiento en el módulo GSM. Usualmente su capacidad es mayor que la proporcionada por la de la tarjeta SIM.

Sintaxis	Respuesta
AT+CPMS=<mem1> [,<mem2>[,<mem3>]]	**+CPMS:<used1>,<total1>,<used2>, <total2>,<used3>,<total3>** **OK**

Seleccionamos el área de almacenamiento que utilizaremos cuando recibamos, leamos, enviemos o borremos SMS.

- **<mem1>:** Memoria desde la que se leen y borran los mensajes, por tanto se utiliza con los comandos AT+CMGR, AT+CMGL o AT+CMGD.
- **<mem2>:** Utilizada para enviar mensajes guardados o para escribirlos. Los comandos que van a emplearla serán AT+CMGW y AT+CMSS.
- **<mem3>:** En la memoria que se inscriba en esta posición se guardarán los nuevos mensajes recibidos.
- **<totalX>:** Nos indica el número total de SMS que puede almacenar la memoria X.
- **<usedX>:** Hace referencia al número de SMS que hay guardados en la memoria X.

Sintaxis	Respuesta
AT+CPMS?	**+CPMS: <mem1>,<used1>,<total1>, <mem2>, <used2>, <total2>, <mem3>, <used3>, <total3>** **OK**

Nos indica las memorias que estamos utilizando, el número de SMS que hay guardados actualmente en cada una de ellas y la máxima cantidad de estos que puede almacenar.

+CSDH COMMAND: SHOW TEXT MODE PARAMETERS

Sintaxis	Respuesta
AT+CSDH?	**+CSDH: <show>** **OK**

- **<show>:**

  0: no muestra parámetros adicionales. Valor por defecto.

  1: si los muestra.

Escogiendo el valor cero se consigue que la repuesta a diferentes comandos no incluya parámetros que pueden ser innecesarios. Por ejemplo para el comando AT+CMGR, que es el que nos permite leer SMS, prescindimos de los siguientes:

- <sca>: nombre introducido en la agenda de la persona que envía el SMS).
- <tosca>: número que indica el tipo de número del centro de servicio. Si es un número internacional tomará el valor 145, si es otro tipo de número el 129.
- <vp>: validez del período del SMS.
- <pid>: identificador del protocolo utilizado. Indica el tipo de comunicación (si se está enviando un SMS, haciendo una llamada de teléfono…)
- <dcs>: tipo de alfabeto utilizado.
- <length>: longitud del SMS.
- <tooa>: tipo de número de teléfono de la dirección de origen.

+CNMI COMMAND: NEW SMS MESSAGE INDICATION

Indicamos al módulo GSM cómo debe actuar ante la recepción de un SMS.

Sintaxis	Respuesta
AT+CNMI?	**+CNMI:<mode>,<mt>,<bm>,**
	<ds>,<bfr>
	OK

- **<mode>:**

  0: Se guardarán los *unsolicited result codes* en el TA. Si este buffer está lleno las indicaciones serán guardadas en otro lugar o se remplazarán las más viejas por las nuevas.

  1: Se rechazarán los unsolicited result codes cuando la vía de comunicación entre el TA y el TE se encuentre ocupada, si no lo está se transmitirá directamente al TE.

  2: Si la conexión entre el TA y el TE no está disponible los *unsolicited result codes* serán guardados en el TA y cuando sea posible transmitidos al TE, si no se enviarán directamente al TE.

- **<mt>:**

  0: cuando un nuevo SMS es recibido no se le proporciona ningún tipo de indicación al TE.

  1: cuando se recibe un nuevo SMS se envía un *unsolicited result code* al TE del tipo **+CMTI:<memory>,<index>**, indicando la memoria en la que se ha guardado el SMS y la posición que ocupa dentro de esta.

  2: el SMS no se guarda en el módulo.

- **<bm>:**

  0: las indicaciones acerca de CBM no son remitidas al TE.

  2: las indicaciones acerca de CBM si son transmitidas al TE.

- **<ds>:**

  0: el TE no recibe ningún tipo De SMS-STATUS-REPORT que le permita conocer si el SMS ha llegado con éxito al destinatario.

  1: el TE recibe un SMS-STATUS-REPORT.

- **<bfr>:**

  0: Las notificaciones guardadas son enviadas.

  1: Las notificaciones guardadas en TA son eliminadas cuando **<mode>** toma el valor 1.

+CMGR COMMAND: READ SMS MESSAGE

Sintaxis	Respuesta
AT+CMGR=<index>	**+CMGR: "estado del SMS", "+Nº desde el que se envía el SMS","","yy/mm/dd,hh:mm:ss±zz"** **Cuerpo del mensaje** **OK**

- **<index>:** Posición de la memoria donde se encuentra almacenado el SMS.

- **Estado del SMS:**

  0 "REC UNREAD" (0 si utilizamos el modo PDU y REC UNREAD si se escoge el modo texto). Mensaje recibido y no leído.

  1 "REC READ". Mensaje recibido y leído.

  2 "STO USENT". Mensaje guardado pero no enviado.

  3 "STO SENT". Mensaje guardado y enviado.

- **Fecha y hora:**

  Yy: año.

  Mm: mes.

  Dd: día del mes.

  Hh: hora.

  Mm: minutos.

  Ss: segundos.

±Zz: desviación en cuartos de hora respecto a la marcada por GMT.

+CMEE COMMAND: REPORT MOBILE TERMINATION ERROR

Elegimos el tipo de respuesta que queremos obtener del módulo cuando se produce algún error de tipo CME al ejecutar algún comando AT.

Sintaxis	Respuesta
AT+CMEE?	**+CMEE: <n>**
	OK

- **<n>:**
- **0:** deshabilita la respuesta de tipo +CME ERROR: <err> y la sustituye por ERROR.
- **1:** cuando se produce un error se utiliza la respuesta +CME ERROR: <err>, donde este último toma valores numéricos.
- **2:** responde del mismo modo que el anterior pero <err> está representado de forma verbal.

Sintaxis	Respuesta
AT+CMEE=[<n>]	**OK**

+CMGS COMMAND: SEND SMS MESSAGE

Este comando envía el SMS pero no guarda ninguna copia de este.

Sintaxis	Respuesta
AT+CMGS=<da>[,<toda>]<CR>	**+CMGS: <mr>**
>Texto<ctrl-z>	**OK**

- **<mr>:** número de referencia del SMS asignado por el módulo.

Primero se tiene que introducir la primera línea, donde **<da>** es el nº de teléfono y **<toda>** que es opcional, hace referencia a si el nº de teléfono está escrito en formato internacional con toda seguridad o no. Si no se escribe nada el módulo directamente coloca en este campo 145 si al nº de teléfono le precede un + (el número está en formato internacional) o 129 si no lo lleva (el nº de teléfono puede ser internacional o no).

Después de introducir estos dos parámetros se debe presionar la tecla enter o enviar del µC al GSM a través de la UART el número binario correspondiente al carácter CR.

Antes de escribir el cuerpo del SMS se debe esperar a que se reciba desde el módulo “>”. Cuando se desee enviar el SMS se tecleará ctrl-Z en HyperTerminal o se enviará a través de la UART el carácter ASCII SUB (0b00011010).

El envío del SMS se puede cancelar mediante escape (ESC), en el caso de cancelarlo el módulo responde con OK.

+CMGW COMMAND: WRITE SMS MESSAGE TO MEMORY

Sintaxis	Respuesta
AT+CMGW=<da><CR>	**+CMGW: <index>**
	>texto<ctrl-z>

El proceso a seguir para implementar el comando es exactamente el mismo que el indicado en el comando AT inmediatamente anterior.

- **<index>:** Indica la posición de la memoria en la que se ha guardado el SMS.

+CMSS COMMAND: SEND SMS MESSAGE FROM STORAGE

Antes de implementarlo se ha debido de guardar algún SMS en la memoria mediante el comando inmediatamente anterior.

Sintaxis	Respuesta
AT+CMSS=<index>[,<da>[,<toda>]]	**+CMSS: <mr>**

- **<index>:** Es el número entero positivo que nos ha devuelto el GSM al implementar el comando inmediatamente anterior.
- **<da>:** Corresponde al número de teléfono al que se va a enviar el SMS, al ir entre corchetes es opcional. Si aquí no se introduce ningún número se utilizará el incluido en el comando AT anterior.
- **<toda>:** Cabe hacer las mismas consideraciones que en AT+CMGS.

+CMGD COMMAND: DELETE SMS MESSAGE

Sintaxis	Respuesta
AT+CMGD=<index>[,<delflag>]	**OK**

- **<index>:** Número que le indica al módulo GSM la posición del SMS que deseamos borrar.
- **<deflag>:**

  0: si se introduce borramos únicamente el SMS indicado por **<index>**.

  1: se ignora **<index>** y se borran los SMS cuyo estado es rec read.

  2: se ignora **<index>** y se borran los mensajes con estados rec read y stored sent.

  3: en este caso también se ignora **<index>** pero se borran los SMS con estado sto unsent además de los SMS con estado indicado en 3.

4: se borran todos los SMS.

ANEXO Nº4: SISTEMAS DE CODIFICACIÓN DE CARACTERES ACEPTADOS POR EL MÓDULO GSM

El módulo hilo admite tres tipos de codificación de caracteres el GSM 7-BIT DEFAULT ALPHABET, el IRA y el UCS2.

CÓDIGO GSM-7 BIT DFAULT ALPHABET

Debe estar implementado obligatoriamente en todo móvil, módem o módulo GSM.

En el código estándar todos los caracteres se encuentran representados por 7 bits, con lo que se permite la codificación de 128 caracteres. Estos toman valores hexadecimales comprendidos entre 0x00 y 0x7F.

En el código extendido hay otros 10 caracteres adicionales que requieren de dos caracteres codificados en siete bits.

Es parecido al código ASCII utilizado por el µC. Pero difiere con este en que:

- Los caracteres de control son sustituidos en su mayoría por caracteres griegos y latinos.
- Los símbolos @, $, y _ toman valores hexadecimales diferentes.
- Carece del acento grave.
- Los caracteres [,], \, ^, {, }, |, ~,€ y el carácter de control form feed forman parte del código GSM extendido. Codificándose en hexadecimal como 0x1BXX, donde 1B corresponde a la codificación de ESC (escape to extensión table) y XX a la del carácter extendido.

Un SMS se encuentra limitado según la ETSI a 140 bytes, con lo cual el número máximo de caracteres que se pueden incluir en un SMS si todos corresponden al código GSM estándar será de 160 ya que estos se empaquetan en grupos de 8 bits (140*8=1120 bits, 1120/7=160). Si dentro del SMS hay alguno de los símbolos correspondientes al código GSM extendido la longitud será menor ya que estos se encuentran representados a través de 14 bits.

En la siguiente página se muestran dos tablas en las que aparece el valor hexadecimal que toma cada uno de los caracteres en este tipo de codificación.

Hex	0x	1x	2x	3x	4x	5x	6x	7x
x0	@	Δ	SP	0	¡	P	¿	p
x1	£	_	!	1	A	Q	a	q
x2	$	Φ	"	2	B	R	b	r
x3	¥	Γ	#	3	C	S	c	s
x4	è	Λ	¤	4	D	T	d	t
x5	é	Ω	%	5	E	U	e	u
x6	ù	Π	&	6	F	V	f	v
x7	ì	Ψ	'	7	G	W	g	w
x8	ò	Σ	(	8	H	X	h	x
x9	Ç	Θ	)	9	I	Y	i	y
xA	LF	Ξ	*	:	J	Z	j	z
xB	Ø	1)	+	;	K	Ä	k	ä
xC	ø	Æ	,	<	L	Ö	l	ö
xD	CR	æ	-	=	M	Ñ	m	ñ
xE	Å	ß	.	>	N	Ü	n	ü
xF	å	É	/	?	O	§	o	à

Ilustración 57: TABLA DEL CÓDIGO GSM 7-BIT DEFAULT ALPHABET

Hex	0x	1x	2x	3x	4x	5x	6x	7x
x0					|			
x1								
x2								
x3								
x4		^						
x5							€	
x6								
x7								
x8			}					
x9			{					
xA								
xB								
xC				[				
xD				~				
xE				]				
xF			\					

Ilustración 58: TABLA EXTENDIDA DEL CÓDIGO GSM 7-BIT DEFAULT ALPHABET

CÓDIGO IRA (ANTERIORMENTE IA5)

Tipo de codificación en el que los símbolos son representados mediante 7 bits pero codificados mediante 8, por lo tanto permitirá una longitud máxima de SMS de 140 caracteres. El octavo se puede utilizar de bit de paridad.

Si la versión utilizada es la IRV, la codificación de los caracteres es casi completamente igual a la respectiva del código ASCII. Varían únicamente los caracteres de control del 0x1C al 0x1F.

b_7	0	0	0	0	1	1	1	1
b_6	0	0	1	1	0	0	1	1
b_5	0	1	0	1	0	1	0	1
b_4 b_3 b_2 b_1	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	IS4	,	<	L	\	l	|
1 1 0 1	13	CR	IS3	–	=	M	]	m	}
1 1 1 0	14	SO	IS2	.	>	N	^	n	~
1 1 1 1	15	SI	IS1	/	?	O	_	o	DEL

TD811600-93

Ilustración 59: TABLA DEL CÓDIGO IRA (VERSIÓN IRV)

CÓDIGO UCS2

Los caracteres en este tipo de codificación se representan mediante 16 bits (2 bytes), pudiéndose introducir en un SMS únicamente 70 caracteres. Este código permite la representación de caracteres de alfabetos como el coreano, el chino, el árabe, el ruso o el japonés.

ANEXO Nº5: DIAGRAMAS DE BLOQUES DEL PROGRAMA

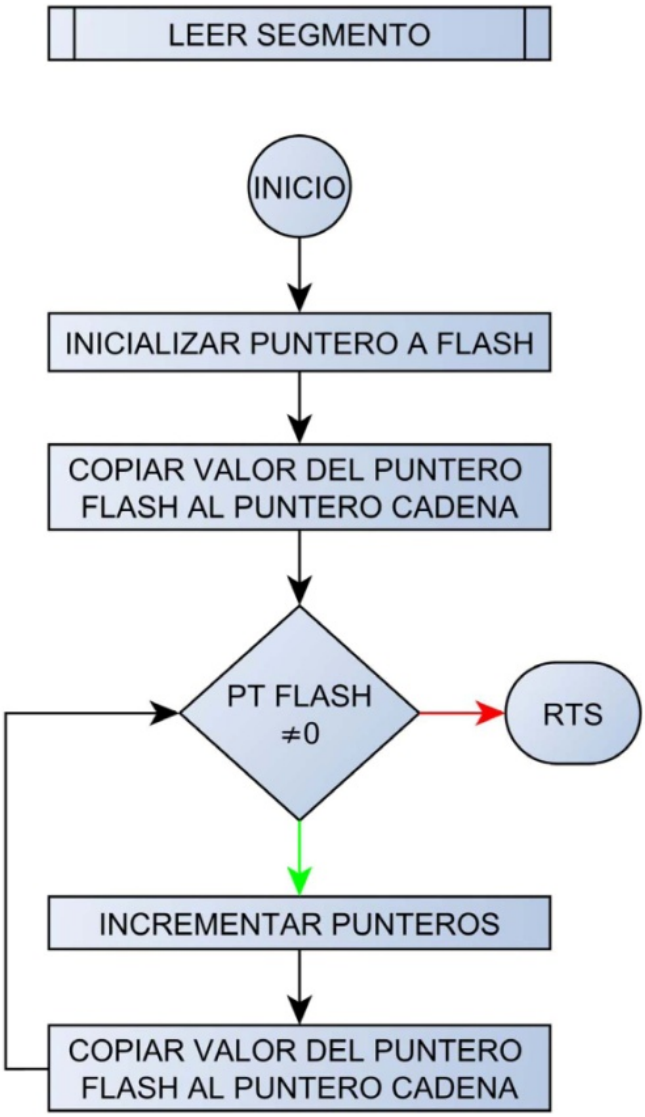


Ilustración 60: LEER SEGMENTO

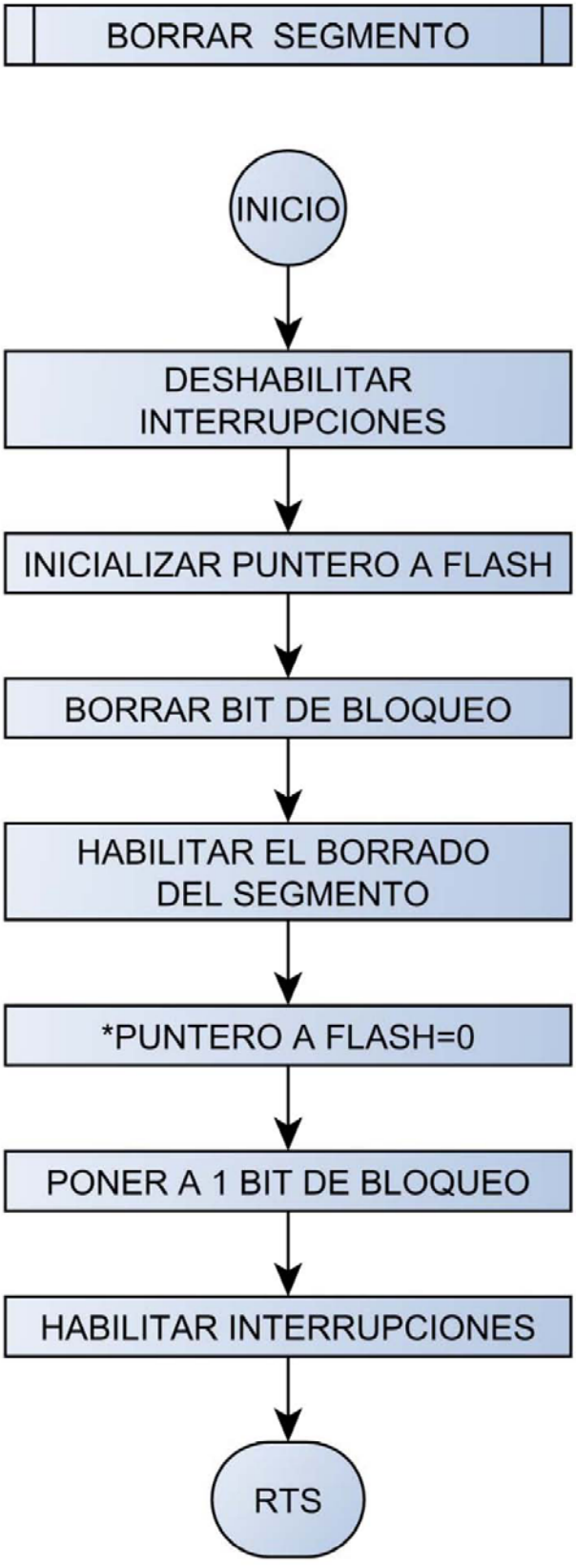


Ilustración 61: BORRAR SEGMENTO

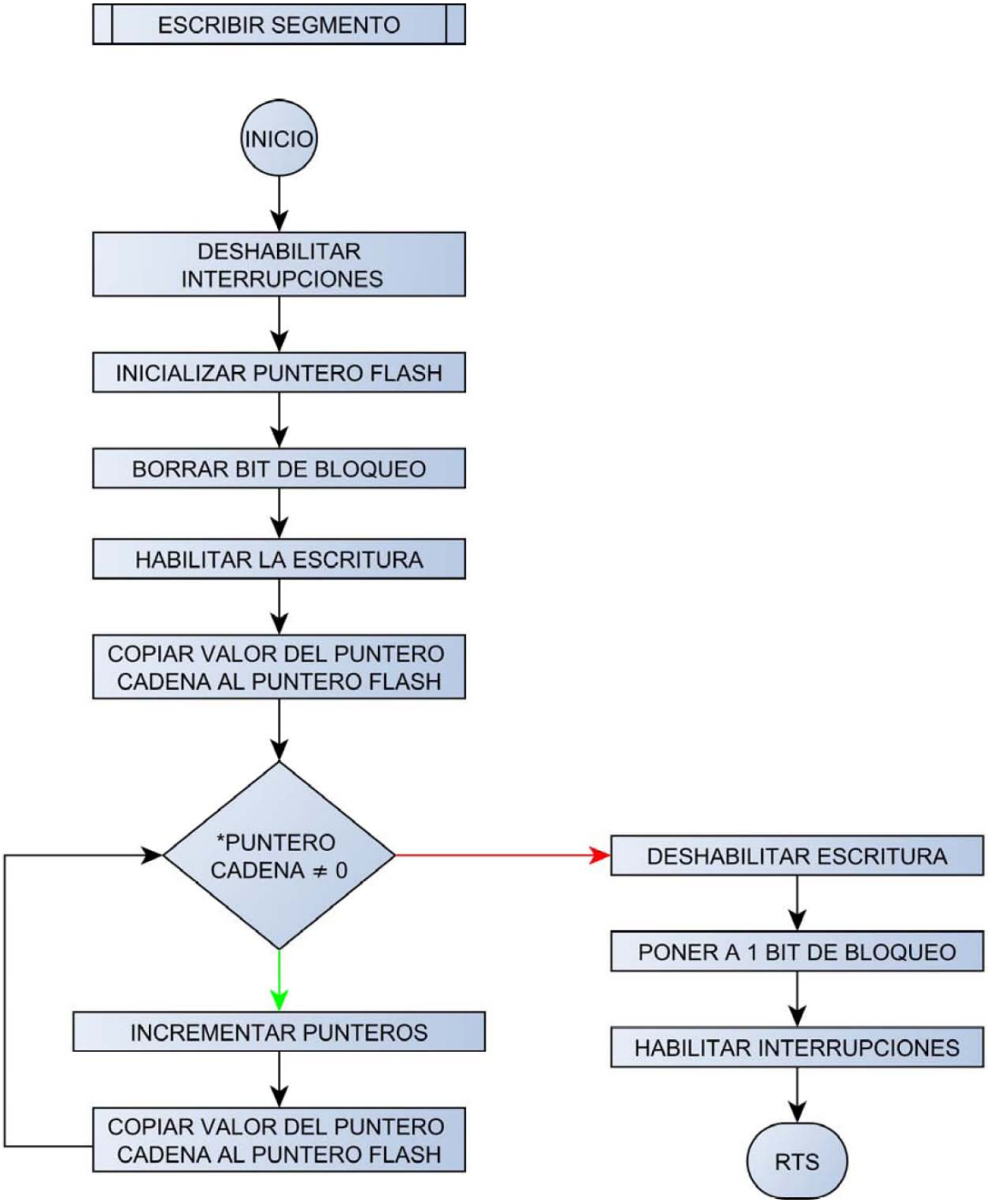


Ilustración 62: ESCRIBIR SEGMENTO

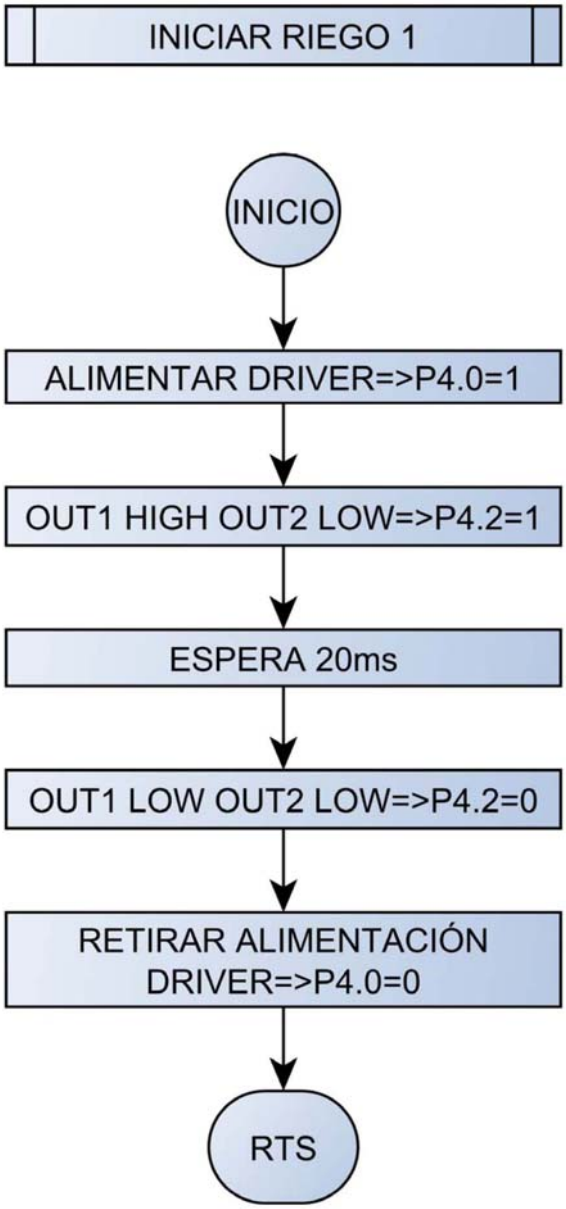


Ilustración 63: INICIAR RIEGO 1

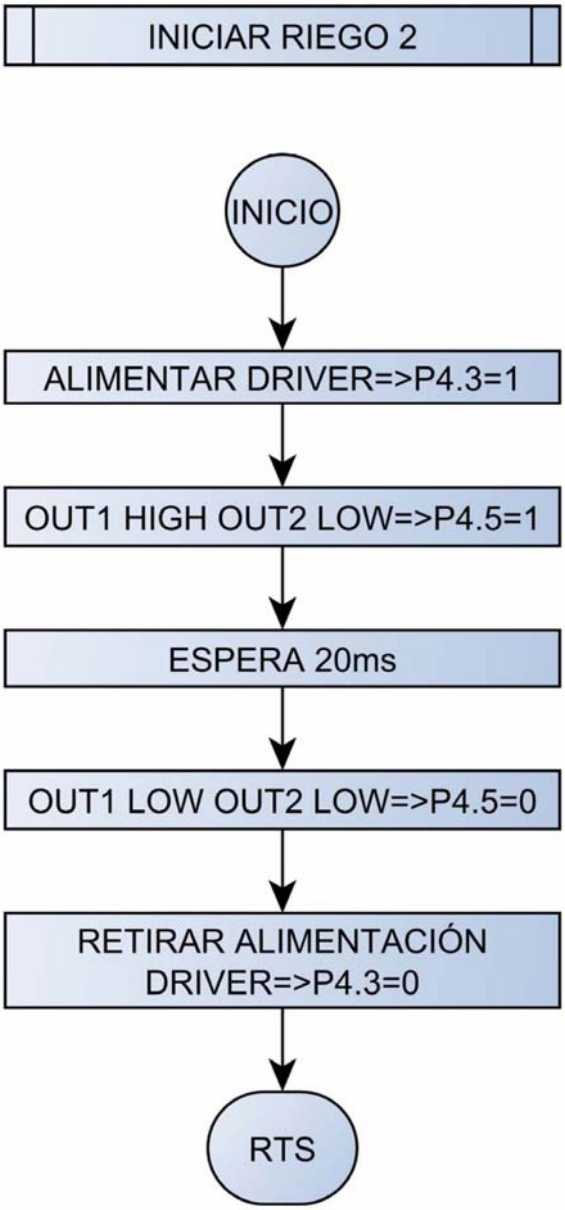


Ilustración 64: INICIAR RIEGO 2

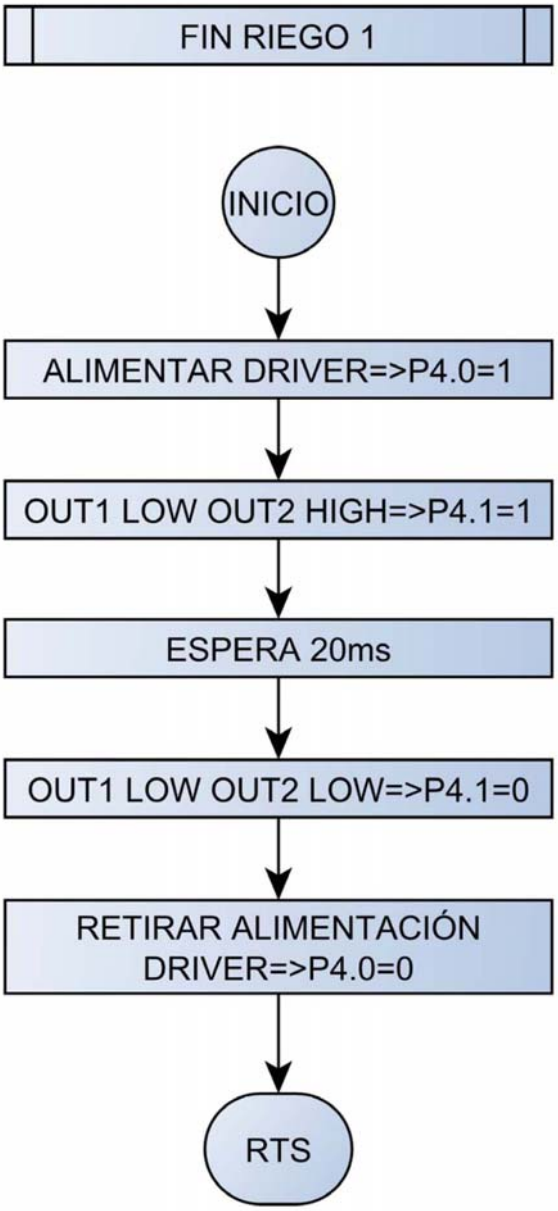


Ilustración 65: FIN RIEGO 1

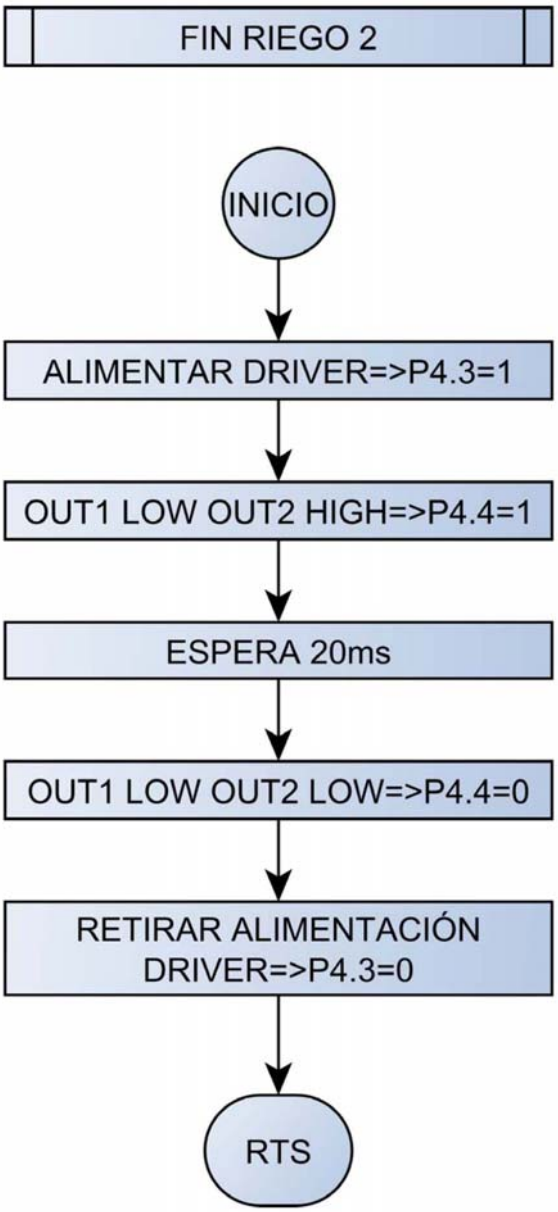


Ilustración 66: FIN RIEGO 2

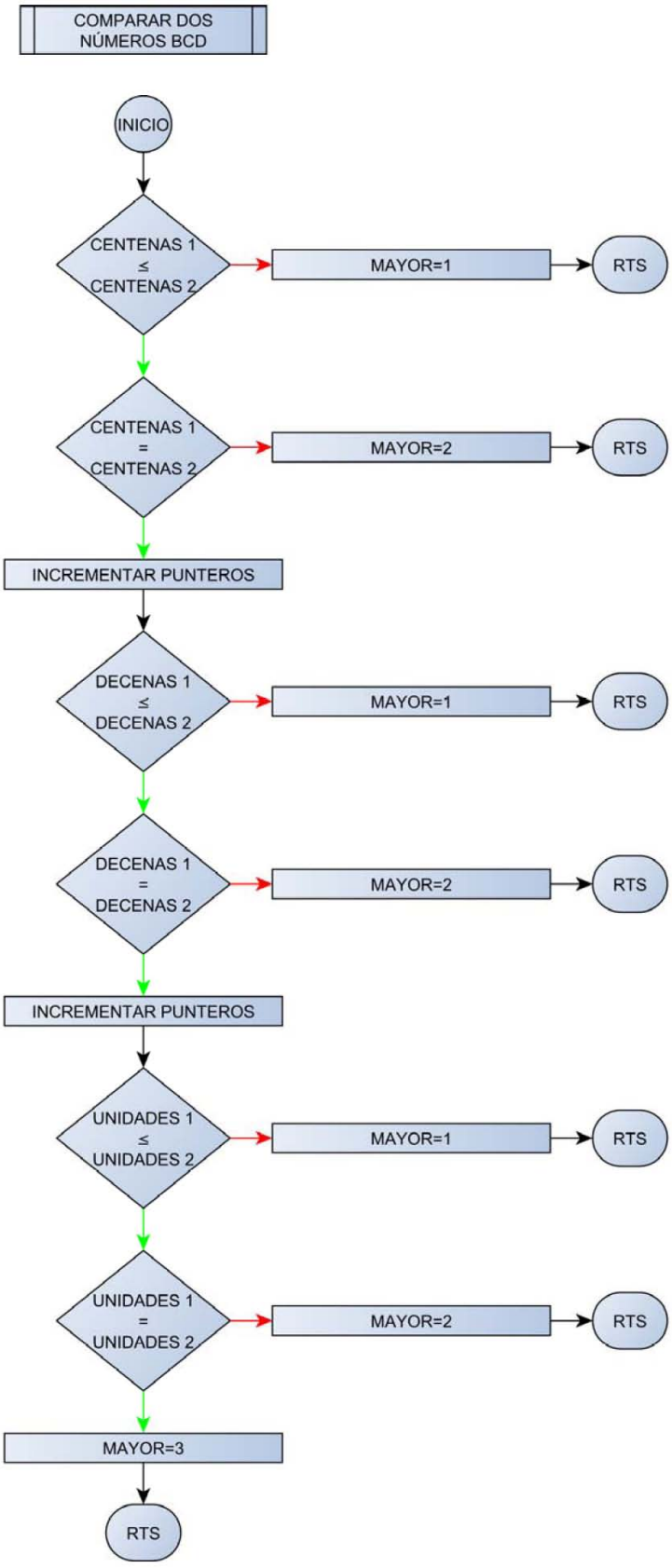


Ilustración 67: COMPARAR DOS NUMEROS BCD

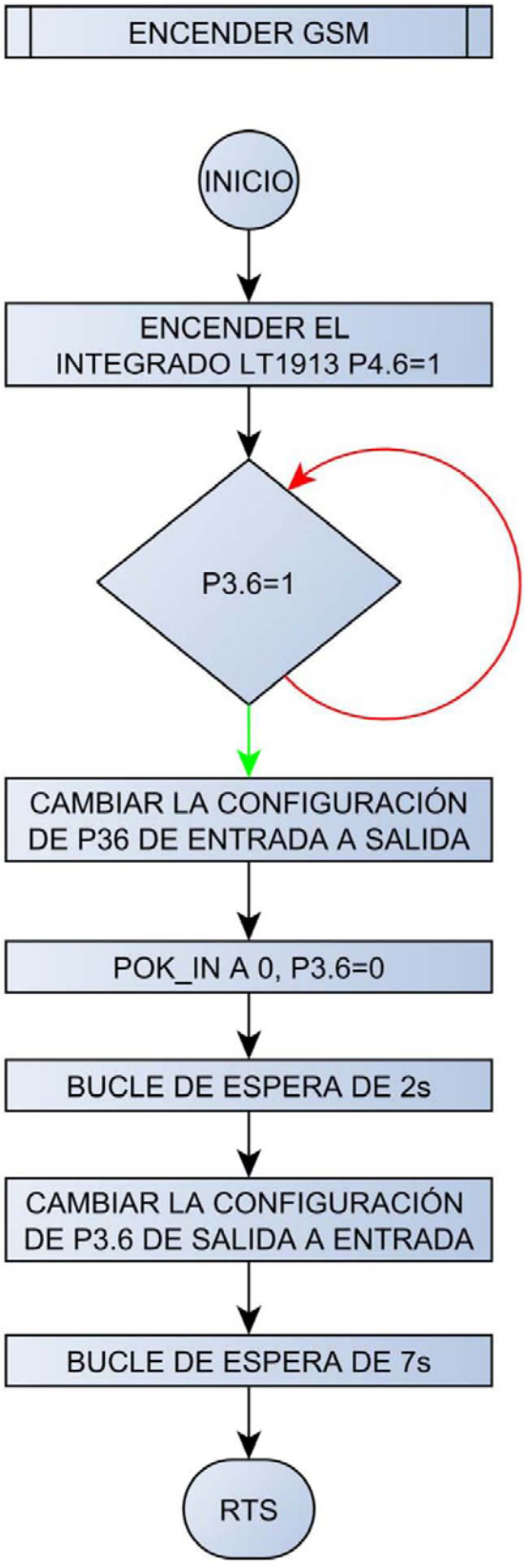


Ilustración 68: ENCENDER GSM

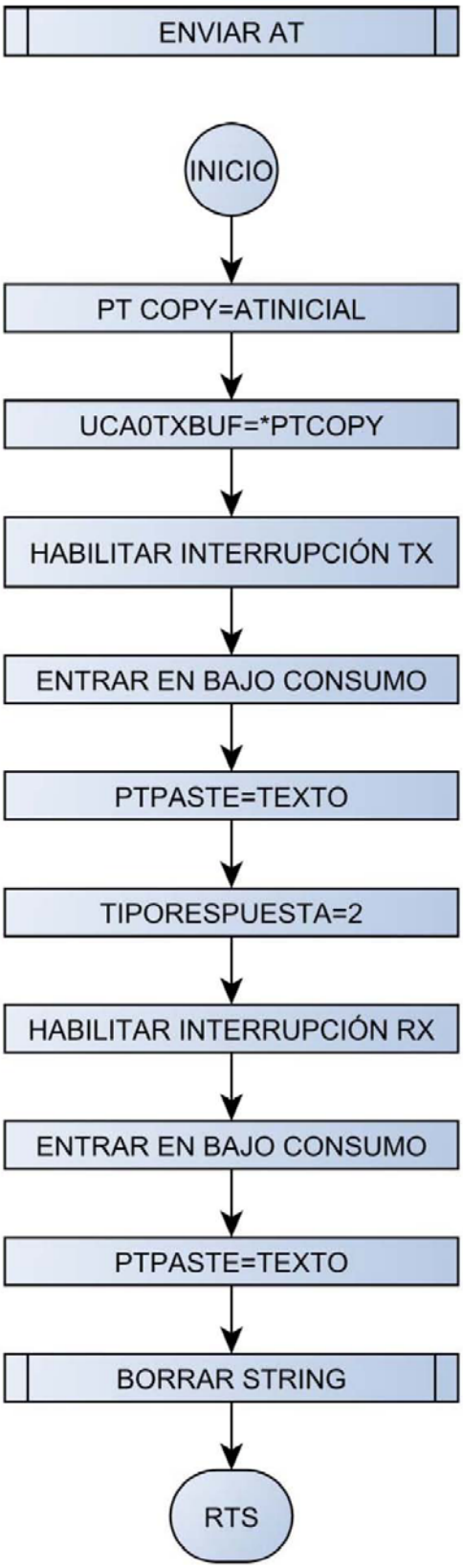


Ilustración 69: ENVIAR AT

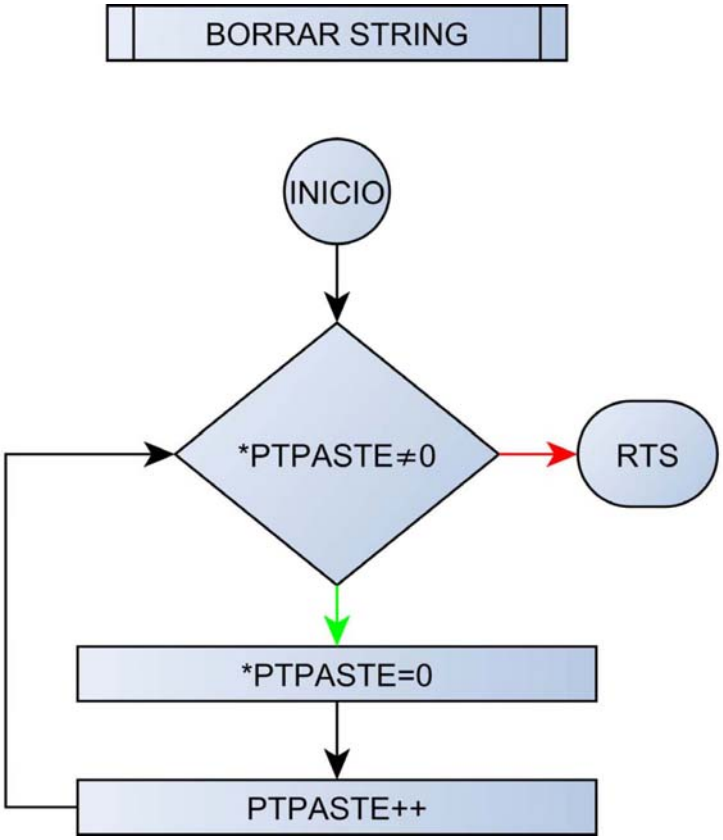


Ilustración 70: BORRAR STRING

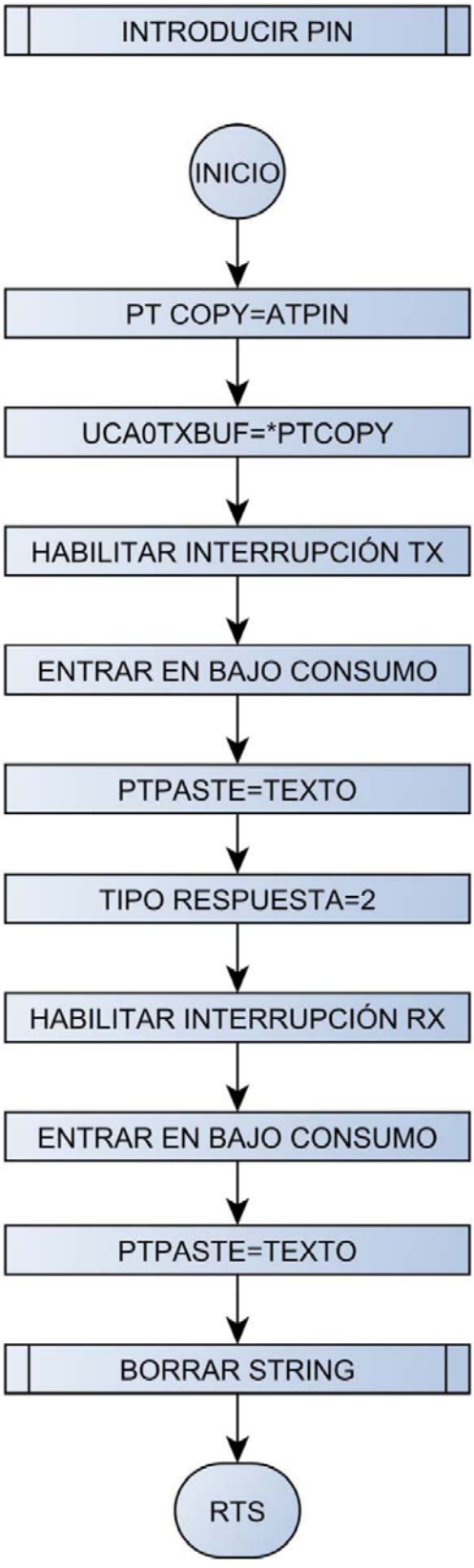


Ilustración 71: INTRODUCIR PIN

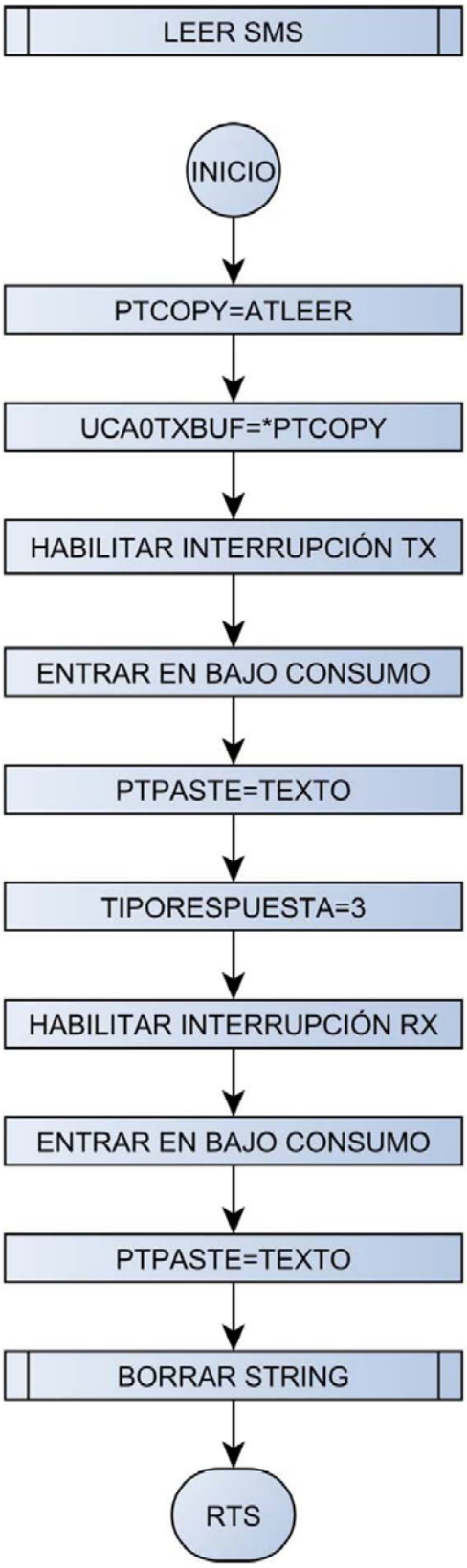


Ilustración 72: LEER SMS

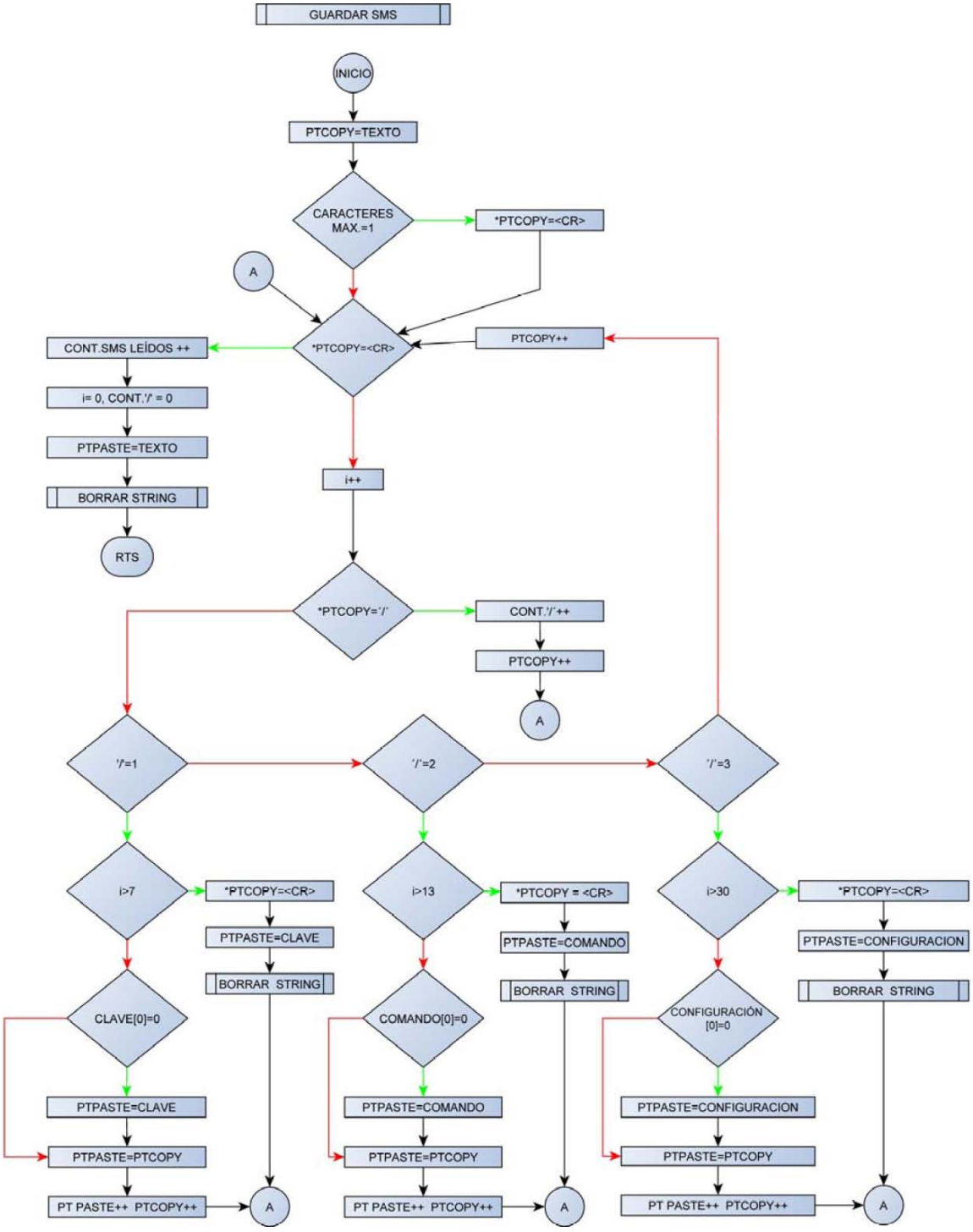


Ilustración 73: GUARDAR SMS

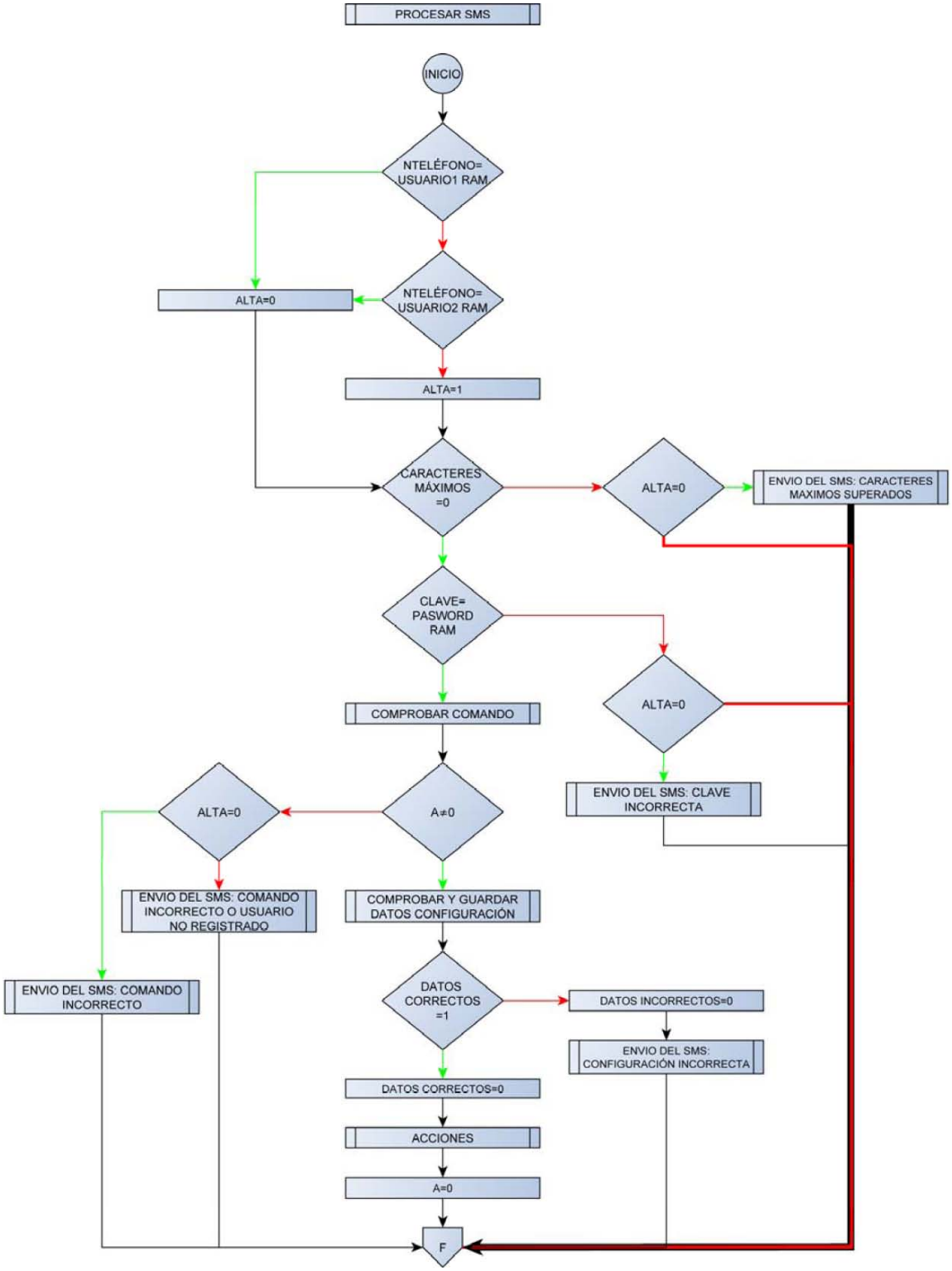


Ilustración 74: PROCESAR SMS

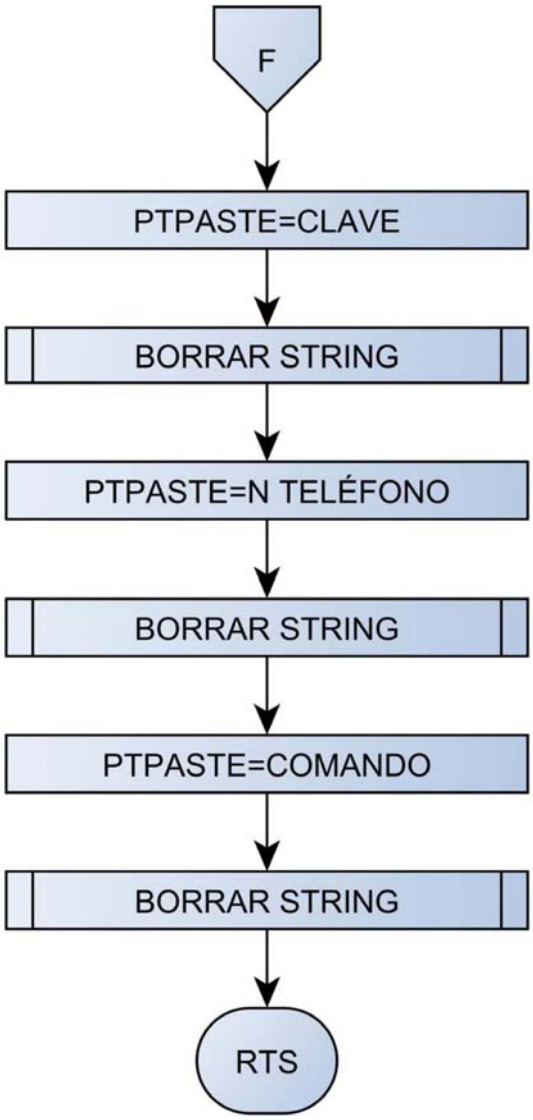


Ilustración 75: PROCESAR SMS

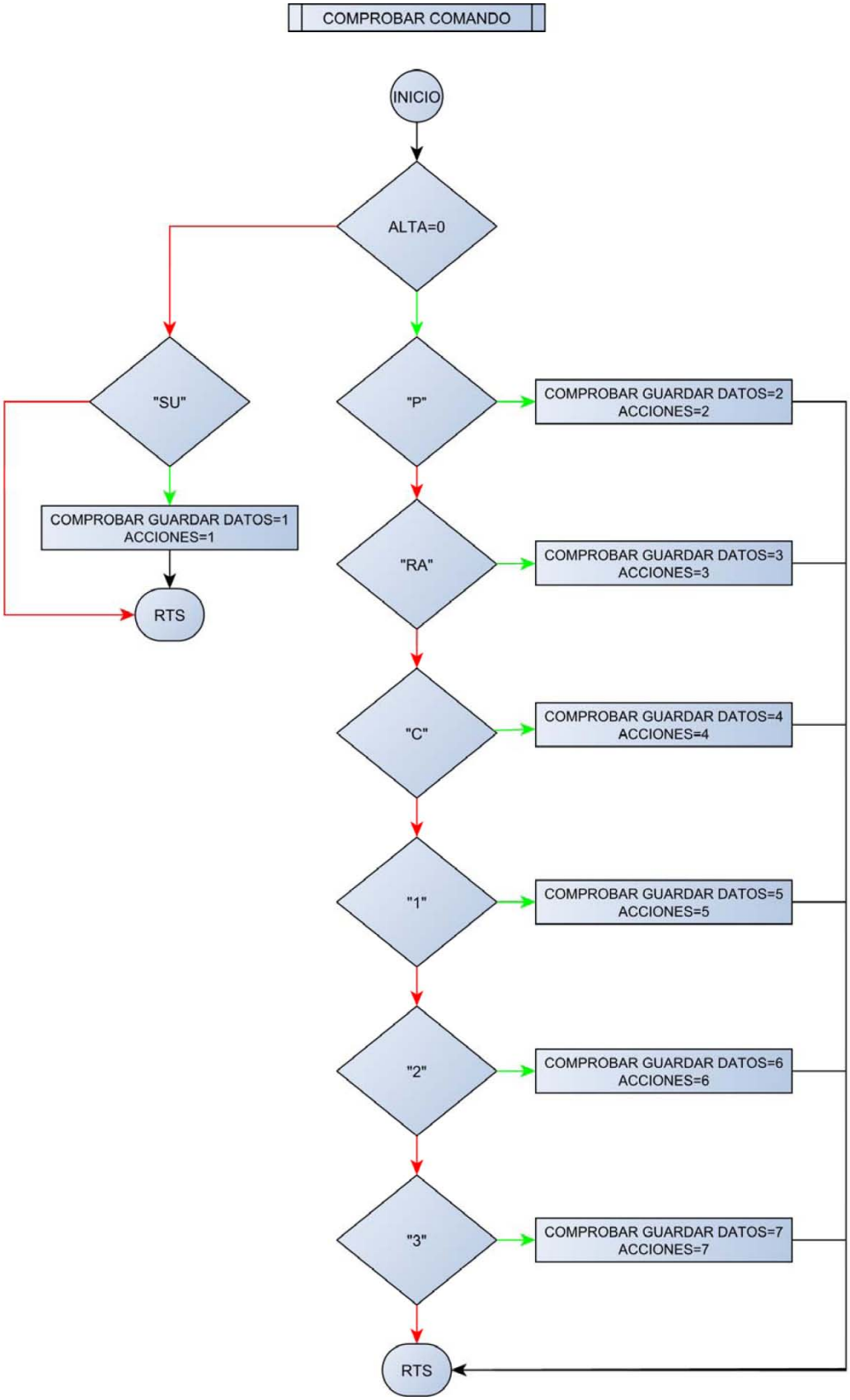


Ilustración 76: COMPROBAR COMANDO

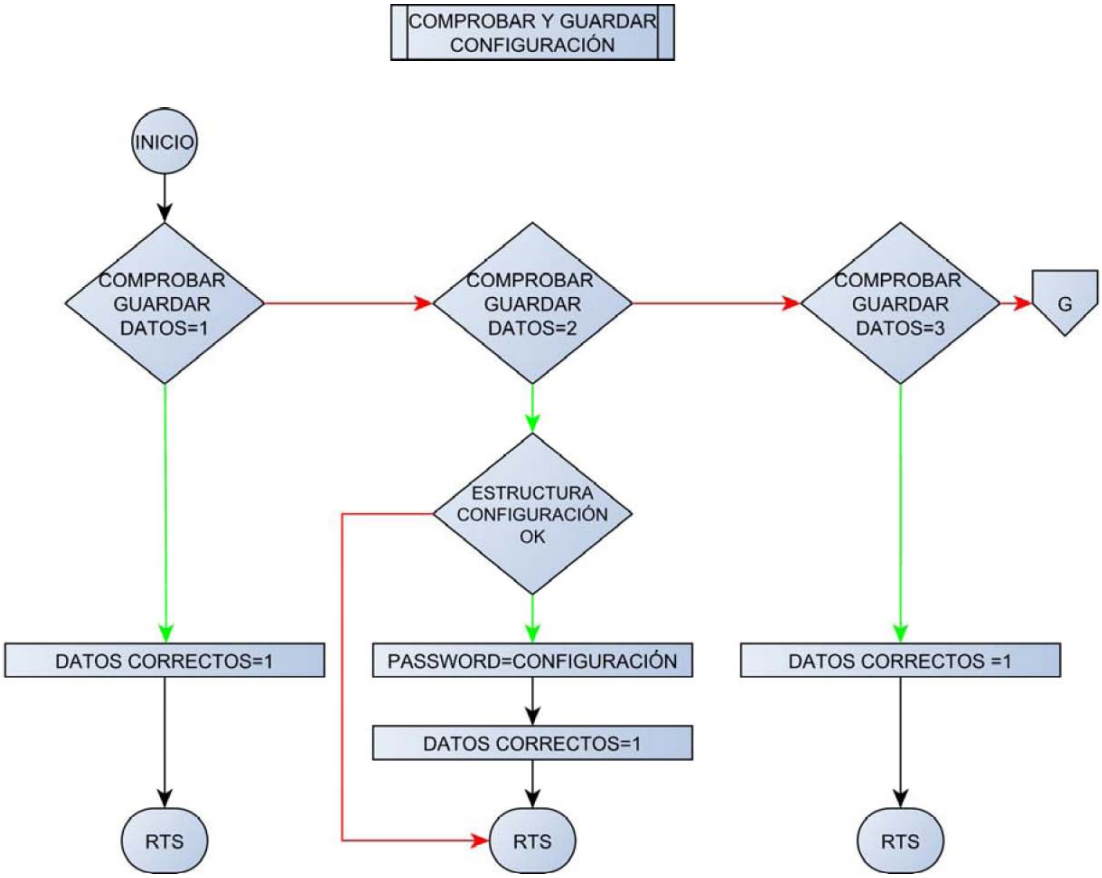


Ilustración 77: COMPROBAR Y GUARDAR CONFIGURACIÓN

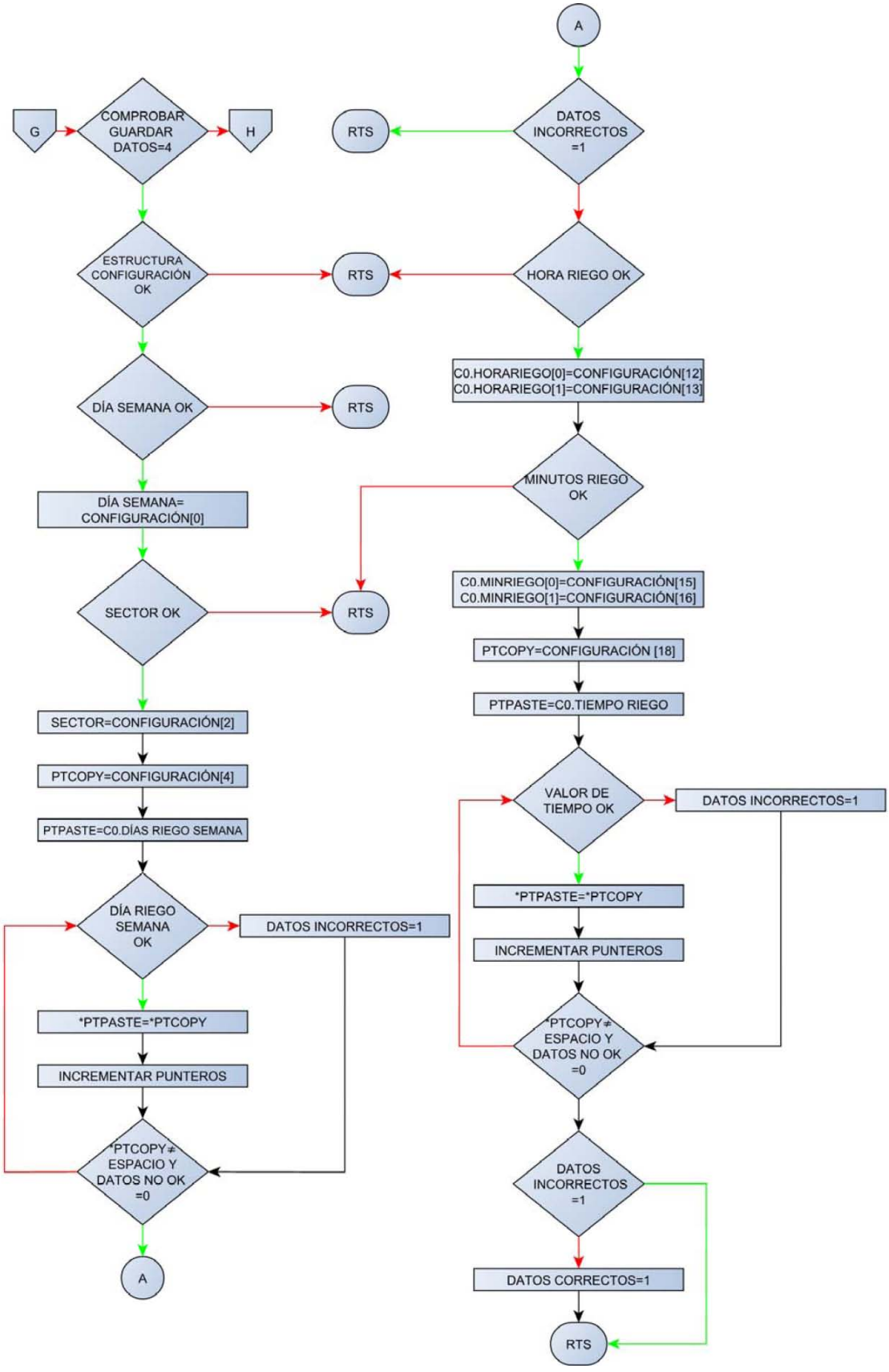


Ilustración 78: COMPROBAR Y GUARDAR CONFIGURACIÓN

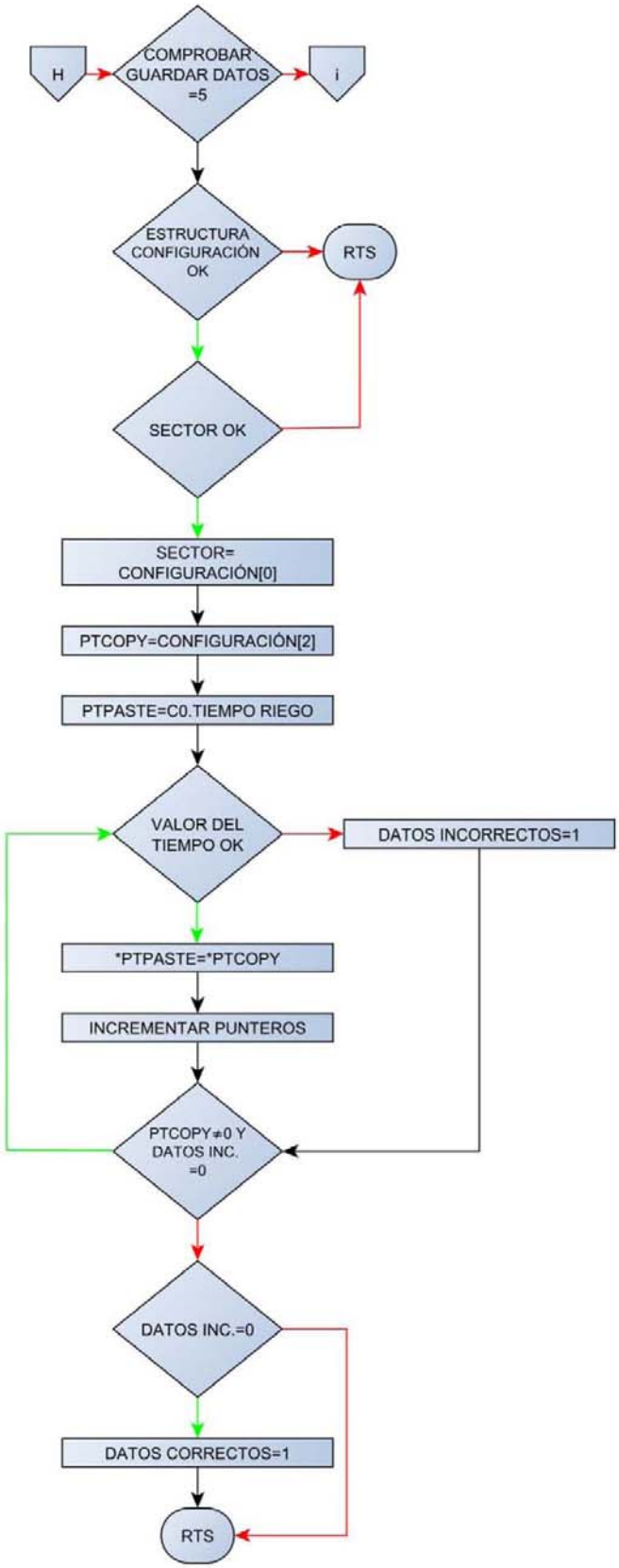


Ilustración 79: COMPROBAR Y GUARDAR CONFIGURACIÓN

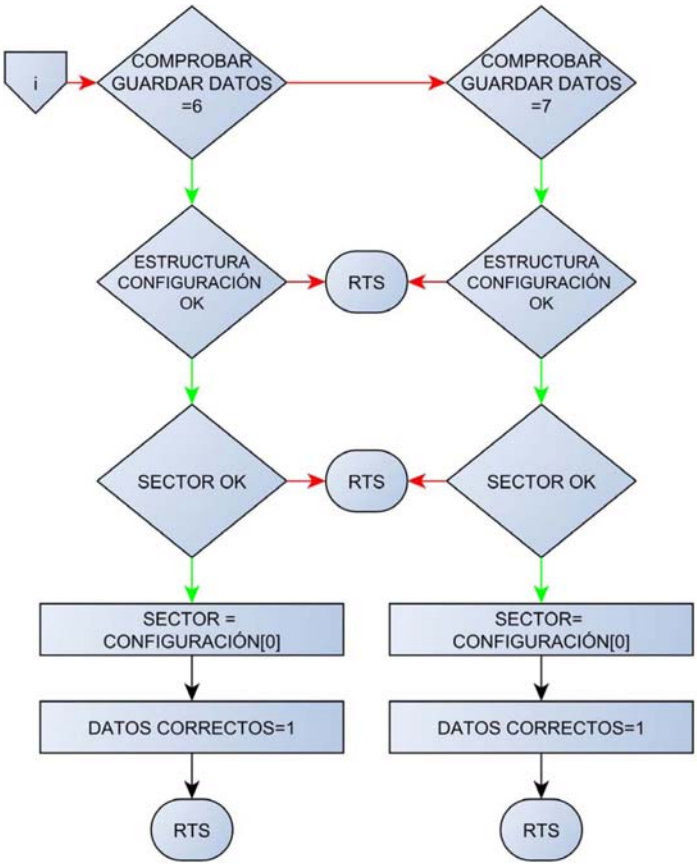


Ilustración 80: COMPROBAR Y GUARDAR CONFIGURACIÓN

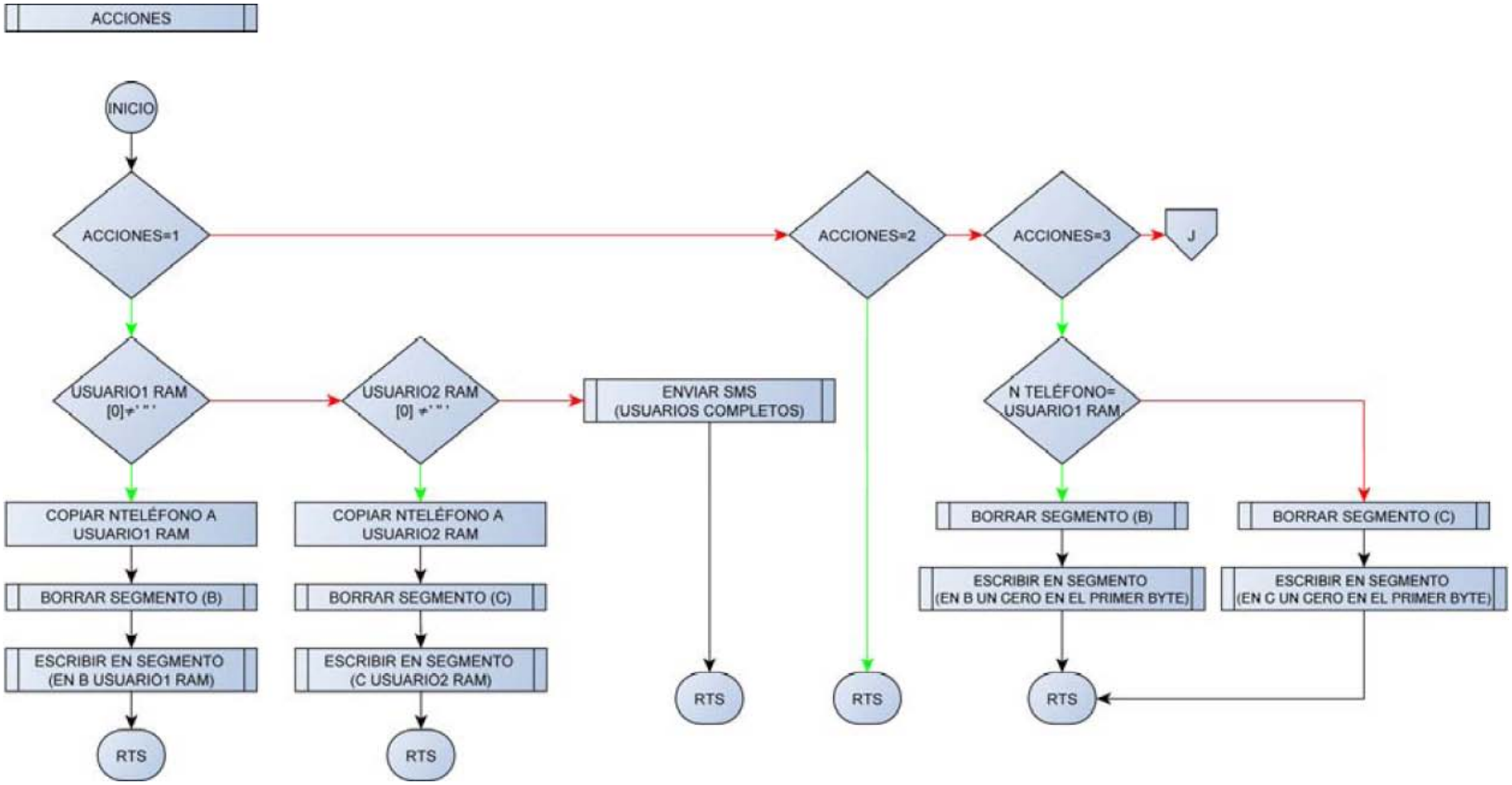


Ilustración 81: ACCIONES

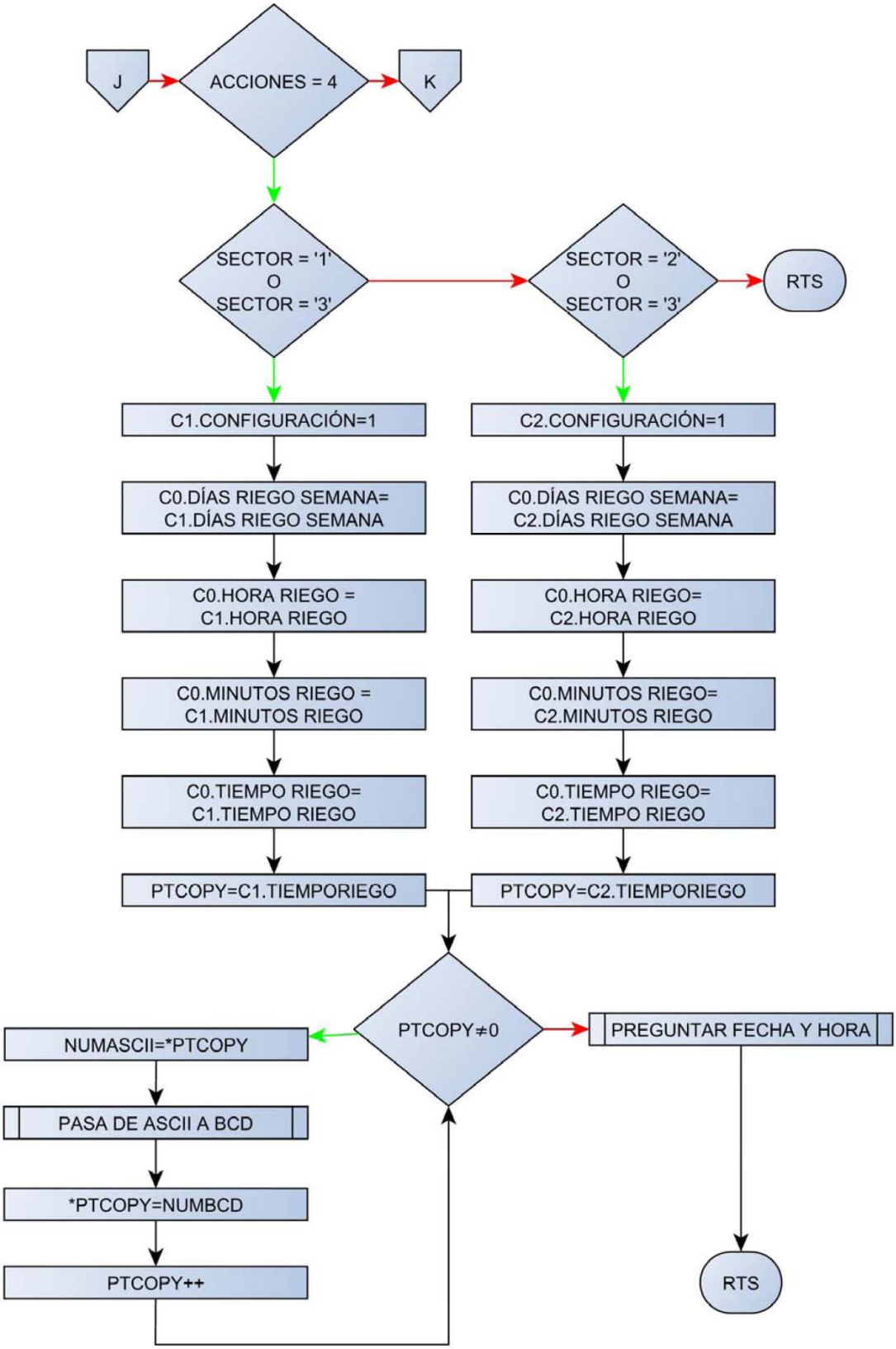


Ilustración 82: ACCIONES

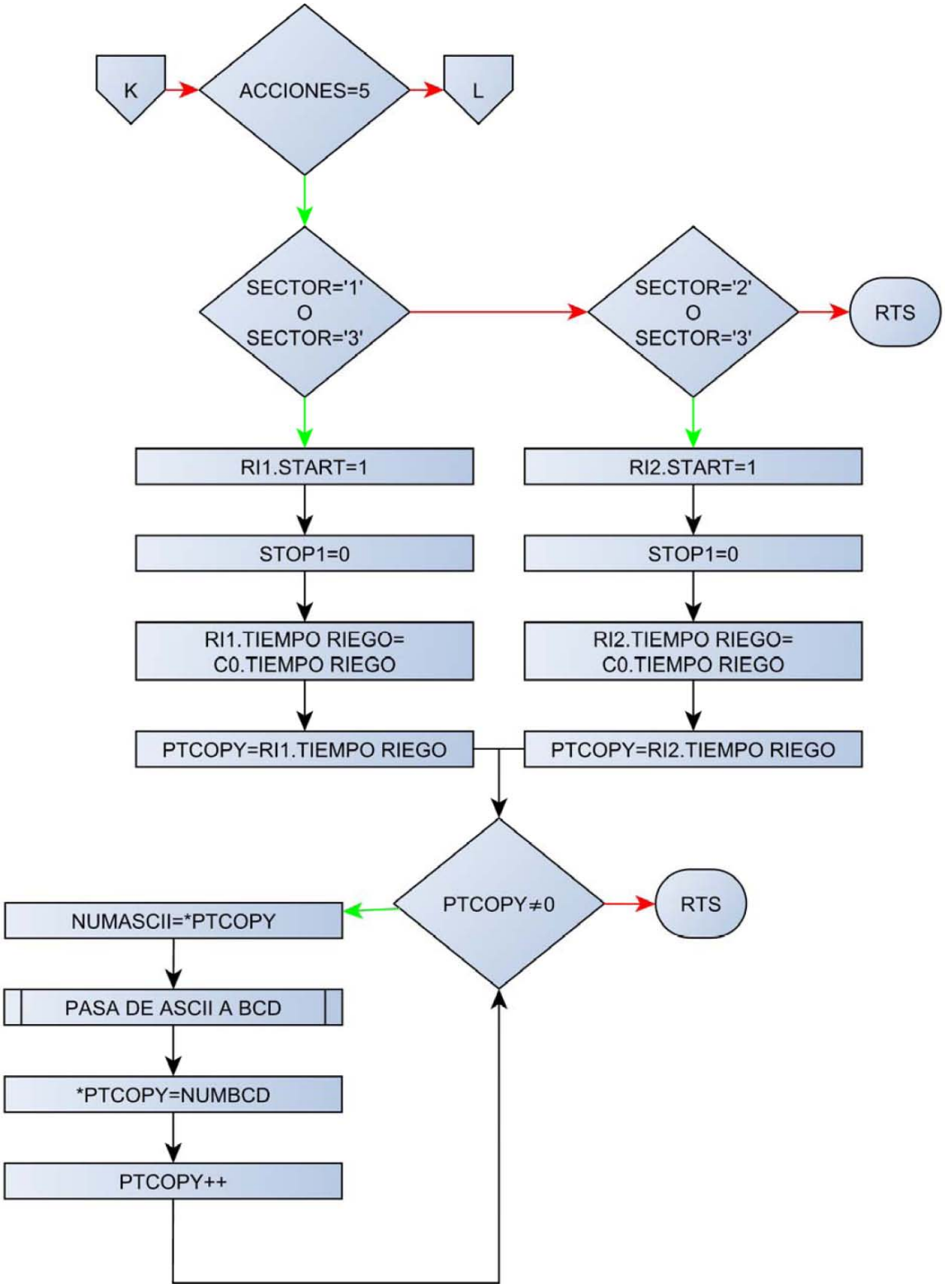


Ilustración 83: ACCIONES

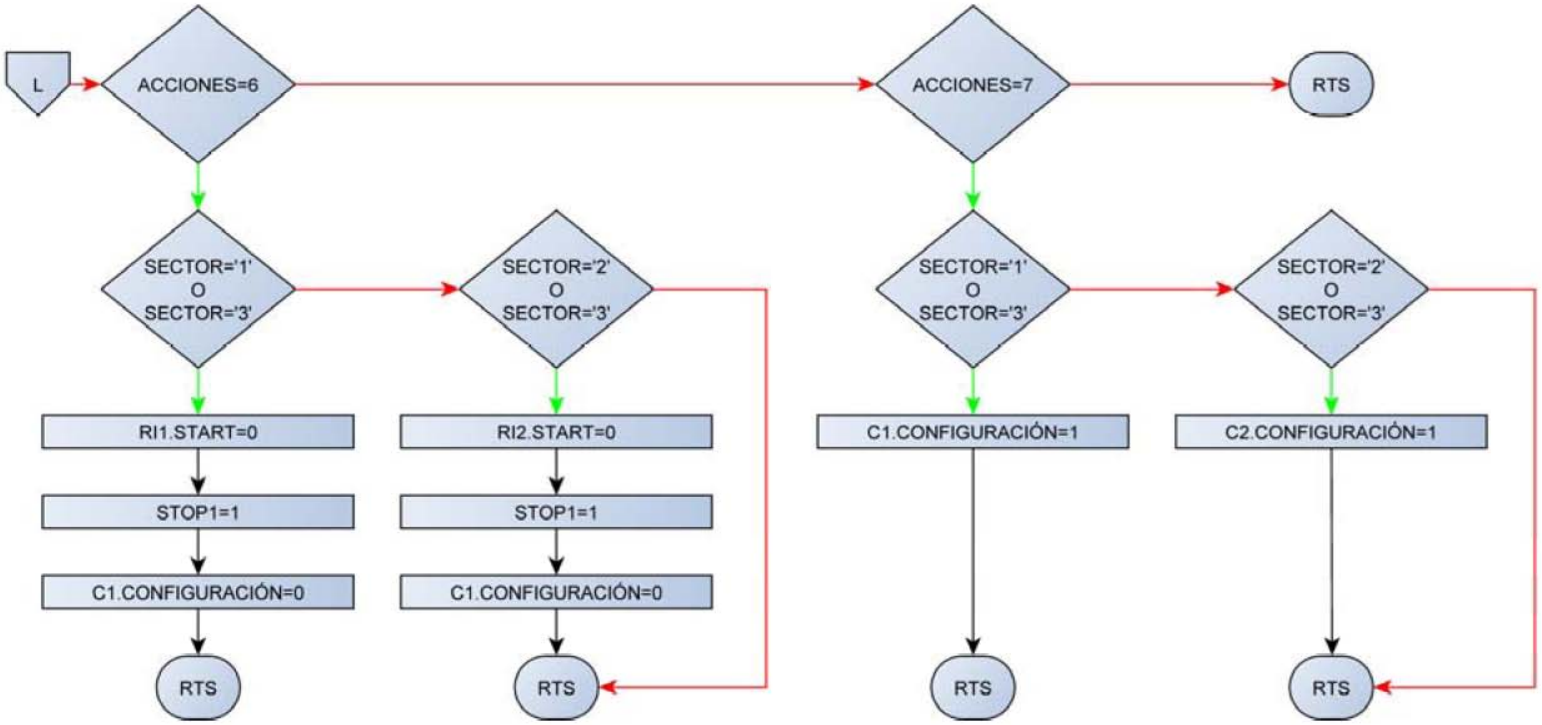


Ilustración 84: ACCIONES

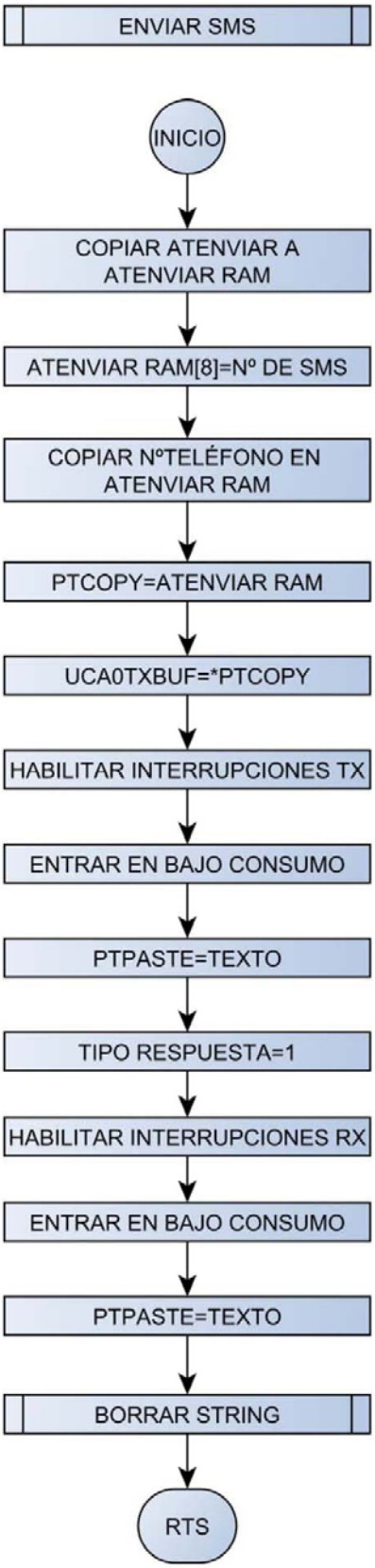


Ilustración 85: ENVIAR SMS

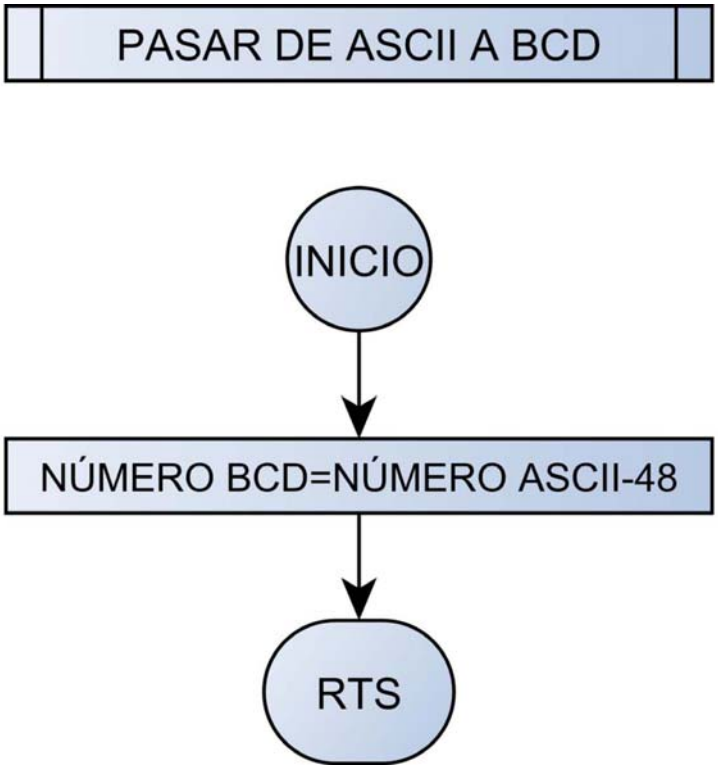


Ilustración 86: PASAR DE ASCII A BCD

Ilustración 87: PREGUNTAR FECHA Y HORA

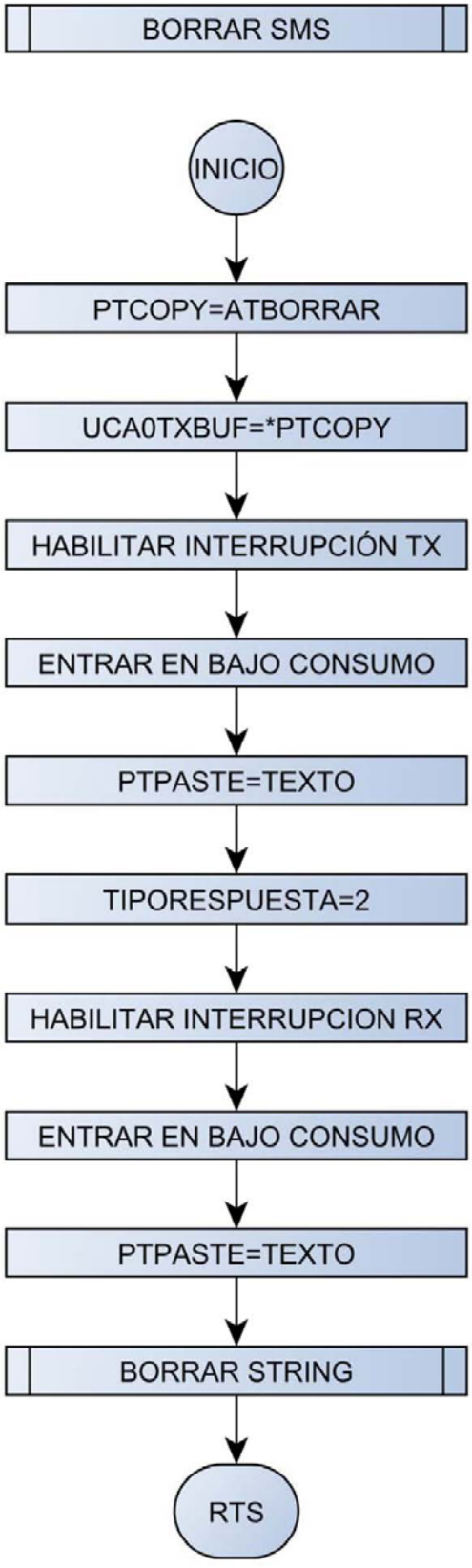


Ilustración 88: BORRAR SMS

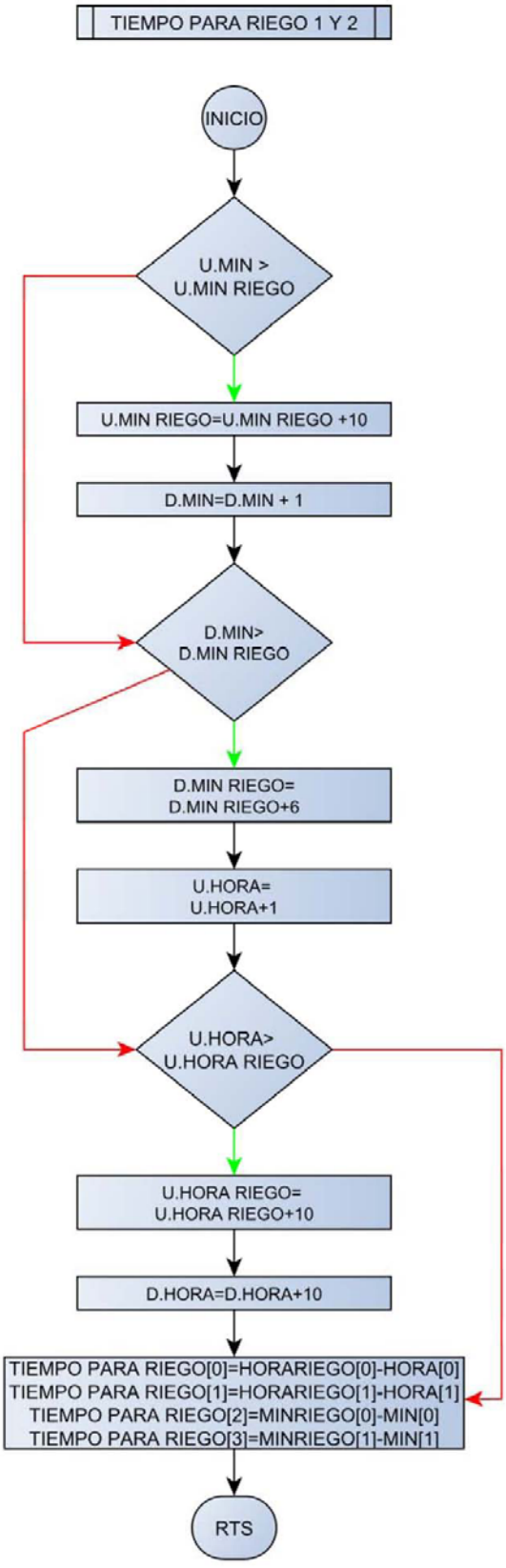


Ilustración 89: TIEMPO PARA RIEGO 1 Y 2

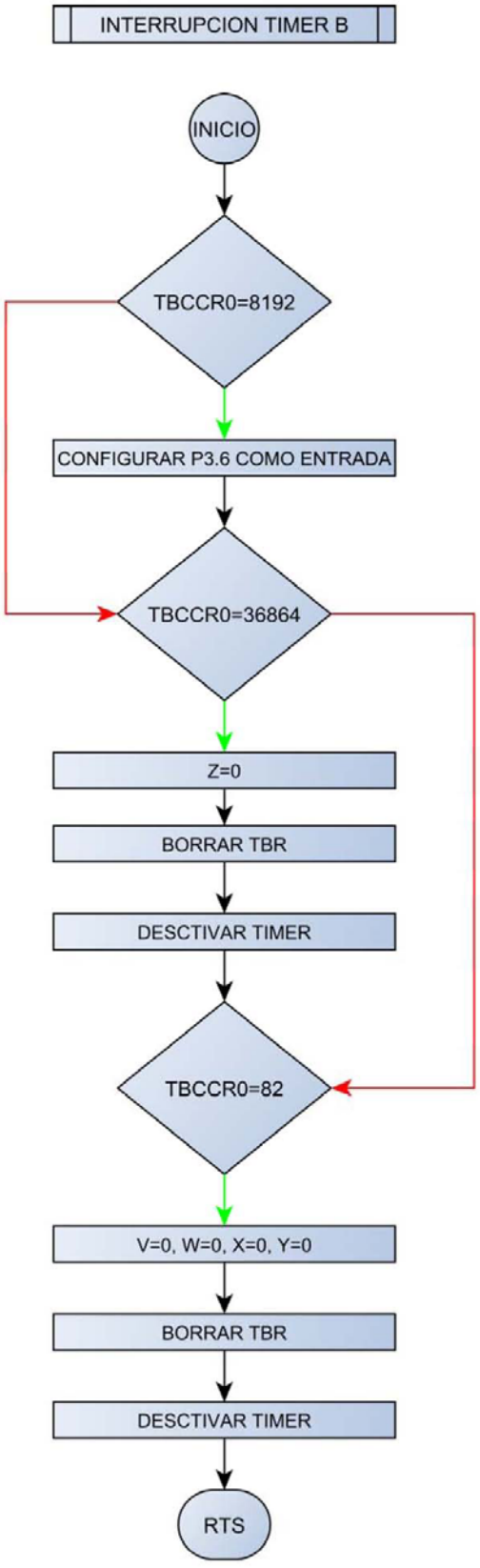


Ilustración 90: INTERRUPCION TIMER B

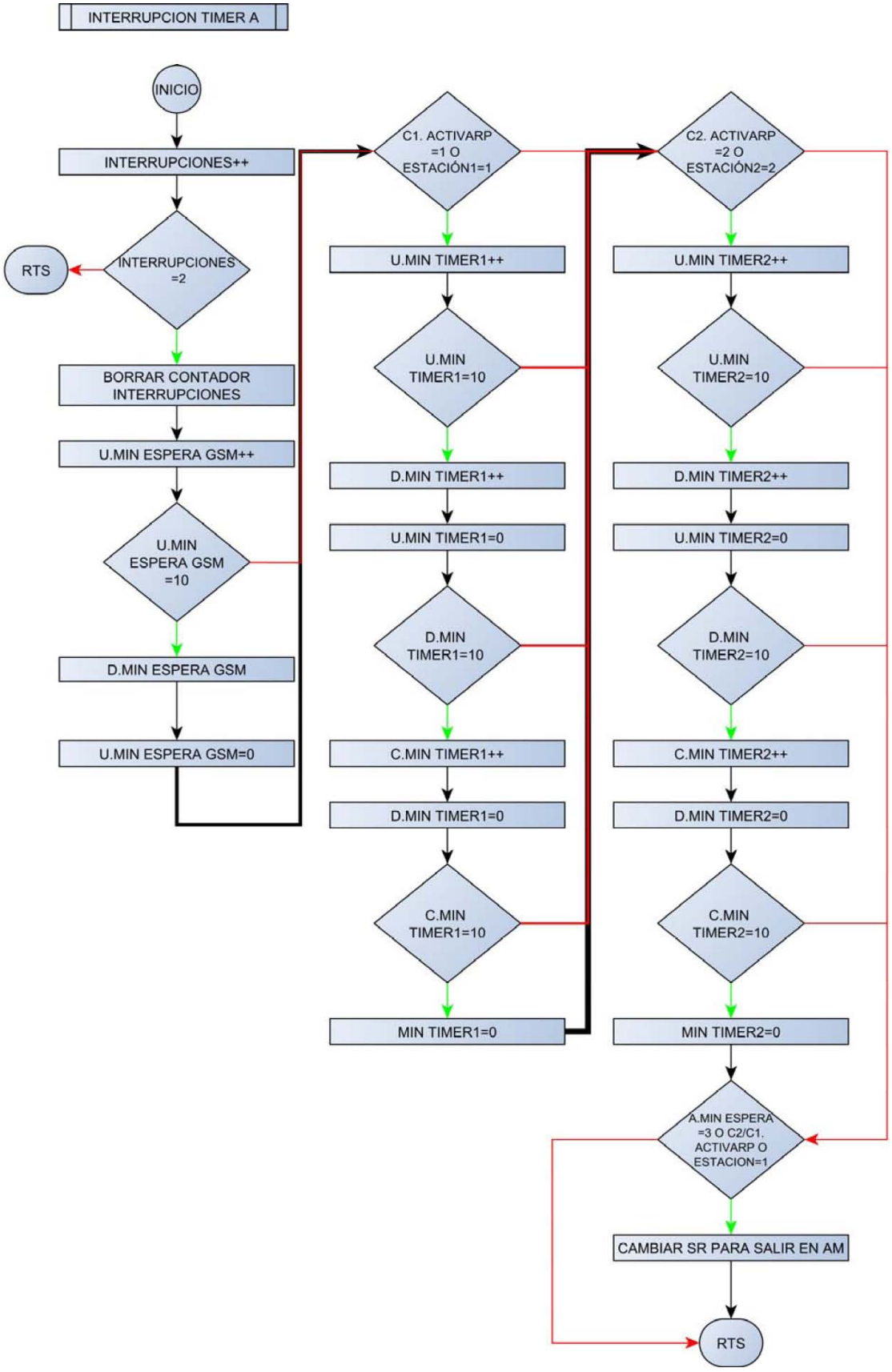


Ilustración 91: INTERRUPCIÓN TIMER A

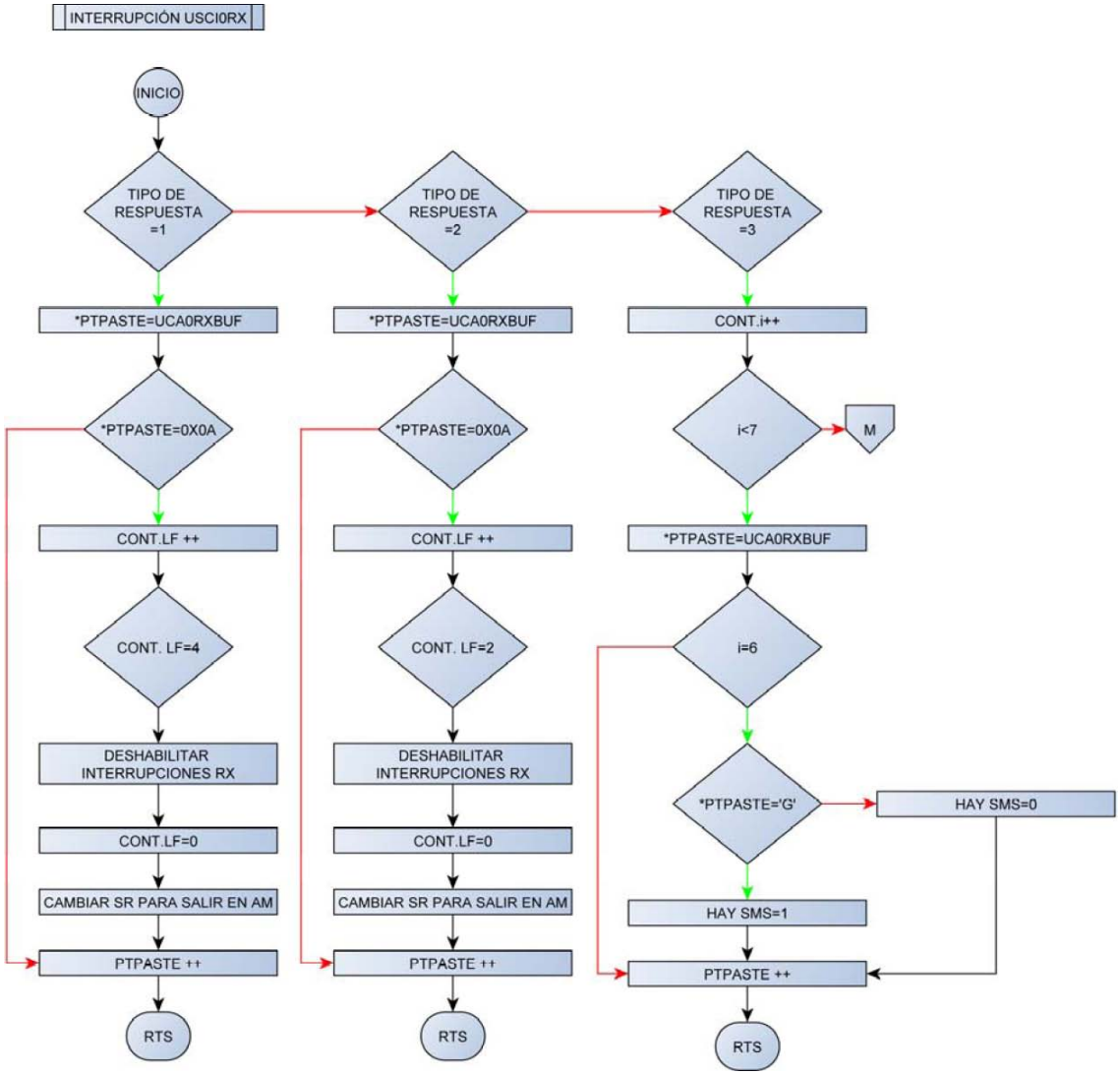


Ilustración 92: INTERRUPCIÓN USCI0RX

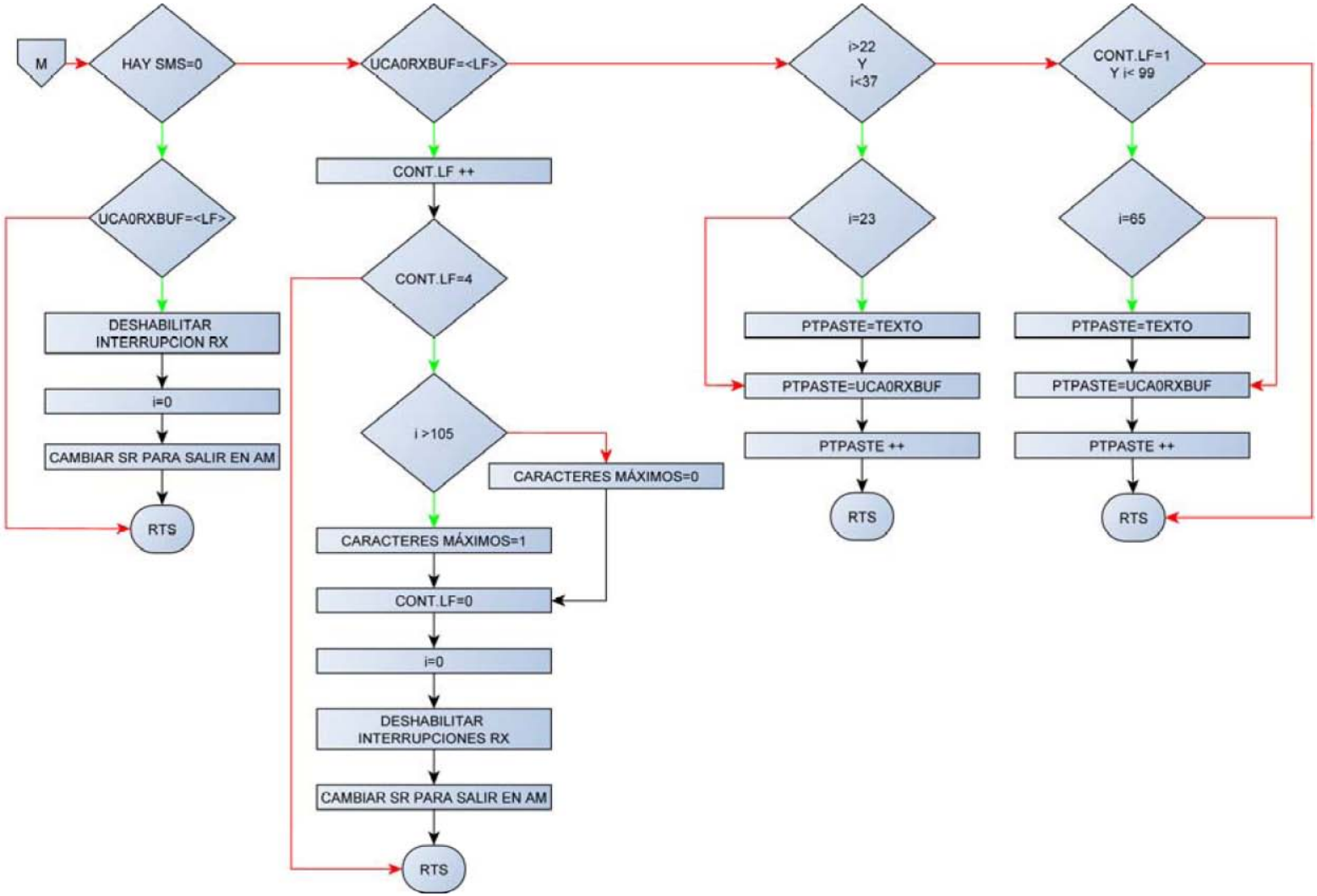


Ilustración 93: INTERRUPCION USCI0RX

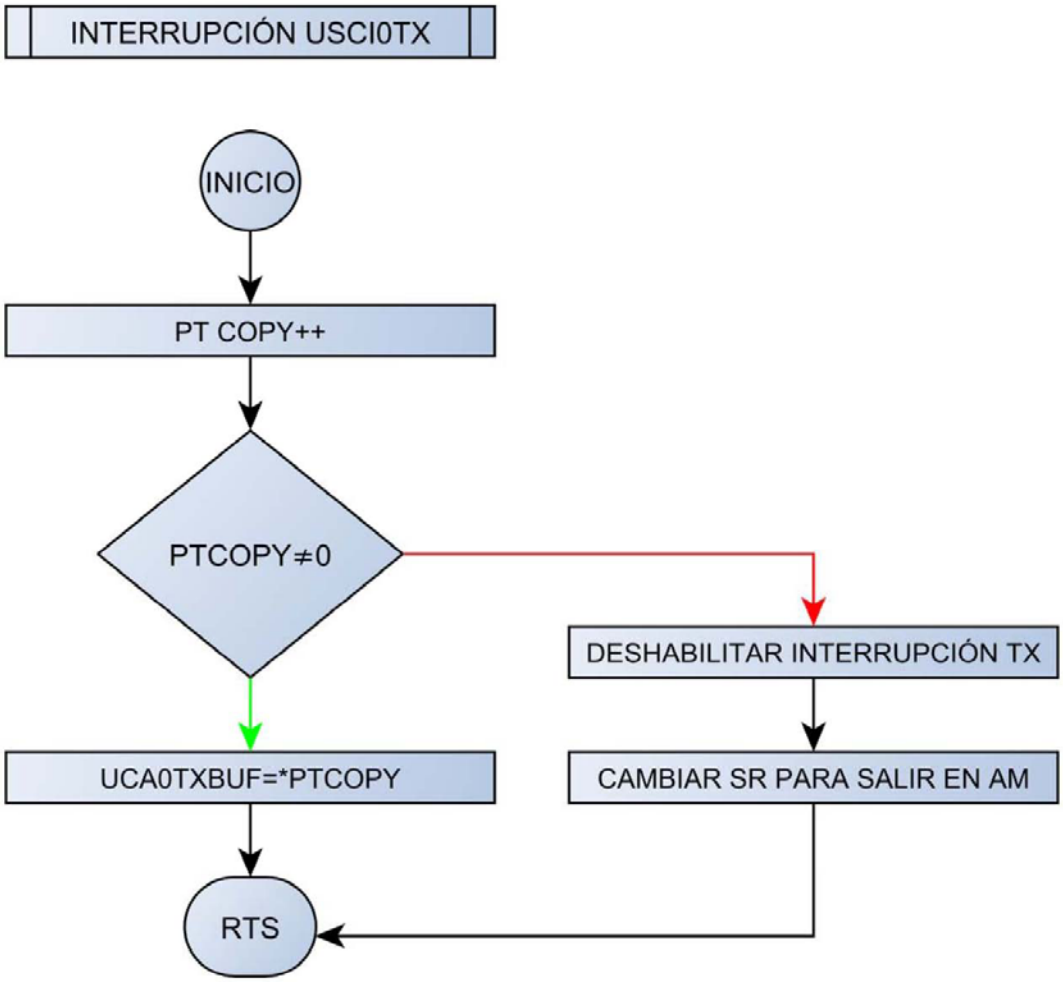


Ilustración 94: USCI0TX

ANEXOS Nº 6: CÓDIGO DEL PROGRAMA

```
#include "msp430f2252.h"
#include "intpin.h"
#include "leerSMS.h"
#include "borrarSMS.h"
#include "guardarSMS.h"
#include "procesarSMS.h"
#include "tpariego1.h"
#include "tpariego2.h"
#include "string.h"
#include "preguntarfechayhora.h"
#include "encenderGSM.h"
#include "comparardosnumBCD.h"
#include "write_segment.h"
#include "read_segment.h"
#include "erase_segment.h"
#include "enviarAT.h"
#include "iniciarriego1.h"
#include "iniciarriego2.h"
#include "finalizarriego1.h"
#include "finalizarriego2.h"

#define info_seg_B 0x1080
#define info_seg_C 0x1040
#define info_seg_D 0x1000

#pragma DATA_SECTION(password,".infoD")
const char password[7] = {"RIXSMS"};
#pragma DATA_SECTION(usuario1, ".infoB")
const char usuario1[15]={""};
#pragma DATA_SECTION(usuario2, ".infoC")
const char usuario2[15]={""};

char i=0;
char contlf=0;
char texto[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char clave[7]={0,0,0,0,0,0,0};
char comando[4]={0,0,0,0};
char configuracion[23]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char *ptpaste;
char *ptcopy;
char *Flash_ptr;
char tiporespuesta=0;
char contsmsleidos=0;
char haysms=0;
char caracteresmax=0;
char ntelefono[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char usuario1RAM[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char usuario2RAM[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char a=0;
```

```
char sector=0;
char alta=0;
char cgdatos=0;
char datoscorrectos=0;
char datosincorrectos=0;
char diames[3]={0,0,0};
char diasemana=0;
char interrupciones=0;
char mayor=0;
char estacion1=0;
char estacion2=0;
char stop1=0;
char stop2=0;
char passwordRAM[7]={0,0,0,0,0,0,0};
struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
}C0={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
        char tiempopararriego[5];
        char activar;
        char activarp;
        char configuracion;
        char heregadohoy;
}C1={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
        char tiempopararriego[5];
        char activar;
        char activarp;
        char configuracion;
        char heregadohoy;
}C2={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
struct {
        char diamesactual[3];
        char hora[3];
        char min[3];
}reloj={0,0,0,0,0,0,0,0,0};
struct {
        char temporizar;
        char tiemporiego[4];
        char start;
}RI1={0,0,0,0,0,0};
```

```
struct {
        char temporizar;
        char tiemporiego[4];
        char start;
}RI2={0,0,0,0,0,0};
struct {
        char min[3];
        char minriego[3];
        char horariego[3];
        char hora[3];
}OP={0,0,0,0,0,0,0,0,0,0,0,0};
struct {
        char cmin1;
        char dmin1;
        char umin1;
}timer1={0,0,0};
struct {
        char cmin2;
        char dmin2;
        char umin2;
}timer2={0,0,0};
struct{
        char dmine;
        char umine;
}timere={0,0};
char z=0;
char y=0;
char x=0;
char w=0;
char v=0;
void main(void) {
        WDTCTL = WDTPW + WDTHOLD;                   // Stop WDT
/****************************** CONFIGURAR PUERTOS *********************************/
        //P1SEL = 0x00    // configuración como I/O por defecto
        P1DIR = 0xFF;     // p1.x salidas, TCK/TMS/TDI/TDO (función controlada por JTAG)
        P1OUT = 0;
        //P2SEL |= 0xC0; // así por defecto, todos los I/0 excepto P2.6 y P2.7 ACLK
        P2DIR = 0xFF;
        P2OUT = 0x00;
        P3SEL |= 0x30;    // P3.4,5 = USCI_A0 TXD/RXD; P3.6 I/O
        P3DIR = 0xBF;     // todos P3.x salidas excepto p3.6 entrada
        P3OUT = 0;        // todos P3.x reset
        //P4SEL |= 0x00  // configuración como puerto por defecto
        P4DIR = 0xFF;     // todos P4.x otputs p4.6 on/off buck
        P4OUT = 0;        // todos p4.x reset
/************************** FLASH MEMORY CONTROLLER CONFIGURATION ****************/
        FCTL2 = FWKEY + FSSEL_1 + FN1;   // MCLK como señal de reloj, y dividida para 3
/***************************** CONFIGURAR TIMERB ***********************************/
        BCSCTL3 |= XCAP_3;                              // XIN/XOUT Cap : 12.5 pF
        TBCTL = TBSSEL_1 + ID_3;                        // aclk, divisor/8, stopmode, TBIFG=0
/***************************** CONFIGURAR TIMERA ***********************************/
        TACTL = TASSEL_1 + ID_3 + MC_3 + TAIE;  // ACLK, divisor/8, up/down mode, TAIFG=1
        TACCR0=61140;                           // valor hasta el que cuenta TAR
```

```
/****************************** CONFIGURAR USCI ************************************/
        UCA0CTL1 |= UCSSEL_1;              // CLK = ACLK
        UCA0BR0 = 0x03;                    // 32kHz/9600 = 3.41
        UCA0BR1 = 0x00;
        UCA0MCTL = UCBRS_3;                // modulation UCBRSx=3
/********************************* INICIALIZAR REGISTROS ***************************/
        contsmsleidos='1';
/*******************************PROGRAMA PRINCIPAL ********************************/
        read_segment(info_seg_D, passwordRAM);          //copio de la Flash los posibles
        read_segment(info_seg_B, &usuario1RAM[0]);      //valores anteriores
        read_segment(info_seg_C, &usuario2RAM[0]);
        for (;;){
                while (timere.dmine!=3){
                        if (C1.activarp==1){
                                if ((timer1.dmin1>C1.tiempopararriego[2])||
                                                ((timer1.dmin1==C1.tiempopararriego[2])&
                                                        (timer1.umin1>=C1.tiempopararriego[3]))){
                                        C1.activarp=0;
                                        timer1.cmin1=0;
                                        timer1.dmin1=0;
                                        timer1.umin1=0;
                                        estacion1=1;
                                        iniciarriego1();
                                }
                        }
                        if (C2.activarp==1){
                                if ((timer2.dmin2>C2.tiempopararriego[2])||
                                                ((timer2.dmin2==C2.tiempopararriego[2])&
                                                        (timer2.umin2>=C2.tiempopararriego[3]))){
                                        C2.activarp=0;
                                        timer2.cmin2=0;
                                        timer2.dmin2=0;
                                        timer2.umin2=0;
                                        estacion2=1;
                                        iniciarriego2();
                                }
                        }
                        if (estacion1==1){
                                if (RI1.temporizar==1){
                                        comparardosnumBCD(&timer1.cmin1,&RI1.tiemporiego[0]);
                                }


                                else {
                                        if (C1.configuracion==1){
                                                comparardosnumBCD(&timer1.cmin1,
                                                &C1.tiemporiego[0]);
                                        }
                                }
                                if ((mayor==0x01)||(mayor==0x03)){
                                        RI1.temporizar=0;
                                        estacion1=0;
                                        mayor=0;
```

```
                        timer1.cmin1=0;
                        timer1.dmin1=0;
                        timer1.umin1=0;
                        finalizarriego1();
                }
        }
        if (estacion2==1){
                if (RI2.temporizar==1){
                        comparardosnumBCD(&timer2.cmin2, &RI2.tiemporiego[0]);
                }
                else {
                        if (C2.configuracion==1){
                                comparardosnumBCD(&timer2.cmin2,
                                &C2.tiemporiego[0]);
                        }
                }
                if ((mayor==0x01)||(mayor==0x03)){
                        RI2.temporizar=0;
                        estacion2=0;
                        mayor=0;
                        timer2.cmin2=0;
                        timer2.dmin2=0;
                        timer2.umin2=0;
                        finalizarriego2();
                }
        }
        __bis_SR_register(LPM3_bits + GIE); // Entrar en bajo consumo, int habilitadas
}
timere.dmine=0;
encenderGSM();                  // No es necesaria con la placa de LIBELIUM
                                // pero viene bien para poner en marcha el cristal
enviarAT();
intpin();
leerSMS(contsmsleidos);
while (haysms==1){
        guardarSMS();
        procesarSMS();
        contsmsleidos++;
        leerSMS(contsmsleidos);
}
if (contsmsleidos>'1'){
        borrarSMS();
        contsmsleidos='1';
}
preguntarfechayhora();
//desconexionbuckGSM();         // Esta función sería necesaria en el caso de no
                                // utilizar la placa de LIBELIUM.
UCA0CTL1 |= UCSWRST;            // Deshabilitar USCI
if (diames[1]!=0){
        if ((reloj.diamesactual[1]!=diames[1])||(reloj.diamesactual[0]!=diames[0])){
                diames[0]=reloj.diamesactual[0];
                diames[1]=reloj.diamesactual[1];
                C1.heregadohoy=0;
```

```
                C2.heregadohoy=0;
                if (diasemana=='7'){
                        diasemana='1';
                }
                else {
                        diasemana++;
                }
        }
        if ((C1.configuracion==1)&(C1.heregadohoy==0)&
                (RI1.start==0)&(estacion1==0)){
                if ((strchr(C1.diasriegosemana,diasemana)!=0)){
                        tparariego1(&reloj.min[1], &C1.minriego[1],
                        &C1.horariego[1], &reloj.hora[1]);
                        if ((C1.tiempopararriego[0]==0)&(C1.tiempopararriego[1]==0)){
                                if (C1.tiempopararriego[2]<3){
                                        if ((C1.tiempopararriego[2]==0)&
                                                (C1.tiempopararriego[3]==0)){
                                                C1.activar=1;
                                        }
                                        else {
                                                C1.activarp=1;
                                        }
                                }
                                C1.heregadohoy=1;
                        }
                }
        }
        if ((C2.configuracion==1)& (C2.heregadohoy==0)& (RI2.start==0)&
                (estacion2==0)){
                if ((strchr(C2.diasriegosemana,diasemana)!=0)){
                        tparariego2(&reloj.min[1], &C2.minriego[1],
                        &C2.horariego[1], &reloj.hora[1]);
                        if ((C2.tiempopararriego[0]==0)&(C2.tiempopararriego[1]==0)){
                                if (C2.tiempopararriego[2]<3){
                                        if ((C2.tiempopararriego[2]==0)&
                                        (C2.tiempopararriego[3]==0)){
                                                C2.activar=1;
                                        }
                                        else {
                                                C2.activarp=1;
                                        }
                                }
                                C2.heregadohoy=1;
                        }
                }
        }
}
if (stop1==1){
        stop1=0;
        if (estacion1==1){
                estacion1=0;
                timer1.umin1=0;
                timer1.dmin1=0;
```

```
                        timer1.cmin1=0;
                        RI1.temporizar=0;
                        finalizarriego1();
                }
        }
        else if (RI1.start==1){
                RI1.start=0;
                if (estacion1==1){
                        timer1.umin1=0;
                        timer1.dmin1=0;
                        timer1.cmin1=0;
                }
                else{
                        estacion1=1;
                        iniciarriego1();
                }
                RI1.temporizar=1;
        }
        else if (C1.activar==1){
                C1.activar=0;
                estacion1=1;
                iniciarriego1();
        }
        if (stop2==1){
                stop2=0;
                if (estacion2==1){
                        estacion2=0;
                        timer2.umin2=0;
                        timer2.dmin2=0;
                        timer2.cmin2=0;
                        RI2.temporizar=0;
                        finalizarriego2();
                }
        }
        else if (RI2.start==1){
                RI2.start=0;
                if (estacion2==1){
                        timer2.umin2=0;
                        timer2.dmin2=0;
                        timer2.cmin2=0;
                }
                else{
                        estacion2=1;
                        iniciarriego2();
                }
                RI2.temporizar=1;
        }
        else if (C2.activar==1){
                C2.activar=0;
                estacion2=1;
                iniciarriego2();
        }
}
```

```
}
/********************************** ISR TIMERB ***********************************/
#pragma vector=TIMERB0_VECTOR
__interrupt void TB0_ISR(void)
{
        if (TBCCR0==8192) {
                P3DIR &= ~0x40;         // cambio la configuración de p3.6 de salida a entrada
        }
        if (TBCCR0==36864) {
                z=0; TBCTL |= 0x04; TBCTL &= 0xFFEF;  //desactivo timer, borro TBR
        }
        if (TBCCR0==82){
                v=0; w=0; x=0; y=0; TBCTL |= 0x04; TBCTL &= 0xFFEF;  //desactivo timer, borro TBR
        }
}
/****************************** ISR TIMERA *********************************/
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void)
{
        switch (TAIV)    // Discriminar entre los posibles valores del vector de interrupción
        {
        case  2:  break;  // TACCR1, no lo uso
        case  4:  break;  // TACCR2, no lo uso
        case 10:        // vector para la detección de overflow
                {interrupciones++;
                if (interrupciones==2) {
                        interrupciones=0;
                        timere.umine++;
                        if (timere.umine==10) {
                                timere.dmine++;
                                timere.umine=0;
                        }
                        if ((C1.activarp==1) || (estacion1==1)){
                                timer1.umin1++;
                                if (timer1.umin1==10) {
                                        timer1.dmin1++;
                                        timer1.umin1=0;
                                        if (timer1.dmin1==10){
                                                timer1.cmin1++;
                                                timer1.dmin1=0;
                                                if (timer1.cmin1==10) {
                                                        timer1.cmin1=0;
                                                }
                                        }
                                }
                        }
                if ((C2.activarp==1) || (estacion2==1)){
                                timer2.umin2++;
                                if (timer2.umin2==10){
                                        timer2.dmin2++;
                                        timer2.umin2=0;
                                        if (timer2.dmin2==10){
                                                timer2.cmin2++;
```

```
                                                timer2.dmin2=0;
                                                if (timer2.cmin2==10) {
                                                        timer2.cmin2=0;
                                                }
                                        }
                                }
                        }
                        if ((timere.dmine==3)||((C1.activarp==1)||
                           (estacion1==1))||((C2.activarp==1)||(estacion2==1))){
                                __bic_SR_register_on_exit(LPM3_bits);      //salida del bajo consumo
                        }
                }
                break;
                }
        }
}
/****************************** USCI A0/B0 Receive ISR *********************************/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
        switch (tiporespuesta)
        {
        case 1:
                *ptpaste=UCA0RXBUF;
                if (*ptpaste==0x0A){
                        contlf++;
                }
                if (contlf==4){
                        IE2 &= ~UCA0RXIE;                       // deshabilitar interrupción RX
                        contlf=0;
                        __bic_SR_register_on_exit(LPM3_bits); // salida del bajo consumo
                }
                ptpaste++;
                break;
                case 2:
                        *ptpaste=UCA0RXBUF;
                        if (*ptpaste==0x0A){
                                contlf++;
                        }
                        if (contlf==2){
                                IE2 &= ~UCA0RXIE;
                                contlf=0;
                                __bic_SR_register_on_exit(LPM3_bits);
                        }
                        ptpaste++;
                        break;
                case 3:
                        i++;
                        if (i<7){
                                *ptpaste=UCA0RXBUF;
                                if (i==6){
                                        if (*ptpaste=='G'){
                                                haysms=1;
```

```
                        }
                        else {
                                haysms=0;
                        }
                }
                ptpaste++;
        }
        else if (haysms==0){
                if (UCA0RXBUF==0x0A){
                        IE2 &= ~UCA0RXIE;
                        i=0;
                        __bic_SR_register_on_exit(LPM3_bits);
                }
        }
        else if (UCA0RXBUF==0x0A){
                contlf++;
                if (contlf==4){
                        if (i>105){
                                caracteresmax=1;
                        }
                        else{
                                caracteresmax=0;
                        }
                        contlf=0;
                        i=0;
                        IE2 &= ~UCA0RXIE;
                        __bic_SR_register_on_exit(LPM3_bits);
                }
        }
        else if ((i>22)&(i<37)){
                if (i==23){
                        ptpaste=ntelefono;
                }
                *ptpaste=UCA0RXBUF;
                ptpaste++;
        }
        else if ((contlf==1)&(i<99)){
                if (i==65){
                        ptpaste=texto;
                }
                *ptpaste=UCA0RXBUF;
                ptpaste++;
        }
        break;
    }
}
```

```
/****************************** USCI A0/B0 Transmit ISR *********************************/
#pragma vector=USCIAB0TX_VECTOR
__interrupt void USCI0TX_ISR(void)
{
        ptcopy++;
        if (*ptcopy!=0){
                UCA0TXBUF = *ptcopy;
        }
        else {
                IE2 &= ~UCA0TXIE;                       // Deshabilitar interrupción TX
                __bic_SR_register_on_exit(LPM3_bits);  // Salir del modo de bajo consumo
        }
}
/************************************ READ SEGMENT ****************************/
#include  "msp430f2252.h"
extern char  *Flash_ptr;
void read_segment(int address, char *pt) //pt apunta a una cadena
{
        Flash_ptr = (char*)address; // Inicializar puntero a Flash, cast de punteros (flash_ptr se inicializa
                                    // con el valor de address, esta dirección contiene un dato tipo char).
        *pt=*Flash_ptr;             // Escribir valor a la dirección apuntada por pt
        while (*Flash_ptr!=0){      // Bucle hasta terminar de copiar la cadena
                Flash_ptr++;
                pt++;
                *pt = *Flash_ptr;
        }
}
/************************************ ERASE SEGMENT ****************************/
#include  "msp430f2252.h"
extern char *Flash_ptr;
void erase_segment(int address)

{
         __bic_SR_register(GIE);              // Deshabilito las interrupciones, si se produjera una el
                                              // controlador de flash responde con 0x3FF, valor que leería
                                              // la CPU como vector de interrupción.
        Flash_ptr = (char*)address;
         FCTL3 = FWKEY;                       // LOCK bit=0
        FCTL1 = FWKEY + ERASE;               // Erase bit=1
        *Flash_ptr = 0;                      // Dummy write para borrar el segmento flash
         FCTL3 = FWKEY + LOCK;               // LOCK bit=1
        __bis_SR_register(GIE);              // Habilito interrupciones
}
/********************************* WRITE SEGMENT ***********************************/
#include  "msp430f2252.h"
extern   char *Flash_ptr;
void write_segment(int address, char *pt)  //pt apunta a una cadena
{
        __bic_SR_register(GIE);        // Deshabilito las interrupciones, si se produjera una el
                                       // controlador de flash responde con 0x3FF, valor que leería
                                       // la CPU como vector de interrupción
        Flash_ptr = (char*)address;  // Inicializar puntero a Flash, cast de punteros (flash_ptr se inicializa
                                     // con el valor de address, esta dirección contiene un dato tipo char).
```

```
        FCTL3 = FWKEY;              // Lock bit=0
        FCTL1 = FWKEY + WRT;        // WRT bit=1
        *Flash_ptr = *pt;           // Escribir valor en flash
        while (*pt!=0){             // Bucle hasta terminar de copiar la cadena
                Flash_ptr++;
                pt++;
                *Flash_ptr = *pt;
        }
        FCTL1 = FWKEY;              // WRT bit=0
        FCTL3 = FWKEY + LOCK;       // LOCK bit=1
        __bis_SR_register(GIE);     // Habilito interrupciones
}
/************************************ INICIAR RIEGO 1 ********************************/
#include "msp430f2252.h"
extern char y;
void iniciarriego1(void){
        P4OUT |= 0x01;          // p4.0=1, cierro MOSFET
        P4OUT |= 0x04;          // p4.2=1, out1 high, out2 low
        y=1;
        TBCCR0 = 82;            // temporizo 20ms (1/(32768/8))*82=20ms
        TBCTL |= MC_1;          // activación timer, MCx=01, up mode
        while (y==1);
        P4OUT &= ~0x04;         // P4.2=0, out1 low, out2 low
        P4OUT &= ~0x01;         // p4.0=0, abro MOSFET
}
/************************************ INICIAR RIEGO 2 ********************************/
#include "msp430f2252.h"
extern char v;
void iniciarriego2(void){
        P4OUT |= 0x08;                  //p4.3=', cierro MOSFET
        P4OUT |= 0x20;                  //p4.5=1, out1 high, out2 low
        v=1;
        TBCCR0 = 82;                    // temporizo 20ms (1/(32768/8))*82=20ms
        TBCTL |= MC_1;                  // activación timer, MCx=01, up mode
        while (v==1);
        P4OUT &= ~0x20;                 //P4.5=0, out1 low, out2 low
        P4OUT &= ~0x08;                 //p4.3=0, abro MOSFET
}
/************************************ FINALIZAR RIEGO 1 ********************************/
#include "msp430f2252.h"
extern char x;
void finalizarriego1(void){
        P4OUT |= 0x01;                  //p4.0='1', cierro MOSFET
        P4OUT |= 0x02;                  //p4.1='1', out1 high, out2 lo
        x=1;
        TBCCR0 = 82;                    // temporizo 20ms (1/(32768/8))*82=20ms
        TBCTL |= MC_1;                  // activación timer, MCx=01, up mode
        while (x==1);
        P4OUT &= ~0x02;                 //P4.1='0', out1 low, out2 low
        P4OUT &= ~0x01;                 //p4.0='0', abro MOSFET
}
```

```
/************************************ FINALIZAR RIEGO 2 ******************************/
#include "msp430f2252.h"
extern char w;
void finalizarriego2(void){
        P4OUT |= 0x08;                  //p4.3='1', cierro MOSFET
        P4OUT |= 0x10;                  //p4.4='1', out1 high, out2 lo
        w=1;
        TBCCR0 = 82;                    // temporizo 20ms (1/(32768/8))*82=20ms
        TBCTL |= MC_1;                  // activación timer, MCx=01, up mode
        while (w==1);
        P4OUT &= ~0x10;                 //P4.4='0', out1 low, out2 low
        P4OUT &= ~0x08;                 //p4.3='0', abro MOSFET
}
/******************************** COMPARAR DOS NUM BCD ****************************/
#include "msp430f2252.h"
extern char mayor;
char comparardosnumBCD(char *ptcmp1, char *ptcmp2){
        if (*ptcmp1<=*ptcmp2){
                if (*ptcmp1==*ptcmp2){
                        ptcmp1++;
                        ptcmp2++;
                        if (*ptcmp1<=*ptcmp2){
                                if (*ptcmp1==*ptcmp2){
                                        ptcmp1++;
                                        ptcmp2++;
                                        if (*ptcmp1<=*ptcmp2){
                                                if (*ptcmp1==*ptcmp2){
                                                        mayor=3;
                                                }
                                                else{
                                                        mayor=2;
                                                }
                                        }
                                        else{
                                                mayor=1;
                                        }
                                }
                                else{
                                        mayor=2;
                                }
                        }
                        else {
                                mayor=1;
                        }
                }
                else {
                        mayor=2;
                }
        }
        else {
                mayor=1;
        }
        return mayor;
```

```
}
/********************************** ENCENDER GSM `**********************************/
#include  "msp430f2252.h"
extern char z;
void encenderGSM(void)
{
        //P4OUT = 0x40;                 //encender buck, (la placa de Libelium tiene Vcc=12V)
        while (P3IN&0x40==0x00);        // si p3.6 está a '1' ya tengo tensión en POK_IN
        P3DIR |= 0x40;                  // cambio la configuración de p3.6 de entrada a salida
        P3OUT = 0;                      // pok_in a '0'
        z=1;
        TBCCR0 = 8192;                  // temporizo 2 seg, 1/(32768/8)*8192=2sg
        TBCCTL0 = CCIE;
        TBCTL |= MC_1;                  // activación timer, MCx=01, up mode
        while (P3DIR==0xFF);
        TBCCR0=36864;                   // temporizo 7 seg, 1/(32768/8)*(28672+8192)=(7sg + 2sg)
        while (z==1);
}
/********************************** ENVIAR AT ***********************************/
#include "borrarstring.h"
#include "msp430f2252.h"
const char ATinicial[4]={"AT\r"};
extern char *ptcopy;
extern char *ptpaste;
extern char texto[40];
extern char tiporespuesta;
void enviarAT(void)
{
        UCA0CTL1 &= ~UCSWRST;                   // Habilitar USCI
        ptcopy = (char*)&ATinicial[0];
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                        // Habilitar interrupción TX
         __bis_SR_register(LPM3_bits + GIE);    // Entrar en modo de bajo consumo, habilitar
                                                interrupciones
         ptpaste=texto;
         tiporespuesta=2;
         IE2 |= UCA0RXIE;                       // habilitar interrupción RX
        __bis_SR_register(LPM3_bits + GIE);
        ptpaste=texto;
         borrarstring();
}
/********************************** BORRAR STRING ******************************/
#include "msp430f2252.h"
extern char *ptpaste;
void borrarstring(void){
        while (*ptpaste!=0){
                *ptpaste=0;
                ptpaste++;
        }
}
```

```
/************************************ INT. PIN ************************************/
#include  "msp430F2252.h"
#include  "stdio.h"
#include "borrarstring.h"
extern char *ptpaste;
extern char *ptcopy;
extern char texto[40];
extern char contlf;
extern char tiporespuesta;
extern char i;
const char ATpin[16]={"AT+CPIN=\"4923\"\r"};
void intpin(void)
{
        ptcopy = (char*)&ATpin[0];
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                        // habilitar interrupción TX
        __bis_SR_register(LPM3_bits + GIE);     // entrada en bajo consumo, interrupciones habilitadas
        ptpaste=texto;
        tiporespuesta=2;
        IE2 |= UCA0RXIE;                        // habilitar interrupción RX
        __bis_SR_register(LPM3_bits + GIE);
        ptpaste=texto;
        borrarstring();
}
/************************************ LEER SMS ************************************/
#include  "msp430f2252.h"
#include  "stdio.h"
#include "borrarstring.h"
extern char tiporespuesta;
extern char texto[40];
extern char *ptcopy;
extern char *ptpaste;
extern char i;
const char ATleer[11]={"AT+CMGR=N\r"};
char ATleerRAM[11]={""};
void leerSMS(char contsmsleidos)
{
        for (i=0; i<12; i++){                   //Copiar ATleer de flash a RAM
                ATleerRAM[i]=ATleer[i];
        }
        i=0;
        ATleerRAM[8]=contsmsleidos;             //Copiar el valor de la variable contsmsleidos a RAM
        ptcopy=ATleerRAM;
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                        // Habilitar interrupción TX
        __bis_SR_register(LPM3_bits + GIE);     // Entrada en bajo consumo, interrupciones habilitadas
        tiporespuesta=3;
        ptpaste=texto;
        IE2 |= UCA0RXIE;                        // Habilitar interrupción RX
        __bis_SR_register(LPM3_bits + GIE);
}
```

```
/*********************************** GUARDAR SMS ********************************/
#include "borrarstring.h"
extern char *ptcopy;
extern char *ptpaste;
extern char caracteresmax;
char contbarras=0;
extern char i;
extern char clave[7];
extern char comando[4];
extern char configuracion[23];
extern char texto[40];
void guardarSMS(void)
{
	ptcopy=texto;
	if (caracteresmax==1){
		*ptcopy=0x0D;
	}
	while(*ptcopy!=0x0D){
		i++;
		if (*ptcopy=='/'){
			contbarras++;
			ptcopy++;
		}
		else if (contbarras==1){
			if (i>7){
				*ptcopy=0x0D;
				ptpaste=clave;
				borrarstring();
			}
			else {
				if (clave[0]==0){
					ptpaste=clave;
				}
				*ptpaste=*ptcopy; ptcopy++; ptpaste++;
			}
		}

		else if (contbarras==2){
			if(i>13){
				*ptcopy=0x0D;
				ptpaste=comando;
				borrarstring();
			}
			else {
				if (comando[0]==0) {
					ptpaste=comando;
				}
				*ptpaste=*ptcopy; ptcopy++; ptpaste++;
			 }
		}
		else if (contbarras==3) {
			if (i>32){
				*ptcopy=0x0D;
```

```
                        ptpaste=configuracion;
                        borrarstring();
                }
                else {
                        if (configuracion[0]==0) {
                                ptpaste=configuracion;
                        }
                        *ptpaste=*ptcopy; ptcopy++; ptpaste++;
                }
        }
        else {
                ptcopy++;
        }
    }
    contbarras=0;
    i=0;
    ptpaste=&texto[0];
    borrarstring();
}
/************************************ PROCESAR SMS *********************************/
#include "msp430f2252.h"
#include "string.h"
#include "compcomando.h"
#include "compguardarconf.h"
#include "acciones.h"
#include "borrarstring.h"
#include "enviarSMS.h"
extern char clave[7];
extern char passwordRAM[7];
extern char ntelefono[15];
extern char usuario1RAM[15];
extern char usuario2RAM[15];
extern char comando[4];
extern char configuracion[23];
extern char alta;
extern char datoscorrectos;
extern char datosincorrectos;
extern char a;
extern char caracteresmax;
extern char *ptpaste;

void procesarSMS(void){
    if (strcmp(ntelefono,usuario1RAM)==0){
        alta=0;
    }
    else if (strcmp(ntelefono,usuario2RAM)==0){
        alta=0;
    }
    else alta=1;
    if (caracteresmax==0){
        if (strcmp(clave,passwordRAM)==0){
            compcomando();
            if (a!=0){
```

```
                                compguardarconf();
                                if (datoscorrectos==1){
                                        datoscorrectos=0;
                                        acciones();
                                        a=0;
                                }
                                else{
                                        datosincorrectos=0;
                                        enviarSMS('5');   // envío del SMS CONFIGURACIÓN
                                                          //INCORRECTA
                                        }
                        }
                        else {
                                if (alta==0){
                                        enviarSMS('3');   // envío del SMS COMANDO INCORRECTO
                                        }
                                else{
                                        enviarSMS('2');   // envío del SMS COMANDO INCORRECTO O
                                                          //USUARIO NO REGISTRADO
                                        }
                        }
                }
                else {
                        if (alta==0){
                                enviarSMS('1');    //envío del SMS CLAVE INCORRECTA
                                }
                }
        }
        else {
                if (alta==0){                           //envío del SMS CARACTERES MÁXIMOS SUPERADOS
                        enviarSMS('7');
                }
        }
        ptpaste=clave;
        borrarstring();
        ptpaste=ntelefono;
        borrarstring();
        ptpaste=comando;
        borrarstring();
        ptpaste=configuracion;
        borrarstring();
}
/******************************** COMPROBAR COMANDO ****************************/
#include "msp430f2252.h"
#include "string.h"
extern char alta;
extern char comando[4];
extern char cgdatos;
extern char a;
void compcomando(void)
{
        if (alta==0){
                if (strcmp (comando, "P")==0) {cgdatos=2; a=2;}
```

```
                else if (strcmp (comando, "RA")==0) {cgdatos=3; a=3;}
                else if (strcmp (comando, "C")==0) {cgdatos=4; a=4;}
                else if (strcmp (comando, "1")==0) {cgdatos=5; a=5;}
                else if (strcmp (comando, "2")==0) {cgdatos=6; a=6;}
                else if (strcmp (comando, "3")==0) {cgdatos=7; a=7;}
        }
        else {if (strcmp (comando, "SU")==0) {cgdatos=1; a=1;}}
}
/*********************  COMPROBAR GUARDAR CONFIGURACIÓN  ***********************/
#include "msp430F2252.h"
#include "pasadeASCIIaBCD.h"
#include "erase_segment.h"
#include "write_segment.h"
#define info_seg_D 0x1000
extern char diasemana;
int comprobar=0;
extern char cgdatos;
extern char configuracion[23];
extern char datoscorrectos;
extern char datosincorrectos;
extern char *ptcopy;
extern char *ptpaste;
extern char passwordRAM[7];
extern char sector;
extern struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
}C0;
void compguardarconf(void){
        switch (cgdatos)
        {
        case  1:
                datoscorrectos=1;
                break;

        case  2:
                if ((configuracion[7]==0)&(configuracion[6]=='.')){
                        ptcopy=configuracion;
                        ptpaste=passwordRAM;
                        while (*ptcopy!='.'){
                                *ptpaste=*ptcopy;
                                ptpaste++;
                                ptcopy++;}
                        datoscorrectos=1;
                        erase_segment(info_seg_D);
                        write_segment(info_seg_D,&passwordRAM[0]);
                }
                break;
        case  3:
                datoscorrectos=1;
                break;
```

```
case 4:
        if ((configuracion[1]==' ')&(configuracion[3]==' ')&(configuracion[11]==' ')&
        (configuracion[17]==' ')&(configuracion[21]=='.')&(configuracion[22]==0)){
                if ((configuracion[0]>'0')&(configuracion[0]<'8')){
                        diasemana=configuracion[0];
                        if ((configuracion[2]>'0')&(configuracion[2]<'4')){
                                sector=configuracion[2];
                        }
                        ptcopy=&configuracion[4];
                        ptpaste=&C0.diasriegosemana[0];
                        while ((*ptcopy!=' ')&(datosincorrectos==0)){
                                if ((*ptcopy>='0')&(*ptcopy<'8')){
                                        *ptpaste=*ptcopy;
                                        ptcopy++;
                                        ptpaste++;
                                }
                                else datosincorrectos=1;
                        }
                        if (datosincorrectos==0){
                                comprobar=((configuracion[12]<<8)+(configuracion[13]));
                                if ((comprobar>=0x3030)&(comprobar<0x3234)){
                                        C0.horariego[0]=configuracion[12];
                                        C0.horariego[1]=configuracion[13];
                                        comprobar=((configuracion[15]<<8)
                                        +(configuracion[16]));
                                        if ((comprobar>=0x3030)&(comprobar<=0x3539)){
                                                C0.minriego[0]=configuracion[15];
                                                C0.minriego[1]=configuracion[16];
                                                ptcopy=&configuracion[18];
                                                ptpaste=&C0.tiemporiego[0];
                                                while ((*ptcopy!='.') &
                                                (datosincorrectos==0)){
                                                        if ((*ptcopy>='0')&(*ptcopy<='9')){
                                                                *ptpaste=*ptcopy;
                                                                ptpaste++;
                                                                ptcopy++;
                                                        }
                                                        else datosincorrectos=1;
                                                }
                                                if (datosincorrectos==0){
                                                        datoscorrectos=1;
                                                }
                                        }
                                }
                        }
                }
        }
        break;
case 5:
        if ((configuracion[1]==' ')&(configuracion[5]=='.')&(configuracion[6]==0)){
                if ((configuracion[0]>'0')&(configuracion[0]<'4')){
                        sector=configuracion[0];
                        ptcopy=&configuracion[2];
```

```
                                        ptpaste=C0.tiemporiego;
                                        while ((*ptcopy!='.')&(datosincorrectos==0)){
                                                if ((*ptcopy>='0')&(*ptcopy<='9')){
                                                        *ptpaste=*ptcopy;
                                                        ptpaste++;
                                                        ptcopy++;
                                                }
                                                else datosincorrectos=1;
                                                                }
                                        if (datosincorrectos==0){
                                                datoscorrectos=1;
                                        }
                                }
                        }
                        break;
                case 6:
                        if ((configuracion[1]=='.')&(configuracion[2]==0)){
                                if ((configuracion[0]>'0')&(configuracion[0]<'4')){
                                        sector=configuracion[0];
                                        datoscorrectos=1;
                                }
                        }
                        break;
                case 7:
                        if ((configuracion[1]=='.')&(configuracion[2]==0)){
                                if ((configuracion[0]>'0')&(configuracion[0]<'4')){
                                        sector=configuracion[0];
                                        datoscorrectos=1;
                                }
                        }
                        break;
                }
}
/************************************ ACCIONES ************************************/
#include "msp430F2252.h"
#include "string.h"
#include "borrarstring.h"
#include "enviarSMS.h"
#include "preguntarfechayhora.h"
#include "erase_segment.h"
#include "write_segment.h"
#include "pasadeASCIIaBCD.h"
#define info_seg_B 0x1080
#define info_seg_C 0x1040
extern struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
}C0;
extern struct {
        char diasriegosemana[8];
        char horariego[3];
```

```
        char minriego[3];
        char tiemporiego[4];
        char tiempopararriego[5];
        char activar;
        char activarp;
        char configuracion;
        char heregadohoy;
}C1;
extern struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
        char tiempopararriego[5];
        char activar;
        char activarp;
        char configuracion;
        char heregadohoy;
}C2;
extern struct {
        char temporizar;
        char tiemporiego[4];
        char start;
}RI1;
extern struct {
        char temporizar;
        char tiemporiego[4];
        char start;
}RI2;
extern char sector;
extern char stop1;
extern char stop2;
extern char ntelefono[15];
extern char usuario1RAM[15];
extern char usuario2RAM[15];
extern char a;
extern char *ptpaste;
extern char *ptcopy;
char numASCII=0;
char numBCD=0;
char cero[2]={0,0};
void acciones(void){
        switch (a)
        {
        case 1:
                if (usuario1RAM[0]!='') {
                        strcpy (usuario1RAM,ntelefono);
                        erase_segment(info_seg_B);
                        write_segment(info_seg_B,usuario1RAM);
                }
                else if (usuario2RAM[0]!='') {
                        strcpy (usuario2RAM,ntelefono);
                        erase_segment(info_seg_B);
```

```
                write_segment(info_seg_C,usuario1RAM);
        }
        else {
                enviarSMS('4');
        }
        break;
case 2:
        break;
case 3:
        if (strcmp(ntelefono,usuario1RAM)==0){
                erase_segment(info_seg_B);
                write_segment(info_seg_B, cero);
                ptpaste=usuario1RAM;
        }
        else {
                erase_segment(info_seg_C);
                write_segment(info_seg_C, cero);
                ptpaste=usuario2RAM;
        }
        borrarstring();
        break;
case 4:
        if ((sector=='1')||(sector=='3')){
                C1.configuracion=1;
                strcpy(C1.diasriegosemana, C0.diasriegosemana);
                strcpy(C1.horariego, C0.horariego);
                strcpy(C1.minriego, C0.minriego);
                strcpy(C1.tiemporiego, C0.tiemporiego);
                ptcopy=C1.tiemporiego;
                while (*ptcopy!=0){
                        numASCII=*ptcopy;
                        pasadeASCIIaBCD(numASCII);
                        *ptcopy=numBCD;
                        ptcopy++;
                }
        }
        if ((sector=='2')||(sector=='3')){
                C2.configuracion=1;
                strcpy(C2.diasriegosemana, C0.diasriegosemana);
                strcpy(C2.horariego, C0.horariego);
                strcpy(C2.minriego, C0.minriego);
                strcpy(C2.tiemporiego, C0.tiemporiego);
                ptcopy=C2.tiemporiego;
                while (*ptcopy!=0){
                        numASCII=*ptcopy;
                        pasadeASCIIaBCD(numASCII);
                        *ptcopy=numBCD;
                        ptcopy++;
                }
        }
        preguntarfechayhora();
        break;
case 5:
```

```
                if ((sector=='1')||(sector=='3')){
                        RI1.start=1;
                        stop1=0;
                        strcpy(RI1.tiemporiego,C0.tiemporiego);
                        ptcopy=RI1.tiemporiego;
                        while (*ptcopy!=0){
                                numASCII=*ptcopy;
                                pasadeASCIIaBCD(numASCII);
                                *ptcopy=numBCD;
                                ptcopy++;
                        }
                }
                if ((sector=='2')||(sector=='3')){
                        RI2.start=1;
                        stop2=0;
                        strcpy(RI2.tiemporiego,C0.tiemporiego);
                        ptcopy=RI2.tiemporiego;
                        while (*ptcopy!=0){
                                numASCII=*ptcopy;
                                pasadeASCIIaBCD(numASCII);
                                *ptcopy=numBCD;
                                ptcopy++;
                        }
                }
                break;
        case 6:
                if ((sector=='1')||(sector=='3')){
                        RI1.start=0;
                        stop1=1;
                        C1.configuracion=0;
                }
                if ((sector=='2')||(sector=='3')){
                        RI2.start=0;
                        stop2=1;
                        C2.configuracion=0;
                }
                break;
        case 7:
                if ((sector=='1')||(sector=='3')){
                        C1.configuracion=1;
                }
                if ((sector=='2')||(sector=='3')){
                        C2.configuracion=1;
                }
                break;
        }
}
/********************************** ENVIAR SMS **********************************/
#include  "msp430F2252.h"
#include  "stdio.h"
#include "borrarstring.h"
extern char *ptpaste;
extern char texto[40];
```

```
extern char contlf;
extern char tiporespuesta;
extern char ntelefono[15];
extern char *ptcopy;
extern char i;
const char ATenviar[28]={"AT+CMSS=N,00000000000000\r"};
char ATenviarRAM[28]={""};
void enviarSMS(char nsms)
{
        for (i=0; i<28; i++){
                ATenviarRAM[i]=ATenviar[i];
        }
        i=0;
        ATenviarRAM[8]=nsms;
        ptcopy=ntelefono;
        ptpaste=&ATenviarRAM[10];
        while (*ptcopy!=0){
                *ptpaste=*ptcopy;
                ptpaste++;
                ptcopy++;
        }
        ptcopy=ATenviarRAM;
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                    // Habilitar interrupciones TX
        __bis_SR_register(LPM3_bits + GIE);
        ptpaste=texto;
        tiporespuesta=1;
        IE2 |= UCA0RXIE;
        __bis_SR_register(LPM3_bits + GIE);
        ptpaste=texto;
        borrarstring();
}
/************************************ PASA DE ASCII A BCD ******************************/
#include "msp430f2252.h"
extern char numBCD;
char pasadeASCIIaBCD(char numASCII){
  return numBCD=numASCII-48;
}
/****************************** PREGUNTAR FECHA Y HORA ******************************/
#include  "msp430f2252.h"
#include  "stdio.h"
#include  "borrarstring.h"
extern char tiporespuesta;
extern char texto[40];
extern char *ptcopy;
extern char *ptpaste;
extern struct {
        char diamesactual[3];
        char hora[3];
        char min[3];
}reloj;
extern char diames[3];
extern char a;
```

```
extern char i;
const char ATreloj[11]={"AT+CCLK?\r"};
void preguntarfechayhora(void)
{
        ptcopy = (char*)ATreloj;
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                        // Habilitar interrupciónTX
        __bis_SR_register(LPM3_bits + GIE);     // Entrar en bajo consumo, habilitar interrupciones
        ptpaste=texto;
        tiporespuesta=1;
        IE2 |= UCA0RXIE;                        // Habilitar interrupciónRX
        __bis_SR_register(LPM3_bits + GIE);


        if (a==0){
                reloj.diamesactual[0]=texto[16];
                reloj.diamesactual[1]=texto[17];
                reloj.hora[0]=texto[19];
                reloj.hora[1]=texto[20];
                reloj.min[0]=texto[22];
                reloj.min[1]=texto[23];
        }
        else {
                diames[0]=texto[16];
                diames[1]=texto[17];
        }
        ptpaste=texto;
        borrarstring();
}
/********************************** BORRAR SMS **********************************/
#include  "msp430f2252.h"
#include  "stdio.h"
#include "borrarstring.h"
extern char tiporespuesta;
extern char *ptpaste;
extern char texto[40];
extern char contlf;
extern char *ptcopy;
extern char i;
const char ATborrar[13]={"AT+CMGD=0,1\r"};
void borrarSMS(void)
{
        ptcopy = (char*)&ATborrar[0];
        UCA0TXBUF=*ptcopy;
        IE2 |= UCA0TXIE;                        // Habilitar interrupción TX
        __bis_SR_register(LPM3_bits + GIE);     // Entrar en bajo cosumo, interrupciones habilitadas
        ptpaste=texto;
        tiporespuesta=2;
        IE2 |= UCA0RXIE;
        __bis_SR_register(LPM3_bits + GIE);
        ptpaste=texto;
        borrarstring();
}
```

```
/******************************** TIEMPO PARA RIEGO 1 ********************************/
#include "msp430f2252.h"
extern struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
        char tiempoparariego[5];
}C1;
extern struct {
        char min[3];
        char minriego[3];
        char horariego[3];
        char hora[3];
}OP;
void tparariego1(char *pmin, char *pminriego, char *phorariego, char *phora){
        OP.min[1]=*pmin;
        if (*pmin>*pminriego){
                OP.minriego[1]=*pminriego+10;
                pmin--;
                pminriego--;
                OP.min[0]=*pmin+1;
        }
        else{
                OP.minriego[1]=*pminriego;
                pmin--;
                pminriego--;
                OP.min[0]=*pmin;
        }
        if (*pmin>*pminriego){
                OP.minriego[0]=*pminriego+6;
                OP.hora[1]=*phora+1;
        }
        else {
                OP.minriego[0]=*pminriego;
                OP.hora[1]=*phora;
        }
        if (*phora>*phorariego){
                OP.horariego[1]=*phorariego+10;
                phora--;
                OP.hora[0]=*phora+1;
                phorariego--;
        }
        else{
                OP.horariego[1]=*phorariego;
                phora--;
                OP.hora[0]=*phora;
                phorariego--;
        }
        OP.horariego[0]=*phorariego;
        C1.tiempoparariego[0]=OP.horariego[0]-OP.hora[0];
        C1.tiempoparariego[1]=OP.horariego[1]-OP.hora[1];
        C1.tiempoparariego[2]=OP.minriego[0]-OP.min[0];
```

```
        C1.tiempopararriego[3]=OP.minriego[1]-OP.min[1];
}
/********************************* TIEMPO PARA RIEGO 2 ****************************/
#include "msp430f2252.h"
extern struct {
        char diasriegosemana[8];
        char horariego[3];
        char minriego[3];
        char tiemporiego[4];
        char tiempopararriego[5];
}C2;
extern struct {
        char min[3];
        char minriego[3];
        char horariego[3];
        char hora[3];
}OP;
void tpararriego2(char *pmin, char *pminriego, char *phorariego, char *phora){
        OP.min[1]=*pmin;
        if (*pmin>*pminriego){
                OP.minriego[1]=*pminriego+10;
                pmin--;
                pminriego--;
                OP.min[0]=*pmin+1;
        }
        else{
                OP.minriego[1]=*pminriego;
                pmin--;
                pminriego--;
                OP.min[0]=*pmin;
        }
        if (*pmin>*pminriego){
                OP.minriego[0]=*pminriego+6;
                OP.hora[1]=*phora+1;
        }
        else {
                OP.minriego[0]=*pminriego;
                OP.hora[1]=*phora;
        }
        if (*phora>*phorariego){
                OP.horariego[1]=*phorariego+10;
                phora--;
                OP.hora[0]=*phora+1;
                phorariego--;
        }
        else{
                OP.horariego[1]=*phorariego;
                phora--;
                OP.hora[0]=*phora;
                phorariego--;
        }
        OP.horariego[0]=*phorariego;
        C2.tiempopararriego[0]=OP.horariego[0]-OP.hora[0];
```

```
        C2.tiempopararriego[1]=OP.horariego[1]-OP.hora[1];
        C2.tiempopararriego[2]=OP.minriego[0]-OP.min[0];
        C2.tiempopararriego[3]=OP.minriego[1]-OP.min[1];
}
```

ÍNDICE DE ILUSTRACIONES

Ilustración 1: IDEA GLOBAL DEL PROYECTO 10
Ilustración 2: EFICIENCIA BUCK VS OUTPUT CURRENT 13
Ilustración 3: DIAGRAMA DE BLOQUES SISTEMA FINAL 14
Ilustración 4: Ilustración 4: DESIGNACIÓN DE PINES EN EL MSP430AF251 17
Ilustración 5: $\boldsymbol{IQuiescent}$ VS $\boldsymbol{I0}$ 18
Ilustración 6: DISTRIBUCIÓN DE LOS PINES EN UNA TARJETA DE 6 CONTACTOS 19
Ilustración 7: DISTRIBUCIÓN DE LOS PINES EN UNA TARJETA DE 8 CONTACTOS 19
Ilustración 8: ACCIÓN DEL CAMPO MAGNÉTICO CREADO POR EL SOLENOIDE SOBRE EL ÉMBOLO 20
Ilustración 9: CIRCUITO SOLENOIDE LATCH DE 3 TERMINALES 23
Ilustración 10: CIRCUITO SOLENOIDE LATCH DE DOS TERMINALES 24
Ilustración 11: CONEXIONADO DEL SOLENOIDE DE TRES TERMINALES EN EL PUENTE EN H 25
Ilustración 12: CONEXIONADO DEL SOLENOIDE DE DOS TERMINALES EN EL PUENTE EN H 25
Ilustración 13: ADAPTACIÓN DE NIVELES ENTRE EL TA72919 Y EL µC 27
Ilustración 14: CORRIENTE VS TIEMPO CARGA RL 28
Ilustración 15: PARÁMETROS ETAPA DE SALIDA CON SOLENOIDE TRES TERMINALES 29
Ilustración 16: PARÁMETEROS ETAPA DE SALIDA CON SOLENOIDE DE DOS TERMINALES 30
Ilustración 17: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN ZENER 33
Ilustración 18: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN BUFFER 34
Ilustración 19: ADAPTACIÓN DE LOS NIVELES TX MICRO Y RX GSM MEDIANTE UN SCHOTTKY 35
Ilustración 20: DIAGRAMA DE BLOQUES DEL CIRCUITO 36
Ilustración 21: DIAGRAMA DE BLOQUES DEL INTEGRADO LT3680 37
Ilustración 22: CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 38
Ilustración 23: CIRCUITO EQUIVALENTE DEL CONDENSADOR 40
Ilustración 24: IMPEDANCIA DE UN CONDENSADOR VS FRECUENCIA 41
Ilustración 25: VARIACIÓN DE LA CAPACIDAD DE UN CONDENSADOR YV5 VS TEMPERATURA 41
Ilustración 26: CIRCUITO EQUIVALENTE DE UNA PISTA DE CIRCUITO IMPRESO 42
Ilustración 27: IMPEDANCIA DE TRES CONDENSADORES EN PARALELO CON ENCAPSULADO SMD ESCALADO VS FRECUENCIA 42
Ilustración 28: ONDA DE TENSIÓN OBTENIDA EXPERIMENTALMENTE A LA SALIDA DEL CONVERTIDOR 45
Ilustración 29: DIAGRAMA DE BLOQUES DEL MÓDULO GSM 47
Ilustración 30: CIRCUITO DEL CONECTOR DEL MODULO GSM 48
Ilustración 31: DIAGRAMA DE BLOQUES DEL REGULADOR LINEAL 49
Ilustración 32: CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL µC 49
Ilustración 33: DIAGRAMA DE BLOQUES DEL µC 51
Ilustración 34: CIRCUITO DEL µC 51
Ilustración 35: FORMAS DE OBTENER LAS SEÑALES DE RELOJ QUE GOBIERNEN A LOS PERIFÉRICOS 53

Ilustración 36: ETAPA DE ADAPTACIÓN DE TENSIÓN ENTRE TX DEL µC Y RX DEL MÓDULO GSM 54
Ilustración 37: CIRCUITO DE LA ETAPA DE ADAPTACIÓN Y CONTROL 55
Ilustración 38: CIRCUITO PARA LA CONFIGURACIÓN DE PARÁMETROS DEL MÓDULO GSM 57
Ilustración 39: DIAGRAMA DE BLOQUES MAIN1 71
Ilustración 40: DIAGRAMA DE BLOQUES MAIN2 72
Ilustración 41: DIAGRAMA DE BLOQUES MAIN3 73
Ilustración 42: DIAGRAMA DE BLOQUES MAIN4 74
Ilustración 43: DIAGRAMA DE BLOQUES MAIN5 75
Ilustración 44: ESQUEMA COMPLETO DEL CIRCUITO DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 80
Ilustración 45: ESQUEMA COMPLETO DEL CIRCUITO DEL CONECTOR DEL MÓDULO GSM 81
Ilustración 46: ESQUEMA COMPLETO DEL CIRCUITO DEL µC 82
Ilustración 47: ESQUEMA COMPLETO DEL CIRCUITO DE LA ETAPA DE SALIDA 83
Ilustración 48: ESQUEMA DEL CIRUITO DE LA PLACA DE LIBELIUM 84
Ilustración 49: PLANO DE PISTAS CARA TOP DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 86
Ilustración 50: PLANO DE PISTAS CARA BOTTOM DE LA FUENTE DE ALIMENTACIÓN DEL MÓDULO GSM 87
Ilustración 51: PLANO DE PISTAS CARA TOP DEL MÓDULO GSM 88
Ilustración 52: PLANO DE PISTAS CARA BOTTOM DEL MÓDULO GSM 89
Ilustración 53: PLANO DE PISTAS CARA TOP DEL µC 90
Ilustración 54: PLANO DE PISTAS CARA BOTTOM DEL µC 91
Ilustración 55: PLANO DE PISTAS CARA TOP DE LA ETAPA DE SALIDA 92
Ilustración 56: PLANO DE PISTAS CARA BOTTOM DE LA ETAPA DE SALIDA 93
Ilustración 57: TABLA DEL CÓDIGO GSM 7-BIT DEFAULT ALPHABET 107
Ilustración 58: TABLA EXTENDIDA DEL CÓDIGO GSM 7-BIT DEFAULT ALPHABET 107
Ilustración 59: TABLA DEL CÓDIGO IRA (VERSIÓN IRV) 108
Ilustración 60: LEER SEGMENTO 110
Ilustración 61: BORRAR SEGMENTO 111
Ilustración 62: ESCRIBIR SEGMENTO 112
Ilustración 63: INICIAR RIEGO 1 113
Ilustración 64: INICIAR RIEGO 2 114
Ilustración 65: FIN RIEGO 1 115
Ilustración 66: FIN RIEGO 2 116
Ilustración 67: COMPARAR DOS NUMEROS BCD 117
Ilustración 68: ENCENDER GSM 118
Ilustración 69: ENVIAR AT 119
Ilustración 70: BORRAR STRING 120
Ilustración 71: INTRODUCIR PIN 121
Ilustración 72: LEER SMS 122
Ilustración 73: GUARDAR SMS 123
Ilustración 74: PROCESAR SMS 124

Ilustración 75: PROCESAR SMS 125
Ilustración 76: COMPROBAR COMANDO 126
Ilustración 77: COMPROBAR Y GUARDAR CONFIGURACIÓN 127
Ilustración 78: COMPROBAR Y GUARDAR CONFIGURACIÓN 128
Ilustración 79: COMPROBAR Y GUARDAR CONFIGURACIÓN 129
Ilustración 80: COMPROBAR Y GUARDAR CONFIGURACIÓN 130
Ilustración 81: ACCIONES 131
Ilustración 82: ACCIONES 132
Ilustración 83: ACCIONES 133
Ilustración 84: ACCIONES 134
Ilustración 85: ENVIAR SMS 135
Ilustración 86: PASAR DE ASCII A BCD 136
Ilustración 87: PREGUNTAR FECHA Y HORA 137
Ilustración 88: BORRAR SMS 138
Ilustración 89: TIEMPO PARA RIEGO 1 Y 2 139
Ilustración 90: INTERRUPCION TIMER B 140
Ilustración 91: INTERRUPCIÓN TIMER A 141
Ilustración 92: INTERRUPCIÓN USCI0RX 142
Ilustración 93: INTERRUPCION USCI0RX 143
Ilustración 94: USCI0TX 144

ÍNDICE DE TABLAS

Tabla 1: CONSUMOS Y PRECIOS MODEMS GSM TIPO TERMINAL 15
Tabla 2: CONSUMO Y PRECIOS MÓDULOS GSM 16
Tabla 3: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR DOROT 3 WAY 21
Tabla 4: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR DOROT 2 WAY 21
Tabla 5: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDA BERMAD 2WAY 21
Tabla 6: DATOS ELÉCTRICOS SOLENOIDES ESTÁNDAR BERMAD 3 WAY 22
Tabla 7: CARACTERÍSTICAS ELÉCTRICAS SOLENOIDE DE TRES TERMINALES 23
Tabla 8: CARACTERÍSTICAS ELÉCTRICAS SOLENOIDE DE DOS TERMINALES 24
Tabla 9: CARACTERÍSTICAS DRIVER PUENTE COMPLETO 26
Tabla 10: TENSIÓN DE SATURACIÓN TRANSISTORES LADO ALTO DRIVER 29
Tabla 11: PROPIEDADES SOLENOIDE TRES TERMINALES DOROT 30
Tabla 12: TENSIÓN DE SATURACIÓN TRANSISTORES DRIVER 31
Tabla 13: PROPIEDADES SOLENOIDE DOS TERMINALES BERMAD 31
Tabla 14: PROPIEDADES SOLENOIDE DOS TERMINALES BERMAD 32
Tabla 15: INCOMPATIBILIDAD DE LA TENSIÓN ENTRE TX DEL MICRO Y RX DEL MÓDULO 33
Tabla 17: TRANSFORMAR CARÁCTERES "HOLA" DE 7 A 8 BITS 62
Tabla 16: CODIFICACIÓN DE "HOLA" SEGÚN 7 BIT GSM DEFAULT ALPHABET 62