# Comunicación inalámbrica por medio del Protocolo ZigBee para la manipulación de un brazo Robótico

Autores:
Ana María Gil
Héctor Fabio Jiménez

*Grupo de Investigación*
L.I.D.E.R

Universidad Tecnológica de Pereira
Facultad Tecnologías, Tecnología Eléctrica
Pereira, Risaralda
2014

# Comunicación inalámbrica por medio del Protocolo ZigBee para la manipulación de un brazo Robótico

Autores:
Ana María Gil
Héctor Fabio Jiménez

*Grupo de Investigación*
L.I.D.E.R

Proyecto de Grado para optar al Título de Tecnólogo en Electricidad

Director de Proyecto:
Ing. Santiago Gómez Estrada

Universidad Tecnológica de Pereira
Facultad Tecnologías, Tecnología Eléctrica
Pereira, Risaralda
2014

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____

Director Proyecto
*Ingeniero Electricista*
Santiago Gómez Estrada

_____

Jurado
*Ingeniero Electricista*
Sigilfredo Arregocés Ocampo

_____

Director del programa de
Tecnología Eléctrica
*Ingeniero Electricista*
Santiago Gómez Estrada

Pereira, 19 de noviembre de 2014

*“Gracias a Dios, y a ellos porque no hubiera podido ser esto posible, mis padres Héctor, Beatriz por su apoyo y amor incondicional, a mis hermanas Leidy, Ana, Cindy que depositaron su confianza en mí y tienen infinita paciencia, a mi compañera de proyecto por permitirme ingresar en tan valiosa experiencia.*

*Los sueños hechos realidad son la combinación entre persistencia, disciplina y pasión.”*

Héctor Fabio

*“Aun cuando parecía desfallecer siempre encontré en ellos el aliento y los consejos que me dieron la fuerza para continuar en este proceso.*
*Gracias a mis padres, hermanos y toda mi familia que siempre me apoyaron y nunca dejaron de creer en mí. A mi madre porque todo lo que soy y tengo se lo debo a su dedicación, esfuerzo y trabajo.*
*A mi compañero Héctor por su disciplina y perseverancia que hicieron posible finalizar este ambicioso y extraordinario proyecto de grado."*

Ana María

*Queremos dar nuestros más sinceros agradecimientos:*

*Al profesor Jhon Jaime Robby Góez por el acompañamiento académico que nos brindó durante la realización de este proyecto además de sus orientaciones, experiencias compartidas y todos sus conocimientos en esta área que nos guiaron por el camino correcto.*

*Al profesor Santiago Gómez Estrada por guiarnos y apoyarnos académicamente y moralmente durante el desarrollo y finalización del proyecto.*

*Al profesor Edison Duque, por sus orientaciones, y nuevas ideas.*

*Al profesor Sigilfredo Arregocés por sus consejos e ideas compartidas.*

*A nuestros compañeros por su acompañamiento, su apoyo y constante crítica en este largo y arduo camino.*

*A la Universidad Tecnológica de Pereira, Vicerrectoría de investigaciones., innovación y extensión por permitirnos, los fondos para hacer realidad nuestro proyecto, y culminar satisfactoriamente nuestros estudios.*

# Declaración de autoría

Nosotros, Ana María Gil, Héctor Fabio Jiménez, declaramos que esta tesis titulada, "*Comunicación inalámbrica por medio del protocolo ZigBee para la manipulación de un Brazo Robótico*" y el trabajo presentado aquí es nuestro. Confirmamos que:

- Este trabajo se llevó a cabo total o parcialmente, mientras cursamos nuestra carrera, en un ciclo de investigación en esta Universidad.
- Cuando cualquier parte de esta tesis ha sido previamente sometida a un título o cualquier otra titulación en esta universidad o cualquier otra institución, esto ha sido claramente establecido.
- Cuando hemos consultado el trabajo publicado por otros, siempre se cita, y atribuye claramente.
- Dónde hemos citado el trabajo de los demás, la fuente siempre se da. Con la excepción de las citas, esta tesis es enteramente nuestro propio trabajo.
- Hemos reconocido las principales fuentes de ayuda.
- Cuando la tesis se basa en el trabajo realizado por nosotros con ayuda de otros como los asesores de proyecto, hemos dejado claro exactamente lo que se hizo por los demás y lo que hemos contribuido nosotros mismos.

Firmado:

Fecha:

# UNIVERSIDAD TECNOLÓGICA DE PEREIRA

## *Resumen*

Facultad de Tecnología
Programa de Tecnología Eléctrica
Tecnólogo en Electricidad

*Comunicación inalámbrica por medio del protocolo ZigBee para la manipulación de un brazo Robótico*

Por Ana María Gil, Héctor Fabio Jiménez

Los sistemas robóticos programables y multifuncionales diseñados para mover materiales, piezas, herramientas o dispositivos especializados son cada vez más usados, por medio de microcontroladores de gama alta. Haciendo uso de esa tecnología realizamos un prototipo que no solo es de mucha utilidad a nivel industrial sino que representa un desafío a nivel educativo, generando una oportunidad de aprendizaje invaluable.

Este proyecto tiene como objetivo implementar el protocolo de comunicación inalámbrica 802.15.4 mediante el uso de los módulos XBee Pro, XBee serie 1 para la manipulación de un brazo robótico a través de un joystick. La metodología utilizada consiste en el análisis y documentación del protocolo de comunicación, conversión análoga digital de una señal, control de servomotores y por último aplicar técnicas para la formulación de algoritmos mediante las diferentes estructuras de control y su codificación en un lenguaje de alto nivel.

# CONTENIDO

**Pág.**

## LISTA DE FIGURAS

**Pág.**

## LISTA DE TABLAS

**Pág.**

# INTRODUCCIÓN

Desde el inicio y existencia del hombre siempre se ha tratado de hacer cada vez más fáciles las tareas más complejas, es así como el hombre en su propia esencia y la forma abstracta de ver los problemas ha podido crear herramientas rudimentarias y artefactos en el pasado Garrotes, lanzas, arcos, flechas, mazos, picas, son ejemplos de ello; con simples conceptos y el poder de la mente humana, una idea se transforma en un resultado tangible, revolucionario y majestuoso que marcará para siempre el desarrollo de la vida en nuestro planeta.

La Revolución industrial fue un periodo histórico comprendido entre la segunda mitad del siglo XVIII y principios del XIX, en este periodo se registró un proceso de transformación en los métodos de producción, comunicación y transporte. El invento y desarrollo del motor a vapor reemplazo a la energía muscular proveniente del hombre y las fuerzas del agua y el viento, con lo cual el trabajo manual pasó a convertirse en mecánico, es en este punto donde la robótica va unida a la construcción de artefactos, que trataban de materializar el deseo humano de crear seres a su semejanza y que lo descargasen del trabajo diario, el ingeniero español Leonardo Torres Quevedo acuñó el término automática. En relación con la teoría de la automatización de tareas tradicionalmente asociadas.

El término Robot empieza a ser nombrado gracias a la obra R.U.R (*Rossum's Universal Robots*) escrita por Karel Capek en 1920, la palabra checa Robota, que significa trabajo forzado se tradujo a Robot .Un robot es una máquina diseñada para ejecutar una o más tareas, con velocidad y precisión, que puede ser controlada por el hombre y a menudo es considerada como una herramienta, según su aplicación tiene ciertos componentes o elementos que permiten el desempeño de diferentes funciones.

Un robot puede estar conformado por:

- Estructuras mecánicas.
- Sistemas de transmisiones.
- Sistemas de accionamiento.
- Sistemas sensoriales.
- Sistemas de control.
- Elementos de accionamiento final.
- Sistemas inerciales.
- Sistemas de posicionamiento y navegación.

Todo depende de la función que este realizará. Este conjunto de elementos permiten que a través de una lógica programable se lleven a cabo ciertas labores que cada vez se asemejan más a las tareas realizadas por el hombre, transformando la forma de vida y trabajo, además incrementando los límites de la experiencia humana.

En los inicios del nuevo milenio la robótica ha sufrido una gran transformación en su alcance. Esta expansión ha sido provocada por la madurez del campo y los avances en las tecnologías

utilizadas. Desde un enfoque industrial dominando en gran parte, la robótica se ha expandido rápidamente. Se espera que la nueva generación de robots de forma segura y fiable sea compañero de hábitat con los seres humanos en los hogares, lugares de trabajo y comunidades, proporcionando apoyo en los servicios del entretenimiento, la educación, la salud, la fabricación y la asistencia. Más allá de su impacto en los robots físicos, la robótica tiene avances día tras día con una gama mucho más amplia de aplicaciones que llegan a través de diversas áreas de investigación y disciplinas científicas, tales como: biomecánica, óptica, neurociencias, la simulación virtual, educación, animación, cirugías y redes sensoriales, entre otros. Por su parte, los retos de las nuevas áreas emergentes están demostrando ser una abundante fuente de estimulación y puntos de vista para el campo de la robótica.

## DESCRIPCIÓN DEL PROBLEMA

A lo largo del tiempo, los seres humanos han indagado acerca de cómo mejorar cada aspecto o cada detalle que represente una dificultad en las diferentes tareas a realizar. En busca de ello, uno de los mecanismos que ha surgido y que ha traído numerosas ventajas, simboliza un ideal para aquellas personas que desean ser parte de la creación, innovación o vincularse a ese mundo de conocimiento.

El robot, como herramienta en cualquier campo, ha significado un progreso amplio en la sociedad y a medida que el tiempo transcurre se desarrollan más máquinas con mayor eficiencia y de mejor calidad. Por ende, no estamos lejos de ver como el ingenio del hombre le dará cualidades admirables, que van desde su autonomía hasta una inteligencia artificial.

En la actualidad se han creado diferentes tipos de robots buscando reemplazar actividades cotidianas, rutinarias y peligrosas para la integridad humana. En la industria, se ha implementado el uso de muchos tipos de robots, entre ellos los brazos robóticos son muy comunes, sus tareas son diversas, y van desde la toma de decisiones, hasta la manipulación y transporte de materiales en todo tipo de procesos, en especial en los más rutinarios y peligrosos para el ser humano.

A nivel comercial, también se encuentran robots que tienen interfaces amigables hombre máquina (*HMI*), en el comercio se difunde cada vez más el uso de electrodomésticos y accesorios robotizados para el hogar, como lavadoras, brilladoras, y para el transporte vehículos con tecnologías e interfaces robotizadas.

En el campo de la medicina, los robots son utilizados en procedimientos de cirugía invasiva mínima, transporte de muestras biológicas o químicas entre instrumentos tales como incubadoras, manejadores de líquidos y lectores.

En el campo militar, se usan robots para espionaje, ataque, eliminación, destrucción de fuerzas hostiles.

También se usan, en la limpieza de residuos tóxicos, la minería, la búsqueda y rescate de personas, la localización de minas terrestres, la exploración del fondo oceánico y la exploración espacial, entre muchas otras aplicaciones [1].

En la robótica cada diseño o estructura es dependiente de la aplicación a realizar, por medio de la agrupación de diversos dispositivos previamente desarrollados y haciendo uso de teorías, programas y conclusiones de diseños presentes en invenciones o elementos que ya fueron creados.

Con ayuda de estos avances, el proyecto desarrollado Comunicación Inalámbrica Por Medio Del Protocolo ZigBee Para La Manipulación De Un Brazo Robótico, el cual tendrá nombre clave "**XBee-Hand**", será un proyecto multifuncional dependiente de la programación,

configuración o aplicación que se le dé, ya que se puede introducir dentro del sector educativo, industrial, comercial y residencial.

Sus características principales son:

- Brazo robótico
- Comunicación inalámbrica
- Control a distancia
- Sistema de control programable

En síntesis, este dispositivo tendrá la capacidad de realizar procesos de manipulación de objetos de una posición a otra, a través de un joystick, el cual puede ser manipulado desde una distancia máxima de 60m inalámbricamente con la destreza suficiente para sujetar un objeto sin dañarlo.
No es difícil ver que desarrollar este proyecto representa para nosotros como estudiantes del programa de tecnología eléctrica una oportunidad muy valiosa para ampliar los conocimientos, conceptos y fundamentos adquiridos, además aplicar y desarrollar habilidades en desarrollo y construcción de circuitos, dispositivos y mecanismos robotizados con control electrónico (*know- how*).

También se ampliarán y afinarán los conocimientos en desarrollo de algoritmos, depuración, simulación, de dispositivos de alta escala de integración, como lo son los microcontroladores.

Por otro lado, se reforzarán, ampliarán y adquirirán nuevos conocimientos en otras áreas tales como: mecanismos, servomotores, comunicaciones y control.
Este sistema se diseñará como un sistema abierto, que puede usarse como una herramienta de aprendizaje de la robótica y de la electrónica aplicada, con la posibilidad de expandirse y mejorarse continuamente.

Finalmente, este proyecto trae como beneficio para la región o el país, que se creará una herramienta para el aprendizaje en el campo de la robótica; además, será un instrumento programable que se adaptará a las necesidades o un problema en específico dentro del campo industrial, comercial, militar, medico entre otros, teniendo en cuenta sus características de manipulación de objetos y manejo del mismo a través de medios inalámbricos con un óptimo uso de energía ,permitiendo el comando a distancia.

# OBJETIVOS

## OBJETIVO GENERAL

Implementar el protocolo de comunicación inalámbrica 802.15.4 mediante el uso de los módulos XBee Pro, XBee serie 1 para la manipulación de un brazo robótico.

## OBJETIVOS ESPECÍFICOS

- Diseñar el control del brazo robótico por medio de un joystick.
- Desarrollar el sistema de control del brazo robótico.
- Implementar el sistema de comunicación inalámbrica por medio del protocolo 802.15.4 *ZigBee*.
- Ensamblar el brazo robótico, el sistema de control y el sistema de comunicaciones.
- Utilizar los recursos del microcontrolador PIC18F4550 para la parte lógica, y solución de problemas del proyecto.
- Identificar y utilizar técnicas para la formulación de algoritmos mediante las diferentes estructuras de control y su codificación en un lenguaje de alto nivel.

# 1. ELEMENTOS EMPLEADOS

## 1.1 COMUNICACIÓN INALÁMBRICA

### 1.1.1 Definición

La comunicación inalámbrica o sin cables es aquella en la que los extremos de la comunicación (emisor/receptor) no se encuentran unidos por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio. Este tipo de comunicación facilita la operación en lugares donde los dispositivos no se encuentra en una ubicación fija (almacenes, oficinas de varios pisos, etc.) actualmente se utiliza de una manera general y accesible para todo público.

La comunicación inalámbrica se encuentra en un desarrollo creciente y masivo, cada vez es mayor la necesidad de conectar equipos e implementar este tipo de transmisión en sistemas electrónicos y sistemas embebidos, en donde los microcontroladores tienen un rol importante para realizar tareas específicas.

### 1.1.2 Módulos XBee

Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto) [2] . Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. XBee es propiedad de Digi International una empresa líder a nivel mundial en el desarrollo de módems de conexión a redes inalámbricos para dispositivos electrónicos basado en el protocolo ZigBee.

**Figura 1:** Módulos XBee


Tomado de [3]

Presentan una solución excepcionalmente potente para los numerosos mercados que adoptan la conexión a redes inalámbricas para sus aplicaciones de comunicaciones de datos. La línea de productos XBee se puede encontrar en diversas aplicaciones industriales y comerciales, como sensores remotos, control y manipulación de robots, control de equipos y

automatizaciones. Si bien existen bastantes módulos inalámbricos, estos son los que mantienen la relación exacta entre precio y calidad, y debido a su pequeño tamaño y fácil programación (solo requiere una conexión serial) son ideales para cualquier proyecto.
En términos simples, los XBee son módulos inalámbricos fáciles de usar e implementar a gran escala.

### 1.1.3 Características del módulo XBee

Los módulos XBee son módulos de radio frecuencia que trabajan en la banda de 2.4 GHz con protocolo de comunicación IEEE 802.15.4 fabricados por MaxStream.

Los módulos tienen 6 convertidores análogo-digital y 8 entradas digitales además de Rx y Tx. [3] Trabajan a 2.4 GHz y generan una red propia a la que puedes conectarte o desconectarte. Entre otras características a tener en cuenta hay que decir que son módulos microprocesador con lo cual tienes solucionados los problemas de fallo de trama, ruidos, entre otros. Los módulos, se comunican con un dispositivo RS232 a niveles TTL, ofrecen una velocidad de comunicación desde 1200 hasta 115.200 baudios pasando por todos los valores convencionales, también disponen de varias I/O que pueden ser configuradas para diferentes funciones [3]. Los módulos XBee pueden ser programados a través de una hyperterminal y una interface serial con un **MAX3232** y una serie de **comandos llamados AT**. El fabricante de los módulos también facilita al usuario un software de programación llamado **X-CTU,** en la Tabla 1 se puede observar el diagrama de pines para este módulo**.**

**Tabla 1:** Diagrama de pines del módulo XBee

| Pin # | Name(s) | Description |
|---|---|---|
| 1 | VCC | 3.3 V power supply |
| 2 | DOUT | Data Out (TX) |
| 3 | DIN | Data In (RX) |
| 4 | DIO12 | Digital I/O 12 |
| 5 | RESET | Module reset (asserted low by bringing pin to ground) |
| 6 | PWM0/RSSI/DIO10 | Pulse-width modulation analog output 0, Received Signal Strength Indicator, Digital I/O 10 |
| 7 | DIO11 | Digital I/O 11 |
| 8 | Reserved | Do not connect |
| 9 | DTR/SLEEP_RQ/ DIO8 | Data Terminal Ready (hardware handshaking signal), Pin Sleep Control (asserted low), Digital I/O 8 |
| 10 | GND | Ground |
| 11 | DIO4 | Digital I/O 4 |
| 12 | CTS/DIO7 | Clear to Send (hardware handshaking), Digital I/O 7 |
| 13 | ON/SLEEP | Sleep indicator (off when module is sleeping) |
| 14 | VREF | Not used in Series 2 |
| 15 | ASSOC/DIO5 | Association indicator: blinks if module is associated with a network, steady if not; Digital I/O 5 |
| 16 | RTS/DIO6 | Request to Send (hardware handshaking), Digital I/O 6 |
| 17 | AD3/DIO3 | Analog Input 3, Digital I/O 3 |
| 18 | AD2/DIO2 | Analog Input 2, Digital I/O 2 |
| 19 | AD1/DIO1 | Analog Input 1, Digital I/O 1 |
| 20 | AD0/DIO0/COMMIS | Analog Input 0, Digital I/O 0, Commissioning Button |

Tomado de [3].

Estos módulos son utilizados en:

- Automatización de casas
- Sistemas de seguridad
- Monitoreo de sistemas remotos
- Aparatos domésticos
- Alarmas contra incendio
- Plantas tratadoras de agua.

MaxStream fabrica más de 70 tipos de módulos XBee con diferentes antenas, potencia y capacidades, pero básicamente hay dos series mayormente usadas de los módulos XBee:

### 1.1.4 XBee Series 1

Son la serie más fácil para trabajar, no necesitan ser configurados, pero incluso así se pueden obtener beneficios. Para comunicaciones Punto-a-Punto, estos módulos trabajan tan bien como los de la Serie 2, pero sin todo el trabajo de pre configuración previa. Utilizan un microchip hecho por Freescale para proporcionar las comunicaciones basadas en estándares de una implementación propietaria de las redes de *mesh* o malla [2].

### 1.1.5 XBee Series 2

La Serie 2 utiliza un microchip de Ember Networks que permite varios protocolos estándares basados en ZigBee. La creación de redes Mesh es el corazón de la creación de redes de sensores robustos y esta serie permite monitorear una gran tasa de datos. Esta seria por ser la segunda tiene algunas mejoras como un mejor alcance y potencia de transmisión, además de eso está soporta completamente todas las características del protocolo ZigBee. La siguiente imagen muestra un resumen de similitudes y diferencies entre ambas series [2].

**Tabla 2:** Diferencias y similitudes entre XBee Serie 1 y Serie 2

| Characteristic | Series 1 | Series 2 |
|---|---|---|
| Typical (indoor/urban) range | 30 meters | 40 meters |
| Best (line of sight) range | 100 meters | 120 meters |
| Transmit/Receive current | 45/50 mA | 40/40 mA |
| Firmware (typical) | 802.15.4 point-to-point | ZB ZigBee mesh |
| Digital input/output pins | 8 (plus 1 input-only) | 11 |
| Analog input pins | 7 | 4 |
| Analog (PWM) output pins | 2 | None |
| Low power, low bandwidth, low cost, addressable, standardized, small, popular | Yes | Yes |
| Interoperable mesh routing, ad hoc network creation, self-healing networks | No | Yes |
| Point-to-point, star topologies | Yes | Yes |
| Mesh, cluster tree topologies | No | Yes |
| Single firmware for all modes | Yes | No |
| Requires coordinator node | No | Yes |
| Point-to-point configuration | Simple | More involved |

| Characteristic | Series 1 | Series 2 |
|---|---|---|
| Standards-based networking | Yes | Yes |
| Standards-based applications | No | Yes |
| Underlying chipset | Freescale | Ember |
| Firmware available | 802.15.4 (IEEE standard), DigiMesh (proprietary) | ZB (ZigBee 2007), ZNet 2.5 (obsolete) |
| Up-to-date and actively supported | Yes | Yes |

Tomado de [3]

### 1.1.6 Tipos de Antena

Existe una gran diversidad de tipos de antenas. En unos casos deben expandir en lo posible la potencia radiada, es decir, no deben ser directivas (ejemplo: una emisora de radio comercial o una estación base de teléfonos móviles), otras veces deben serlo para canalizar la potencia en una dirección y no interferir a otros servicios (antenas entre estaciones de radioenlaces). Al ser estos módulos dispositivos de radio necesitan antenas para obtener la mejor calidad en la recepción y transmisión de datos, mostraremos los tipos de antenas que hay a continuación.

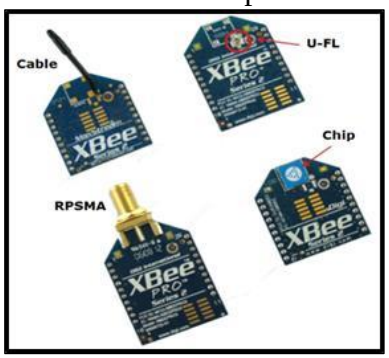
- **Whip o Cableada:** Como su nombre lo indica es un conductor que está atado al cuerpo del módulo XBee, esta ofrece una radiación omnidireccional, entiendo esto como la transmisión de potencia máxima será en los 360 grados.

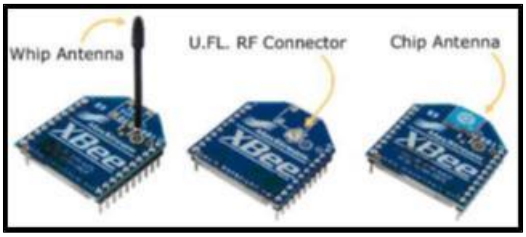
- **Antena de Chip:** Esto es más o menos lo que parece. La antena de chip es una cerámica plana integrado por un chip que esta al ras con el cuerpo del XBee. Eso hace que sea más pequeño y más robusto, pero estas ventajas tienen un precio. Esta antena tienen un cardioide como patrón de radiación, esto es en forma de corazón, lo que significa que la señal se atenúa en muchas direcciones.

- **Conector U.FL:** Este tipo de conector en el módulo da la posibilidad de conectar una antena externa. Se utiliza cuando el módulo estar dentro de compartimiento metálico.

- **Conector RPSMA:** El conector RPSMA es más grande y más voluminoso, se puede utilizar con una antena externa, montada directamente al XBee sin un cable de conexión. [3]

**Figura 2:** Diferentes tipos de antenas de los módulos XBee




Tomado de [3]

### 1.1.7 Topologías de Red

Una red de módulos XBee se encuentra conformado por dos o más nodos, en toda red ZigBee existen tres tipos de roles según el papel que tengan que desempeñar dentro de esta ellos son:

- **Coordinador:** Este nodo se encarga de la principal tarea dentro de una red ZigBee como es la de formar la red entre nodos, establecimiento de comunicación entre toda la red, establecer el identificador de red (*PAN ID*), encargado de la administración y gestión de la red, enrutador de paquetes, por cada red ZigBee solo abra un coordinador que permitirá o denegara la inclusión de otros nodos.

- **Router:** El rol de un Router es mantener y crear información sobre la red para determinar la mejor ruta para transmitir la información, como es de esperarse este debe unirse a una red ZigBee antes de poder actuar como router retransmitiendo paquetes a otros routers o dispositivos finales, típicamente esto suelen conectarse a una alimentación eléctrica continua y de alta duración ya que estos deben estar encendido la mayor parte del tiempo utilizando al menos el 70% de la alimentación eléctrica de un nodo coordinador, dentro de una red de este tipo puede existir varios routers.

- **Dispositivos Finales ó End Device:** Los nodos actuando teniendo este rol no tienen la capacidad de enrutar paquetes, tienen la capacidad de unirse a las redes, enviar y recibir información, pero eso es todo, tienen la posibilidad de entrar en modo sleep o intermitencia permitiendo un ahorro de consumo energético al no tener que realizar funciones de enrutamiento. La desventaja de estos es que siempre requieren de un nodo padre como un router o coordinador. El nodo padre ayuda a estos en la conexión con otros segmentos de la red y a transmitir sus mensajes.

En teoría la cantidad máxima de nodos que puede existir en una red ZigBee ya que cada nodo o elemento de la red tendrá asignado una dirección de 16 bits. Es:

$$2^{16} = 65536$$

En este tipo de redes la aplicabilidad de la teoría de grafos es muy utilizada por el tipo de conexiones mesh que se realizan, pues para enrutar los paquetes se deben encontrar caminos cortos, siendo algunos grafos conexos o no conexos estando aquí la aplicabilidad.

El conjunto de los grafos completos es denominado usualmente $\mathbb{K}$ , siendo $\mathbb{K}_n$ el grafo completo de *n* vértices.

$$\mathbb{K}_2 = \frac{n(n-1)}{2} \quad (1)$$

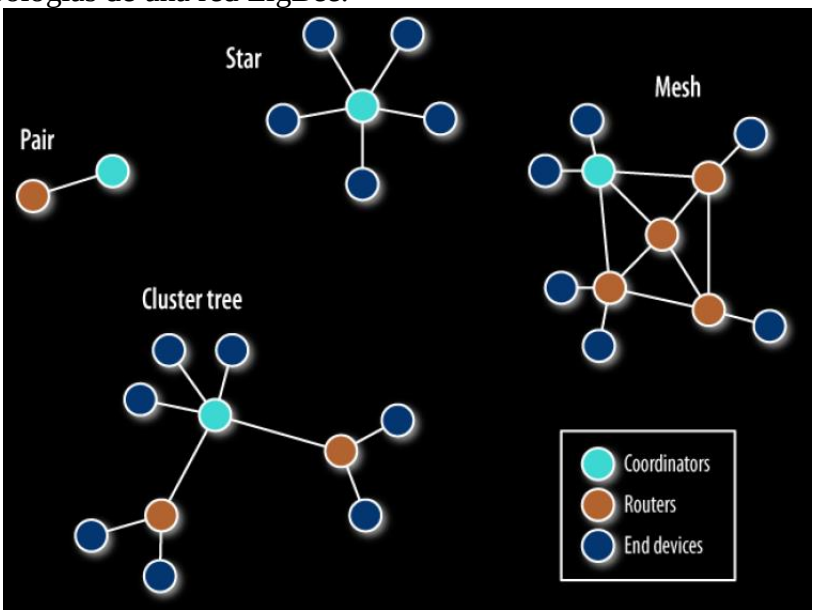
Para el caso de existir 4 nodos tendremos una cantidad máxima posible:

$$\mathbb{K}_n = \frac{4(4-1)}{2} \quad (2)$$

$$\mathbb{K}_n = 6 \quad (3)$$

Habiendo descrito los roles dentro de una red ZigBee soporta diferentes configuraciones de red como son los mostrados de la Figura 3.

**Figura 3:** Topologías de una red ZigBee.




Tomada de [3].

- **Punto a Punto *(Pair Network)*:**

Esta es la configuración más simple que consta de dos módulos, donde uno de los dos debe ser el coordinador y el otro puede ser un dispositivo final o router cualquier dispositivo puede comunicarse con el otro siempre y cuando poseen los mismos parámetros establecida dos. Se utiliza este tipo de redes cuando se requiere confiabilidad de enrutamiento.

- **Estrella (Star Network):**

En esta configuración uno de los dispositivos asume el rol de coordinador de la red y es responsable de inicializar todos los dispositivos en la red, el resto de los demás se comportan como dispositivos finales, cada mensaje enviado debe pasar a través del nodo coordinador el cual enruta la información al correspondiente dispositivo, en esta configuración los dispositivos finales no se comunican entre sí.

- **Árbol (Cluster tree Network):**

Esta topología es un caso especial pues el rol de coordinador puede ser precedido a otros routers y proveen servicios de sincronización hacia otros dispositivos o coordinadores.

- **Malla (Mesh Network):**

En esta configuración el nodo coordinador es responsable de inicializar la red y elegir los parámetros de la red, gran parte de su tiempo se encargara de gestionar y administrar la red a su vez la red puede ser ampliada a través del uso de routers para alcanzar los dispositivos finales, en donde el algoritmo de enrutamiento pregunta y responde para eliminar las rutas que no sean optimas basadas en la latencias y tiempos de respuesta.

### 1.1.8 Modos de Operación

Todo módulo XBee posee al menos 5 modos de operación o 5 modos de trabajo estos son:

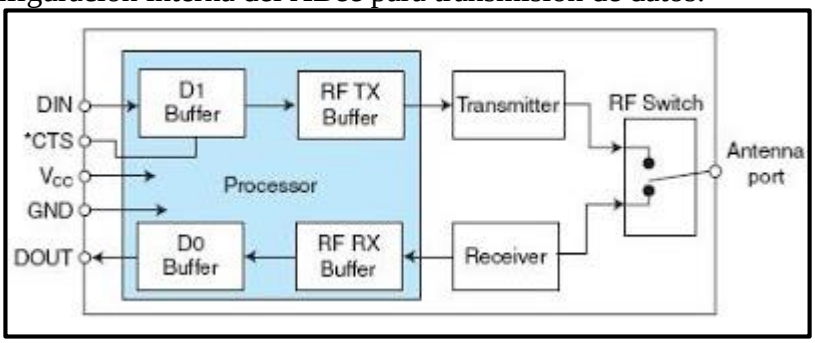
#### 1.1.8.1 Recepción

Este modo de operación funciona cuando un paquete de radio frecuencia es recibido y es válido utilizando la misma dirección de parámetro MY de 16 bits, así que todo paquete RF recibido en DIN, será guardado en un buffer DOUT para ser enviado al dispositivo con conexión serial RS-232.

#### 1.1.8.2 Transmisión

Cuando un dispositivo externo envía un dato a través de la conexión serial (*e.g Microcontrolador (Tx) y XBee (Rx))* estos son recibidos en un buffer del pin 3 del XBee como se muestra en la Figura 4.

**Figura 4:** Configuración interna del XBee para transmisión de datos.




Tomado de [4].

Inmediatamente el módulo saldrá del modo IDLE e intentara transmitir los datos utilizando la configuración escrita en su firmware como es la dirección de destino, que determina cual será el nodo receptor.

Antes de realizar él envió de los datos, el XBee establece una conexión con el nodo receptor, si esto llegara a fallar el paquete será desechado. Cuando los datos son transmitidos de un nodo a otro, un reconocimiento de nivel de red es transmitido atrás a través de la ruta establecida al nodo de la fuente. Este paquete de reconocimiento (también conocido como *acknowledgement*) indica al nodo de la fuente que el paquete de datos fue recibido exitosamente por el nodo destino. Si un reconocimiento de red no es recibido, el nodo de la fuente transmitirá de nuevo los datos tres veces, si no es recibido después de los tres intentos un ACK de fallo se registra.

#### 1.1.8.3 Bajo Consumo

Uno de los modos más interesantes en estos dispositivos XBee es su modo de bajo consumo (*sleep mode*) pues en muchos proyectos en los cuales se involucra el uso de dispositivos móviles y transmisión RF (*Radio Frequency*) se utiliza comúnmente las baterías o alguna otra fuente de alimentación, al ser una fuente de capacidad finita se debe tratar de optimizar al máximo el uso de estos recursos.

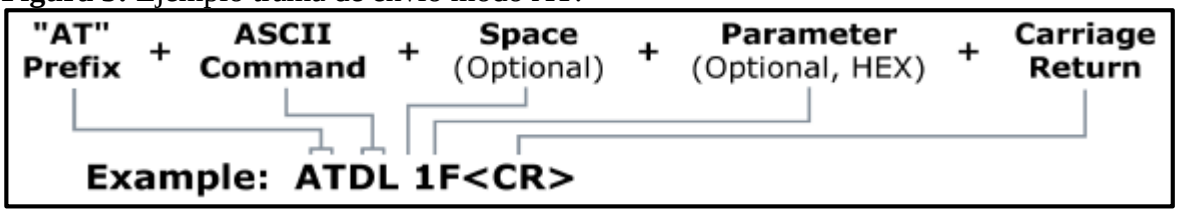
Los dispositivos XBee cuentan con este modo que permite un gran ahorro de energía, pues cuando estos se encuentran temporalmente dormidos en un estado de hibernación o reposo como el *sleep* no ejecutan ninguna tarea de transmisión o recepción de datos , hasta que se recibe una señal de activación conocida como *Wake up.*

Algunas de las desventajas que presenta este modo de bajo consumo para los dispositivos finales es que como ellos se despiertan solo hasta que el nodo coordinador o nodo padre envía la señal de activación, de lo contrario los nodos padres (routers, coordinadores) deberán almacenar los mensajes que llegan al dispositivo final, la desventaja es que estos solo almacenan un único mensaje.

### 1.1.8.4 Comando

Este modo permite ingresar comandos AT al módulo XBee, para configurar, ajustar o modificar parámetros. Permite ajustar parámetros como la dirección propia o la de destino, así como su modo de operación entre otras cosas. Un microcontrolador que maneje UART, USART y tenga los comandos guardados en memoria o los adquiera de alguna u otra forma. Para ingresar a este modo se debe esperar un tiempo dado por el comando GT (Guard Time, por defecto ATGT=0x3E8 que equivalen a 1000ms) luego ingresar +++ y luego esperar otro tiempo GT. Como respuesta el módulo entregará un OK. El módulo XBee viene por defecto con una velocidad de 9600bps. En caso de no poder ingresar al modo de comandos, es posible que sea debido a la diferencia de velocidades entre el módulo y la interfaz que se comunica vía serial, en la Figura 5 se muestra lo que comprende un trama de envío en modo AT.

**Figura 5:** Ejemplo trama de envío modo AT.




Tomado de [5].

Un código de ejemplo para el compilador c18 v3.37 para ingresar en este modo se muestra en el capítulo 3 titulo 3.3.

## 1.2 SERVOMOTORES

### 1.2.1 Definición

Un servomotor (o servo) es un motor de corriente continua que tiene la capacidad de ser controlado en posición. Es capaz de ubicarse en cualquier posición dentro de un rango de operación (generalmente de 180º) y mantenerse estable en dicha posición. Los servos se suelen utilizar en robótica, automática y modelismo (vehículos por radio-control, RC) debido a su gran precisión en el posicionamiento [6] .

En general, los servos suelen estar compuestos por 4 elementos fundamentales:

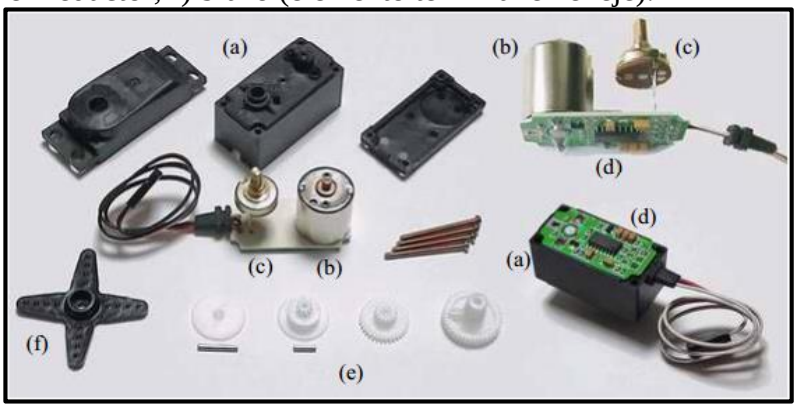
- **Motor de corriente continua (DC):** Es el elemento que le brinda movilidad al servo. Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado sus dos terminales es inverso, el sentido de giro también se invierte.

- **Engranajes reductores:** Tren de engranajes que se encarga de reducir la alta velocidad de giro del motor para acrecentar su capacidad de torque (o par-motor).

- **Sensor de desplazamiento:** Suele ser un potenciómetro colocado en el eje de salida del servo que se utiliza para conocer la posición angular del motor.

- **Circuito de control:** Es una placa electrónica que implementa una estrategia de control de la posición por realimentación. Para ello, este circuito compara la señal de entrada de referencia (posición deseada) con la posición actual medida por el potenciómetro. La diferencia entre la posición actual y la deseada es amplificada y utilizada para mover el motor en la dirección necesaria para reducir el error [6].

**Figura 6:** Componentes de un servo: a) carcasa; b) motor DC; c) potenciómetro; d) circuito de control; e) tren reductor; f) brazo (elemento terminal en el eje).

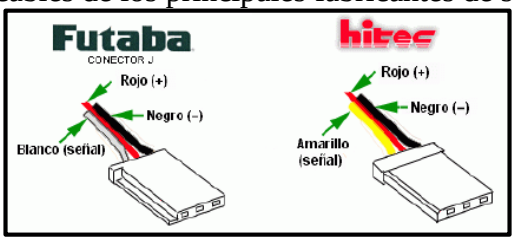



Tomado de [6].

### 1.2.2 Principios de funcionamiento

Los servos disponen de tres cables Figura 7: dos cables de alimentación (positivo y negativo/masa) que suministran un voltaje 4.8-6V y un cable de control que indica la posición deseada al circuito de control mediante señales PWM ("*Pulse Width Modulation*").

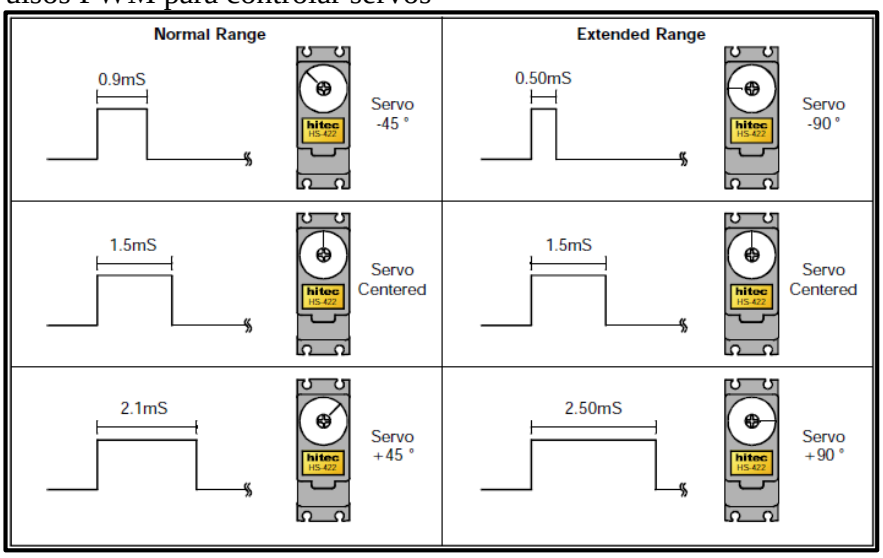
**Figura 7:** Colores de los cables de los principales fabricantes de servos.




Tomada de [6].

Las señales PWM utilizadas para controlar los servos están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20ms (50Hz). Todos los servos pueden funcionar correctamente en un rango de movimiento de 90º, que se corresponde con pulsos PWM comprendidos entre 0.9 y 2.1ms. Sin embargo, también existen servos que se pueden mover en un rango extendido de 180º y sus pulsos de control varían entre 0.5 y 2.5ms (Figura 8). Antes de utilizar un servo habrá que comprobar experimentalmente su rango de movimiento para no dañarlo. Para mantener fijo un servo en una posición habrá que enviar periódicamente el pulso correspondiente; ya que si no recibe señales, el eje del servo quedará libre y se podrá mover ejerciendo una leve presión [6].

**Figura 8:** Pulsos PWM para controlar servos




Tomada de [6].

### 1.2.3 Tipologías

Existen dos tipos de servos: analógicos y digitales. Ambos tipos de servos son iguales a nivel de usuario: tienen la misma estructura (motor DC, engranajes reductores, potenciómetro y placa de control) y se controlan con las mismas señales PWM. La principal diferencia entre ellos radica en la adición de un microprocesador en el circuito de control de los servos digitales. Este microprocesador se encarga de procesar la señal PWM de entrada y de controlar el motor mediante pulsos con una frecuencia 10 veces superior a los servos analógicos [6].

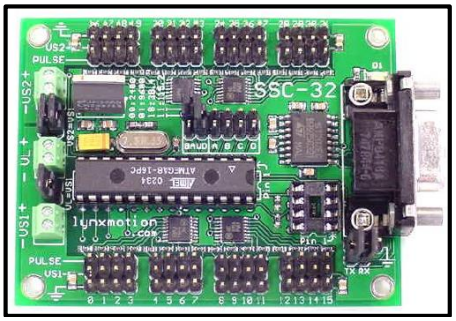
El aumento en la frecuencia de excitación del motor en los servos digitales permite disminuir su tiempo de respuesta (menor deadband), aumentar su resolución de movimiento y suavizar su aceleración/deceleración. El uso de un microprocesador permite también a los servos digitales programar distintos parámetros de configuración que son fijos en los analógicos: sentido de giro, posición central inicial, topes en el recorrido del servo, velocidad de respuesta del servo y resolución. Para establecer estos parámetros se deben utilizar aparatos específicos de cada marca. El principal inconveniente de los servos digitales es que consumen más energía que los analógicos al tener que generar más pulsos de control para el motor [6].

## 1.3 TARJETA DE CONTROL DE SERVOMOTORES SSC-32

Un SSC (*Serial Servo Controller*, Controlador Serie de Servos) es un dispositivo utilizado para controlar servos desde un PC a través del puerto serie. Los SSC aceptan comandos con un determinado formato desde el puerto serie del PC y los transforman en pulsos PWM que son enviados a los servos que se desea controlar [6].

La tarjeta SSC-32 (Figura 9) es un controlador de la empresa *Lynxmotion* que permite controlar hasta un máximo de 32 servos. Se trata de un controlador más completo que el Mini SSC ya que dispone de un conjunto de funcionalidades adicionales: control de servos por tiempo/velocidad/posición, movimiento síncrono de varios servos, consulta de posición de los servos y utilización de los pines de control de los servos como salidas digitales TTL. Además, dispone de 4 entradas (A, B, C y D) que pueden ser leídas de manera digital (bits) o de manera analógica (tensiones) [6].

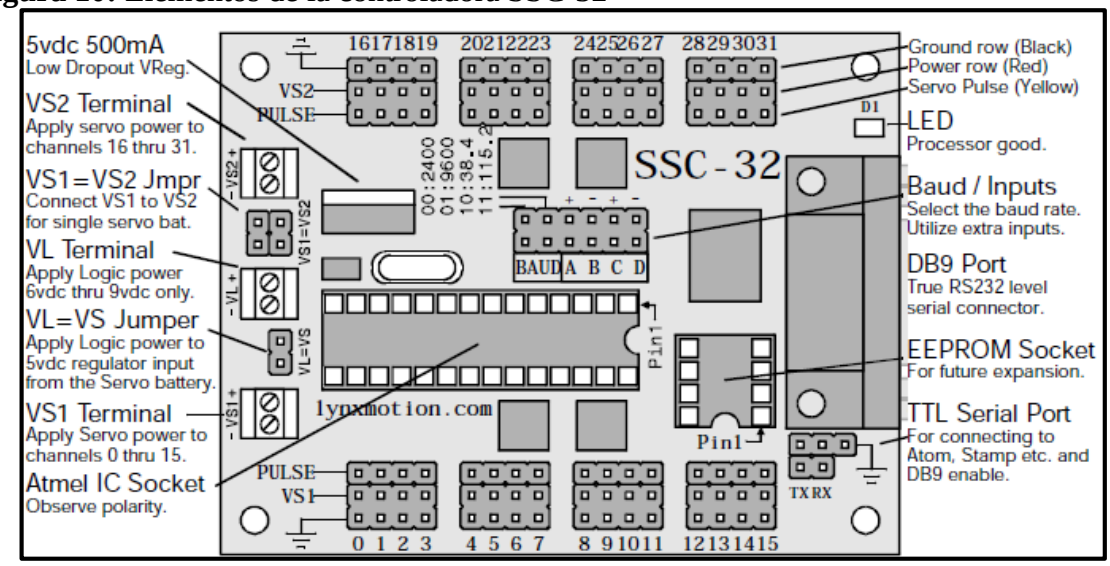
**Figura 9:** Controlador SSC-32




Tomado de [7].

Este controlador presenta un gran número de ventajas respecto al Mini SSC: control más complejo de los servos, disponibilidad de 4 entradas, diversas configuraciones de alimentación (fuente única para placa y servos o fuentes separadas), mayor rango de velocidades del puerto serie (2400bps, 9600bps, 38.4 kbps y 115.2kbps), posibilidad de conectar el SSC-32 a un micro-controlador a través de comunicación serie TTL. Su único inconveniente es su mayor tamaño al disponer de un conector DB9 estándar ver Figura 10.

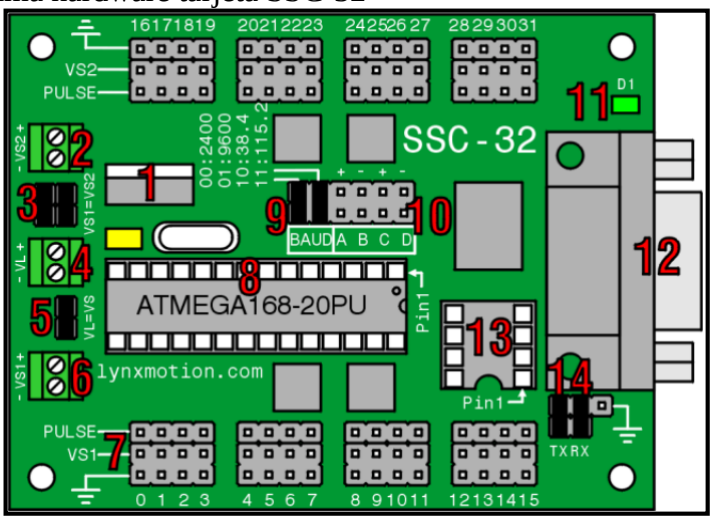
**Figura 10:** Elementos de la controladora SSC-32




Tomado de [7].

### 1.3.1 Información hardware tarjeta SSC-32

**Figura 11:** Diagrama hardware tarjeta SSC-32




Tomado de [8].

1. El regulador de tensión de baja caída (LDO) proporcionará una salida de 5 voltios CC con una entrada de tan sólo 5,5 voltios CC. Este dato es importante al alimentar el robot a través de una batería. Admite una entrada máxima de 9 voltios CC. El regulador tiene una potencia nominal de 500 mA, aunque se debe intentar reducir a 250 mA para evitar que el regulador se sobrecaliente en exceso [8].

2. Este terminal es la alimentación de los canales de los servos (del 16 al 31). Se deben aplicar de 4,8 a 6 voltios CC para los servos analógicos y digitales. Se pueden obtener directamente de un pack de pilas de 5 pilas NiMH. Los servos HSR-5980 o HSR-5990 se pueden alimentar con 7,2 Vdc - 7,4 Vdc. Se pueden obtener directamente de un pack de 2 pilas de litio o un pack de 2 pilas de NiMH [8], ver Tabla 3.

**Tabla 3:** Alimentación canales de los servos (16-31) tarjeta SSC-32

| Placa | Entrada |
|---|---|
| VS2+ | ROJO |
| VS1 - | NEGRO |

3. Estos jumpers se utilizan para conectar VS1 a VS2. Utilice esta opción cuando vaya a alimentar todos los servos desde la misma batería. Use ambos jumpers. Si se desea utilizar dos packs de pilas independientes, uno en cada lado; elimine estos dos jumpers [8].

4. Esta es la tensión lógica o VL Es la alimentación para la electrónica del circuito. Esta entrada se utiliza normalmente con un conector de batería de 9 voltios CC para alimentar los ICs y cualquier cosa que esté conectada a las líneas de 5 voltios CC de la placa. El rango válido para este terminal es 6 Vdc - 9 Vdc. Esta entrada se utiliza para aislar la alimentación de la lógica de la alimentación de los servos. Es necesario eliminar el jumper VS1=VL si los servos se van a alimentar de forma independiente desde los conectores VS. El circuito SSC-32 debería consumir 35 mA sin tener nada conectado a la salida de 5 Vdc [8], ver Tabla 4.

**Tabla 4:** Alimentación circuitos electrónicos tarjeta SSC-32

| Placa | Entrada |
|---|---|
| VL+ | ROJO |
| VL - | NEGRO |

5. Este jumper permite alimentar el Microcontrolador y los servos desde la misma alimentación del conector VL. Se requieren al menos 6 voltios CC para que funcione correctamente. Si el Microcontrolador se resetea cuando haya varios servos en movimiento, entonces es recomendable alimentar el Microcontrolador de forma independiente a través de la entrada VL. Una batería de 9 voltios CC es perfectamente adecuada para ello. Este jumper debería eliminarse si se va a alimentar el Microcontrolador de forma independiente [8].

**6.** Este terminal es la alimentación de los canales de los servos (del 0 al 15). Se deben aplicar de 4,8 a 6 voltios CC para los servos analógicos y digitales. Se pueden obtener directamente de un pack de pilas de 5 pilas NiMH. Los servos HSR-5980 o HSR-5990 se pueden alimentar con 7,2 Vdc - 7,4 Vdc. Se pueden obtener directamente de un pack de 2 pilas de litio o un pack de 2 pilas de NiMH [8], ver Tabla 5.

**Tabla 5:** Alimentación canales de los servos (0-15) tarjeta SSC-32

| Placa | Entrada |
|---|---|
| VL1+ | ROJO |
| VL1 - | NEGRO |

**7.** Aquí es donde se tiene que conectar los servos y los demás dispositivos de salida. Es necesario, desactivar la alimentación al conectar cualquier elemento al bus de E/S [8] ver Tabla 6.

**Tabla 6:** Conexión servos a la tarjeta SSC-32

| Placa | Cable |
|---|---|
| Pulso | Amarillo ó Blanco |
| VS | Rojo |
| Tierra | Negro ó Marrón |

**8.** Aquí es donde se encuentra el chip IC de Atmel. Se Debe insertar con cuidado el Pin 1 con la esquina superior derecha como se indica en la Figura 11. Evitar doblar los pines [8].

**9.** Las dos entradas de BAUDIOS (BAUD) permiten configurar la tasa de baudios. [8] Ver las configuraciones aceptables en la tarjeta relacionada en la Tabla 7.

**Tabla 7:** Configuración velocidad de transmisión de datos tarjeta SSC-32

| Jumper | Tasa de baudios | Uso |
|---|---|---|
| 0 0 | 2400 | Procesadores de menor velocidad |
| 0 1 | 9600 | Procesadores de menor velocidad |
| 1 0 | 38,4k | Comunicación Atom/Stamp |
| 1 1 | 115,2k | Comunicación con el PC, Actualización de firmware |

**10.** Las entradas ABCD tienen soporte estático y biestable. Las entradas tienen resistencias internas de tipo pull up (50k) débiles que se utilizan con los comandos de entrada digital de lectura. Es recomendable utilizar un interruptor normalmente abierto desde la entrada a tierra [8].

**11.**Este es el LED indicador del buen estado del procesador. Se iluminará de forma continua cuando se aplique la alimentación y permanecerá iluminado hasta que el procesador haya

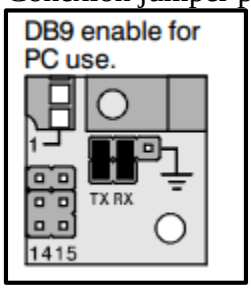
recibido un comando serie válido. Se apagará y volverá a parpadear siempre que reciba datos serie [8].

**12.**Simplemente debe conectar un cable con conector DB9 Macho/Hembra desde este conector a un puerto serie libre de 9 pines de su ordenador para recibir los datos de posicionamiento de los servos. También se puede utilizar un adaptador de USB a puerto serie. Tener en cuenta que existen numerosos adaptadores USB-SERIE que requieren una alimentación independiente para funcionar correctamente [8].

**13.**Se trata de un zócalo para una EEPROM de 8 pines. La EEPROM es compatible con el firmware 2.01GP [8].

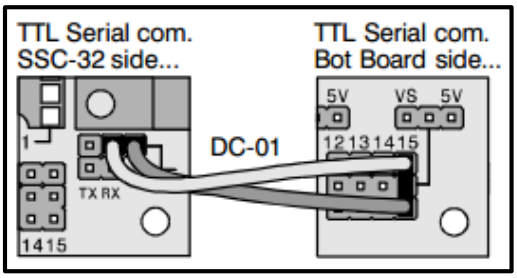
**14.** Este es el puerto serie a nivel TTL. Instalar dos Jumper como se muestra en la Figura 12 para habilitar el puerto DB9. Instalar dos conectores para utilizar la comunicación serie a nivel TTL Figura 13 desde un Microcontrolador ver Figura 13 [8].

**Figura 12:** Conexión jumper para DB9




Tomada de [8].

**Figura 13:** Conexión de jumper para nivel TTL.




Tomada de [8].

## 1.4 MICROCONTROLADOR

### 1.4.1 Definición

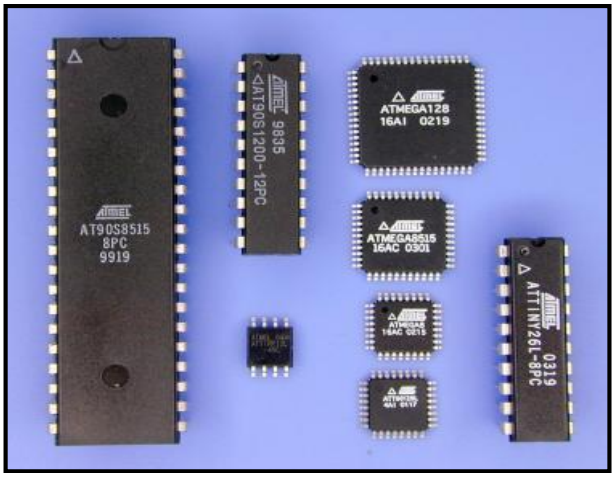
Un micro controlador es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida [9].

Son diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación.

Los micro controladores representan la inmensa mayoría de los chips de computadoras vendidos, sobre un 50 % son controladores “simples” y el restante corresponde a DSP’s más especializados. Pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

Un micro controlador difiere de una unidad central de procesamiento normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de circuitos integrados externos de apoyo. La idea es que el circuito integrado se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips. Hay que agregarle los módulos de entrada y salida (puertos) y la memoria para almacenamiento de información. [9]

**Figura 14:** Micro controlador




Tomada de [9].

### 1.4.2 Arquitectura Interna Microcontroladores

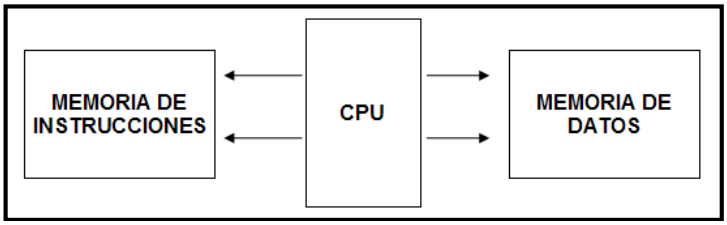
La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU está conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar [10]. Podemos observar claramente que las principales ventajas de esta arquitectura son:

- El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.

- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

- Una pequeña desventaja de los procesadores con arquitectura Harvard Figura 15, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador). [10]

**Figura 15:** Arquitectura Harvard microcontroladores




Tomado de [10].

### 1.4.3 Familias de microcontroladores PIC

Los microcontroladores PIC se pueden clasificar, atendiendo al tamaño de sus instrucciones, en tres grandes grupos o gamas:

- **Gama baja:** microcontroladores con instrucciones de 12 bits.
- **Gama media:** microcontroladores con instrucciones de 14 bits.
- **Gama alta:** microcontroladores con instrucciones de 16 bits.

Los microcontroladores PIC también se agrupan en cinco grande familias: PIC10, PIC12, PIC16, PIC17 y PIC18. Los PIC10 son, básicamente, microcontroladores de 6 terminales. La familia de los PIC12 agrupa a los microcontroladores disponibles en encapsulado de 8 terminales. Algunas de estas cinco familias tienen números subfamilias, como sucede con los PIC16. Además algunas de estas familias incluyen dispositivos de más de una gama, como los PIC16 y PIC12, que tienen dispositivos de gama baja y media. Los PIC17 y PIC18 son gama alta. El criterio empleado para clasificar un PIC dentro de una familia es, pues, un tanto completo. La Tabla 8 muestra la ubicación de los PIC según sus terminales: [11].

**Tabla 8:** Resumen de la relación familia-gama en los microcontroladores

| FAMILIA | GAMA | | | RASGO DISTINTIVO |
|---|---|---|---|---|
| | BAJA | MEDIA | ALTA | |
| PIC10 | X | | | 6 Terminales |
| PIC12X5 | X | | | 8 Terminales |
| PIC12 (excepto PIC12X5) | | X | | 8 Terminales |
| PIC16X5 | X | | | - |
| PIC16 (excepto PIC16X5) | | X | | - |
| PIC17 | | | X | - |
| PIC18 | | | X | Gama alta mejorada |

### 1.4.4 Microcontroladores PIC18

Los microcontroladores PIC18 (gama alta) constituyen una numerosa familia de microcontroladores, que en su gran mayoría tienen memoria de programa tipo FLASH. Tienen un repertorio de 77 instrucciones de 16 bits. La memoria de programa puede ser de hasta 2MB ($2^{20}$ bytes ó $2^{10}$ palabras de 16 bits), y la memoria de datos puede llegar al 4 k (4096) registros de 8 bits cada uno. Algunos miembros de la familia PIC18 admiten una expansión externa de la memoria de programa. Poseen una pila de 31 niveles de profundidad, así como un sistema de interrupción muy elaborado, con interrupciones internas provenientes de los dispositivos de entrada y salida integrados en el Microcontrolador, y tres interrupciones externas, poseen un gran número y variedad de dispositivos de entrada y salida integrados [12].

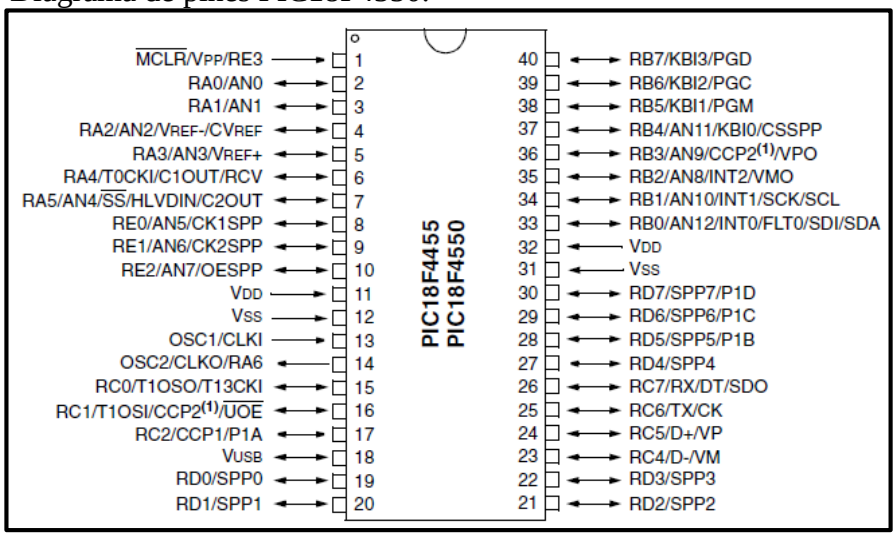
Varios dispositivos PIC18 están diseñados para trabajar con tensiones bajas (2.0 V a 3.6 V) y con corrientes inferiores de 2 mA.

#### 1.4.4.1 Microcontrolador PIC18F4550

La particularidad de este Microcontrolador es que está diseñado para el soporte de la comunicación USB, ya que incluye un controlador interne y un número de terminales para que se conectado directamente a un puerto USB. Soporta cristales y osciladores de varias frecuencias como entrada (4-20 MHZ), cuenta con 35 terminales de entrada y salida de propósito general, la presentación de su empaquetamiento es diversa, para desarrollo de esta guía se eligió el tipo 40PIN-PDIP que es un empaquetamiento de plástico de doble línea. Los puertos de entrada y salida son compatibles con la tecnología TTL y CMOS, se recomienda al usuario investigar sobre este tema. En cuanto a memoria, posee 32kb de flash para almacenamiento de programas, 2kb de SRAM para memoria volátil, y 256 bytes de EEPROM (memoria no-volátil) para almacenamiento permanente de datos como configuraciones y demás. El PIC18F4550 cuenta con un convertidor analógico/digital de hasta 12 bits de muestreo, que es útil para el desarrollo de aplicaciones de adquisición de datos. [12].

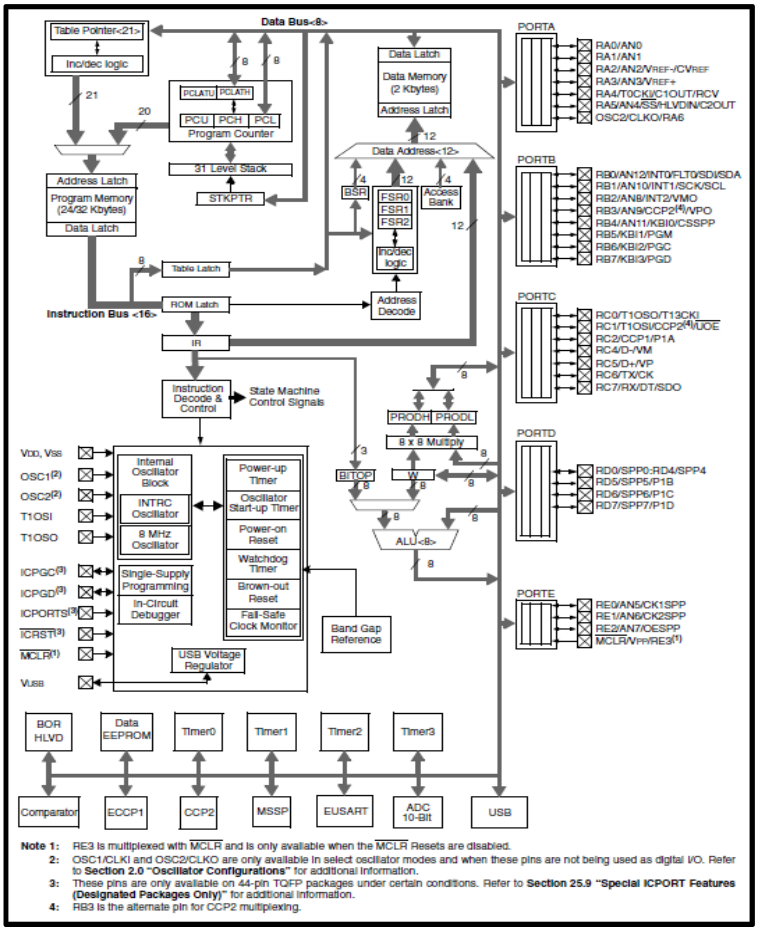
A continuación se mostrará una serie de figuras que relacionan los aspectos más importantes y utilizados: diagrama de pines Figura 16, arquitectura interna Figura 17, diagrama del oscilador Figura 18, módulo de conversión A/D Figura 19, Diagrama bloque transmisión y recepción EUSART Figura 20 y Figura 21, el diagrama de la lógica interna de las interrupciones Figura 22 y la información eléctrica del PIC18F4550 Figura 23.

**Figura 16:** Diagrama de pines PIC18F4550.



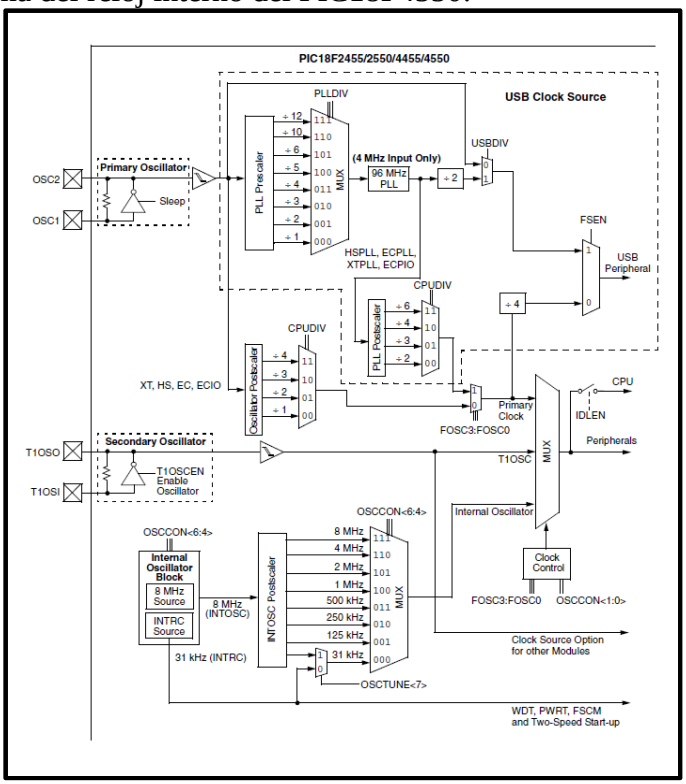

Tomado de [13].

**Figura 17:** Arquitectura interna PIC18F4550



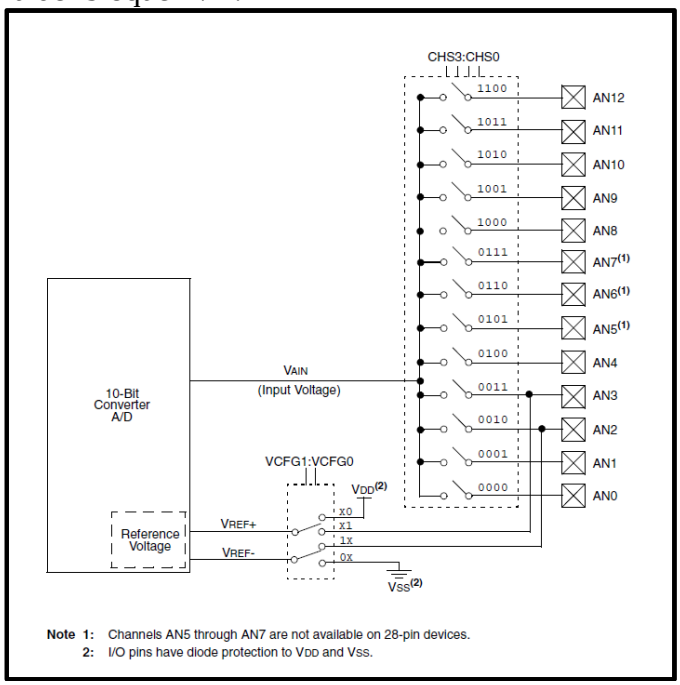

Tomado de [13].

**Figura 18:** Diagrama del reloj interno del PIC18F4550.



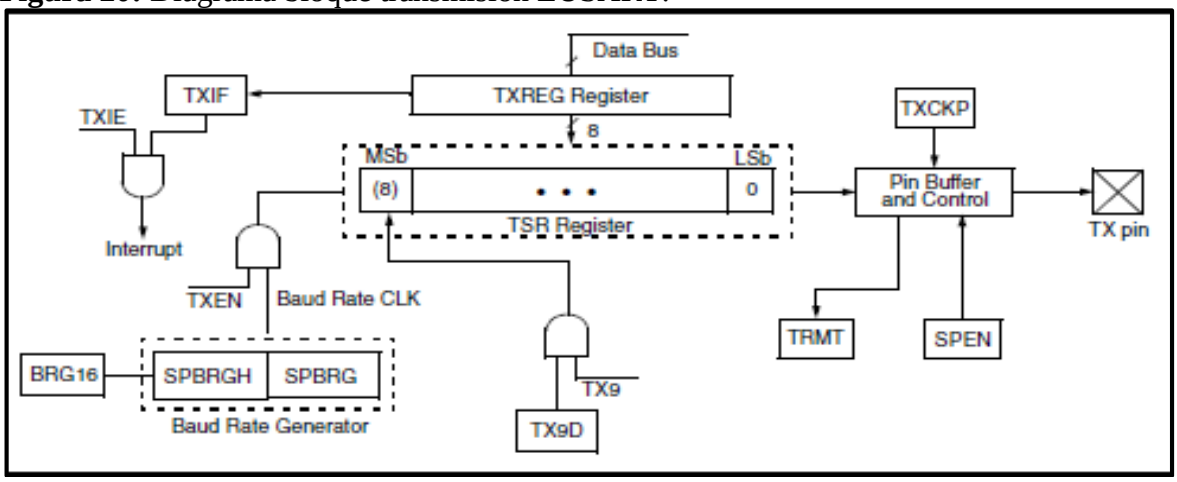

Tomado de [13].

**Figura 19:** Diagrama del bloque A/D.



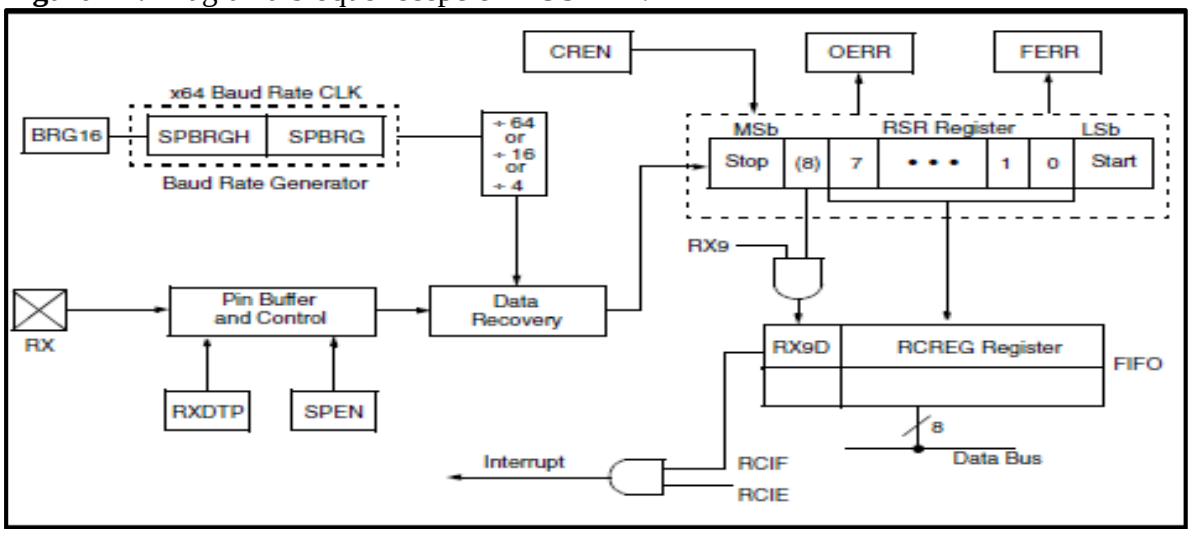

Tomado de [13].

**Figura 20:** Diagrama bloque transmisión EUSART.



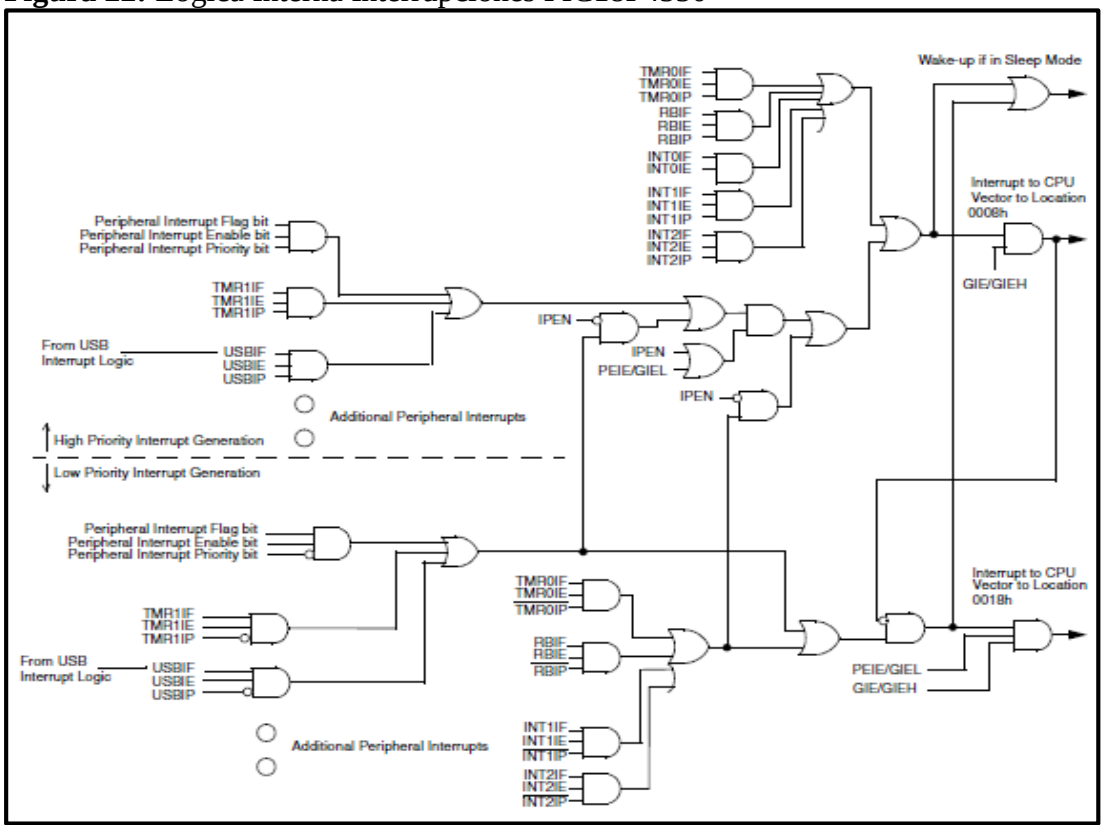

Tomada de [13].

**Figura 21:** Diagrama bloque recepción EUSART.




Tomada de [13].

**Figura 22:** Lógica Interna Interrupciones PIC18F4550




Tomada de [13].

**Figura 23:** Características Eléctricas PIC18F4550

**Absolute Maximum Ratings(†)**

| | |
|---|---|
| Ambient temperature under bias | -40°C to +85°C |
| Storage temperature | -65°C to +150°C |
| Voltage on any pin with respect to Vss (except VDD, $\overline{MCLR}$ and RA4) | -0.3V to (VDD + 0.3V) |
| Voltage on VDD with respect to Vss | -0.3V to +7.5V |
| Voltage on $\overline{MCLR}$ with respect to Vss **(Note 2)** | 0V to +13.25V |
| Total power dissipation **(Note 1)** | 1.0W |
| Maximum current out of Vss pin | 300 mA |
| Maximum current into VDD pin | 250 mA |
| Input clamp current, IIK (VI < 0 or VI > VDD) | ±20 mA |
| Output clamp current, IOK (VO < 0 or VO > VDD) | ±20 mA |
| Maximum output current sunk by any I/O pin | 25 mA |
| Maximum output current sourced by any I/O pin | 25 mA |
| Maximum current sunk by all ports | 200 mA |
| Maximum current sourced by all ports | 200 mA |

**Note 1:** Power dissipation is calculated as follows:
Pdis = VDD x {IDD – Σ IOH} + Σ {(VDD – VOH) x IOH} + Σ(VOL x IOL)

**2:** Voltage spikes below Vss at the $\overline{MCLR}$/VPP/RE3 pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$/VPP/RE3 pin, rather than pulling this pin directly to Vss.

Tomado de [13].

### 1.4.4.2 Organización de memoria PIC18F4550

El Microcontrolador PIC18F4550 dispone de las siguientes memorias [14]:

- **Memoria de programa**: memoria flash interna de 32.768 bytes
  - Almacena instrucciones y constantes/datos.
  - Puede ser escrita/leída mediante un programador externo o durante la ejecución del programa mediante unos punteros.

- **Memoria RAM de datos:** memoria SRAM interna de 2048 bytes en la que están incluidos los registros de función especial.
  - Almacena datos de forma temporal durante la ejecución del programa.
  - Puede ser escrita/leída en tiempo de ejecución mediante diversas instrucciones.

- **Memoria EEPROM de datos:** memoria no volátil de 256 bytes.
  - Almacena datos que se deben conservar aun en ausencia de tensión de alimentación
  - Puede ser escrita/leída en tiempo de ejecución a través de registros.

- **Pila:** bloque de 31 palabras de 21 bits
  - Almacena la dirección de la instrucción que debe ser ejecutada después de una interrupción o subrutina.

- **Memoria de configuración**: memoria en la que se incluyen los bits de configuración (12 bytes de memoria flash) y los registros de identificación (2 bytes de memoria de solo lectura).

## 1.5 JOYSTICK

### 1.5.1 Definición

Es un dispositivo con una palanca especial para ser tomado de manera ergonómica con 1 mano, y una serie de botones integrados en la palanca que controlan en la pantalla los movimientos y acciones de los objetos en los videojuegos ver Figura 24. Estos dispositivos se conectan en los puertos de la computadora y envían señales que esta misma procesa y en algunos modelos reciben órdenes para vibrar y crear en el Gamer ó jugador, una sensación de realismo [15].

**Figura 24:** Joystick marca Logitech®, modelo Extreme 3D Plus, con palanca, botones y gatillo, conector USB.

Tomada de [15].

### 1.5.2 Características generales de los joysticks

- Es un dispositivo que se adapta al manejo con una mano, integra botones básicos para controlar los videojuegos, y dependiendo el modelo también puede tener opcionalmente una serie de botones extras en la palanca.

- El tamaño de la palanca es grande, ya que se toma con toda la mano, a diferencia de los Gamepad que se utilizan ambas manos para controlarlo.

- Ha competido en el mercado directamente contra otros dispositivos como el Gamepad y contra los RaceWheel ó volantes para juego.

- Han habido 2 tipos básicos de palanca en el Joystick; los digitales (basado en mecanismos que permiten 2 estados lógicos: encendido y apagado por medio de pequeños pulsadores) y los análogos que tienen potenciómetros para detectar las posiciones) [15].

### 1.5.3 Partes del joystick

**Figura 25: Partes de un joystick**


Tomada de [15].

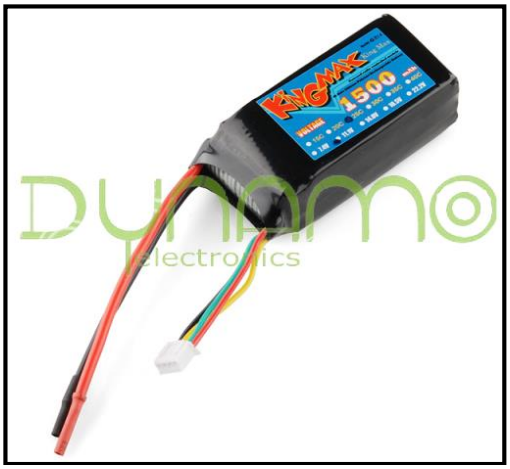
a. **Cubierta:** protege los circuitos internos el dispositivo y da estética al producto.
b. **Palanca:** permiten el control de movimiento de los gráficos del juego en pantalla con varias direcciones.
c. **Gatillo:** se usa para realizar el movimiento más común que es el disparo.
d. **Botones superiores:** tienen funciones secundarias.
e. **Botones inferiores:** tiene funciones primarias.

## 1.6 BATERÍAS

### 1.6.1 Baterías de polímero de litio (LiPo)

Son una variación de las baterías de iones de litio (Li-ion). Sus características son muy similares, pero permiten una mayor densidad de energía, así como una tasa de descarga bastante superior. Estas baterías tienen un tamaño más reducido respecto a las de otros componentes Figura 26. [16]

**Figura 26:** Batería Polímero litio ion 1500 11.1V




Tomado de [17].

Cada celda tiene un voltaje nominal de 3,7 V, voltaje máximo 4,2 y mínimo 3,0. Este último debe respetarse rigurosamente ya que la pila se daña irreparablemente a voltajes menores a 3 voltios. Se suele establecer la siguiente nomenclatura XSYP que significa X celdas en serie, e Y en paralelo. Por ejemplo 3s2p son 2 baterías en paralelo, donde cada una tiene 3 celdas o células. Esta configuración se consigue conectando ambas baterías con un cable paralelo. [16]

### 1.6.2 Ventajas y desventajas de las baterías (LiPo)

- Mayor densidad de carga, por tanto tamaño reducido.

- Buena tasa de descarga, bastante superior a las de iones de litio.
- Quedan casi inutilizadas si se descargan por debajo del mínimo de 3 voltios.

### 1.6.3 Tipos de baterías LiPo

Las baterías LiPo se venden generalmente de 1S a 4S lo que significa:

- Li-PO 1S: una celda, 3,7 V.
- Li-PO 2S: dos celdas, 7,4 V.
- Li-PO 3S: tres celdas, 11,1 V.
- Li-PO 4S: cuatro celdas, 14,8 V.

### 1.6.4 Usos

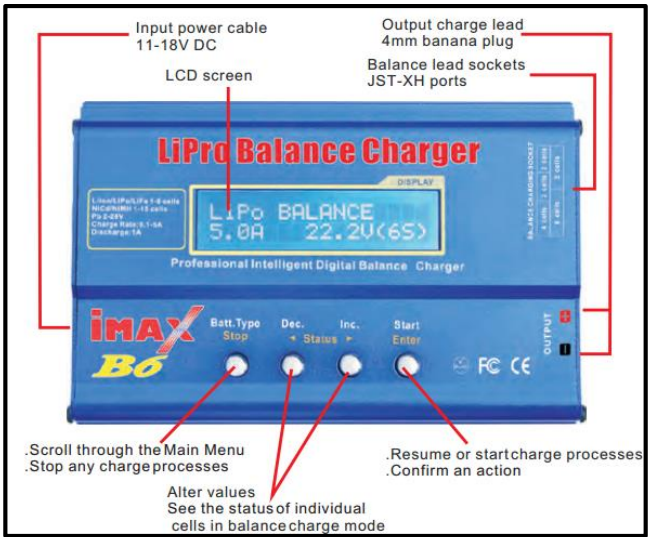
Su tamaño y peso las hace muy útiles para equipos pequeños que requieran potencia y duración, como manos libres bluetooth.

### 1.6.5 Cargador baterías LiPo

Para este proyecto se adquirió el Cargador iMax B6 que carga y balancea con precisión baterías de polímero de lito, de LiFe (A123), de NiCd y de NiMH, con el cual se puede cargar hasta 6s (LiPo/LiFe) muestreando la tensión de cada celda en tiempo real Figura 27.

El menú es intuitivo y fácil de usar, e incluye cables de entrada y salida, este cargador tiene conector de carga JST-XH compatible con baterías Zippy, HTX, iMax entre otras.
El cargador no incluye adaptador de 12v y el conector de entrada DC no es estándar, sin embargo incluye un conector adaptado a caimanes, lo cual hace fácil su adaptación con cualquier adaptador [18].

**Figura 27:** Cargador iMax B6




Tomado de [18].

# 2. ROBÓTICA

## 2.1 ORIGEN Y DESARROLLO DE LA ROBÓTICA

La robótica como hoy en día la conocemos, tiene su origen hace cientos de años atrás, en aquel tiempo los robots eran conocidos como autómatas, y la robótica no era reconocida como ciencia, la palabra robot viene de una programación. Una imitación de actividades que el ser humano pueda hacer o que aún desconoce. La palabra robot viene de "Robota", esta significa labor del ser humano. [19]

Los robots llevan más de 40 años de estar en los procesos industriales del ser humano ; los robots verdaderos se dieron a conocer a finales de los años 50 y principio de los 60; esto gracias a un nuevo desarrollo de la tecnología, es decir la invención de los transistores y circuitos integrados, los primeros robots industriales salen a la luz pública conocidos con el nombre de Unimates los cuales fueron diseñados por Gorge Devol y Joe Engelverger , este último creó el UNIMATION y fue el primero en mercadear estas máquinas, con el cual se ganó el título de "Padre de la robótica" [19]

Se pueden distinguir cinco fases relevantes en el desarrollo de la Robótica Industrial:

- El laboratorio ARGONNE diseña, en 1950, manipuladores amo-esclavo para manejar material radioactivo.

- Unimation, fundada en 1958 por Engelberger y hoy absorbida por Whestinghouse, realiza los primeros proyectos de robots a principios de la década de los sesentas de nuestro siglo, instalando el primero en 1961 y posteriormente, en 1967, un conjunto de ellos en una factoría de general motors. Tres años después, se inicia la implantación de los robots en Europa, especialmente en el área de fabricación de automóviles. Japón comienza a implementar esta tecnología hasta 1968.

- Los laboratorios de la Universidad de Stanford y del MIT acometen, en 1970, la tarea de controlar un robot mediante computador.

- En el año de 1975, la aplicación del microprocesador, transforma la imagen y las características del robot, hasta entonces grande y costoso.

- A partir de 1980, el fuerte impulso en la investigación, por parte de las empresas fabricantes de robots, otros auxiliares y diversos departamentos de Universidades de todo el mundo, sobre la informática aplicada y la experimentación de los sensores.

Ya en la década de los 80 los brazos industriales modernos incrementaron su capacidad y desempeño, mediante circuitos electrónicos más avanzados como microcontroladores y lenguajes de programación. Estos avances se lograron gracias a las grandes inversiones de las empresas automovilísticas.

Hoy en día, robots hacen una misma función cuantas veces se necesite sin cansarse ni aburrirse, siempre van a dar un mismo resultado y lo único que necesitan es una fuente de poder (electricidad). Estos robots cuentan con procesadores baratos y rápidos que hacen a estos más inteligentes y menos caros. Aunque la investigación por hacer que los robots "piensen" más eficazmente, la meta es hacer que en un futuro los robots sean suficientemente flexibles para hacer cualquier acción que un ser humano pueda hacer.

La evolución de los robots industriales desde sus principios ha sido vertiginosa. En poco más de 30 años las investigaciones y desarrollos sobre robótica industrial han permitido que los robots tomen posiciones en casi todas las áreas productivas y tipos de industria. En pequeñas o grandes fábricas, los robots pueden sustituir al hombre en aquellas áreas repetitivas y hostiles, adaptándose inmediatamente a los cambios de producción solicitados por la demanda variable [19].

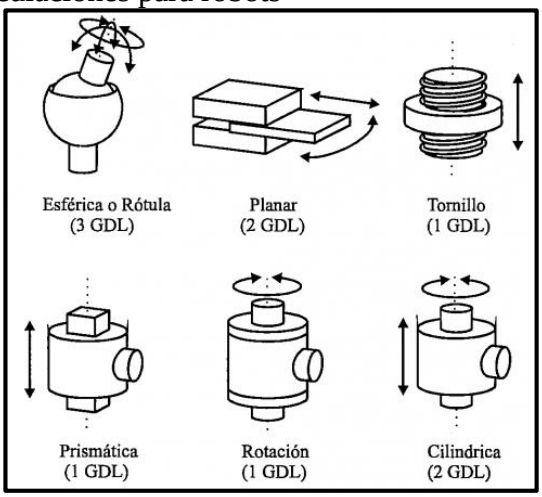
## 2.2 MORFOLOGÍA DEL ROBOT

El brazo robótico o manipulador es un brazo angular o antropomórfico que puede constar de una **base rotatoria, hombro, codo, muñeca y el manipulador o herramienta final.** Los movimientos de estas articulaciones están controlados por medio de servomotores, un servomotor por articulación, donde cada articulación tiene sólo un grado de libertad, o sea, desempeña un movimiento de desplazamiento o de giro.

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. La constitución física de la mayor parte de los robots industriales guarda cierta similitud con la anatomía del brazo humano, por lo que en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cuerpo, brazo, codo y muñeca. [20]

El movimiento de cada articulación puede ser de desplazamiento, de giro, o una combinación de ambos. De este modo son posibles los seis tipos diferentes de articulaciones que se muestran en la Figura 28, aunque, en la práctica, en los robots sólo se emplean la de rotación y la prismática. De este modo son posibles los seis tipos diferentes de articulaciones que se muestran en la Figura 28 [20].

**Figura 28:** Tipos de articulaciones para robots



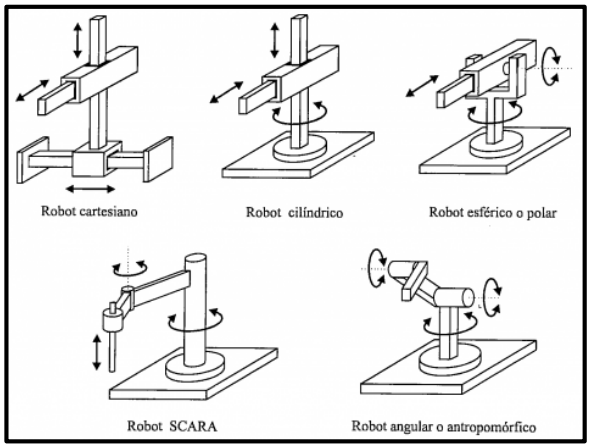

Tomado de [20].

### 2.2.1 Grados de libertad

Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior, se denomina **Grado De Libertad** (GDL). En la Figura 29 se indica el número de GDL de cada tipo de articulación. El número de grados de libertad del robot viene dado por la suma de los grados de libertad de las articulaciones que lo componen. El número de GDL del robot coincide con el número de articulaciones de que se compone. El empleo de diferentes combinaciones de articulaciones en un robot, da lugar a diferentes configuraciones, con características a tener en cuenta tanto en el diseño y construcción del robot como en su aplicación.

Las combinaciones más frecuentes son las representadas en la Figura 29 donde se atiende únicamente a las tres primeras articulaciones del robot, que son las más importantes a la hora de posicionar su extremo en un punto del espacio.

**Figura 29:** Estructuras mecánicas frecuentes en robots industriales




Tomado de [20].

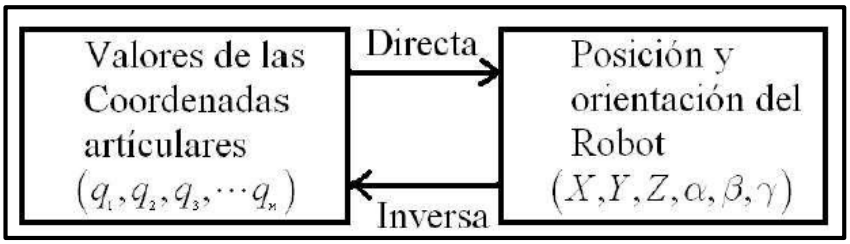
Puesto que para posicionar y orientar un cuerpo de cualquier manera en el espacio son necesarios seis parámetros, tres para definir la posición y tres para la orientación, si se pretende que un robot posicione y oriente su extremo (y con él la pieza o herramienta manipulada) de cualquier modo en el espacio, se precisarán de seis GDL. [20]

## 2.3 CINEMÁTICA DEL ROBOT

### 2.3.1 Cinemática del robot:

La cinemática del robot estudia el movimiento con respecto a un sistema de referencia. Existen 2 problemas fundamentales, cinemática directa e inversa. La cinemática directa consiste en determinar la posición y orientación del extremo final del robot en función de las coordenadas articulares y la cinemática inversa determina las coordenadas articulares en función de la posición final del robot Figura 30, [21].

**Figura 30:** Cinemática del robot.




Tomada de [21].

La cinemática del robot trata también de encontrar las relaciones entre las velocidades de cada articulación y la del extremo. A esto se le conoce como modelo diferencial. Denavit y Hartenberg propusieron un método sistemático para representar la geometría espacial de los elementos de una cadena cinemática de un robot.

### 2.3.2 Problema cinemático directo:

Un robot se puede considerar como una cadena cinemática formada por eslabones unidos por articulaciones. Se establece un sistema de referencia fijo solidario a la base, se describe la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. El problema cinemático directo se reduce a encontrar la matriz de transformación T la cual es función de las coordenadas articulares Figura 31. Para sistemas de hasta 3 grados de libertad se puede usar un método trigonométrico.

**Figura 31:** Funciones de posición y ángulos de desplazamiento del brazo.

$$
\begin{aligned}
x &= f_x(q_1, q_2, q_3, q_4, q_5, q_6) \\
y &= f_y(q_1, q_2, q_3, q_4, q_5, q_6) \\
z &= f_z(q_1, q_2, q_3, q_4, q_5, q_6) \\
\alpha &= f_\alpha(q_1, q_2, q_3, q_4, q_5, q_6) \\
\beta &= f_\beta(q_1, q_2, q_3, q_4, q_5, q_6) \\
\gamma &= f_\gamma(q_1, q_2, q_3, q_4, q_5, q_6)
\end{aligned}
$$

Tomado de [21].

### 2.3.3 Problema cinemático inverso:

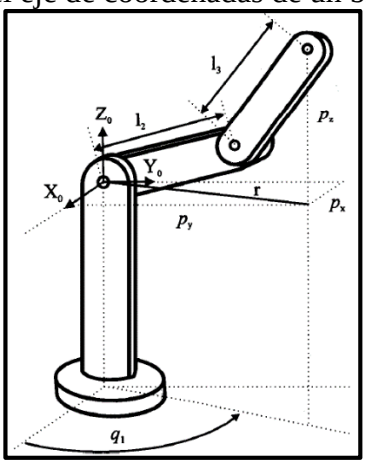
El problema cinemático inverso consiste en encontrar los valores que debe adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial. Existen más de una solución, se puede despejar de la matriz de transformación homogénea, se puede obtener usando trigonometría. Lo adecuado es encontrar una solución cerrada, es decir encontrar una relación matemática explícita de la forma:

$$\boldsymbol{q_k = f_k(x, y, z, \alpha, \beta, \gamma)} \qquad \textbf{(4)}$$

Donde:

$$k = 1 \ldots n \rightarrow (GDL)$$

**Figura 32:** Representación en el eje de coordenadas de un brazo robótico.




Tomada de [22].

En la Figura 32 se puede observar que a partir de una coordenada establecida se analiza un sistema de coordenadas y las distancias o características propias del mismo para así encontrar los ángulos de rotación. A continuación se muestran las ecuaciones utilizadas por el modelo

geométrico, las cuales definen los ángulos q2 y q3 a partir de coordenadas x, y previamente definidas:

$$q_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) \quad (5)$$

$$r^2 = p_x^2 + p_y^2 \quad (6)$$

$$r^2 + p_z^2 = l_2^2 + l_3^2 + 2l_2 l_3 \cos q_3 \quad (7)$$

$$\cos q_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2l_2 l_3} \quad (8)$$

$$\sin q_3 = \pm\sqrt{1 - \cos^2 q_3} \quad (9)$$

$$q_2 = \beta - \alpha \quad (10)$$

$$\beta = \tan^{-1}\left(\frac{p_z}{r}\right) = \tan^{-1}\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) \quad (11)$$

$$\alpha = \tan^{-1}\left(\frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right) \quad (12)$$

$$q_3 = \tan^{-1}\left(\frac{\pm\sqrt{1 - \cos^2 q_3}}{\cos q_3}\right) \quad (13)$$

Tomadas de [22].

Además del método geométrico, también se puede resolver por medio de la cinemática directa que utiliza matrices de transformación homogénea, o por medio del Método de Denavit – Hartenberg (D-H), que implementa 4 matrices de transformación básicas, que dependen exclusivamente de las características constructivas del robot.

Para este proyecto específicamente, se utilizó el método geométrico aun cuando es un robot de 4 GDL, ya que una de las articulaciones se puede obviar del modelamiento, debido a que esta solo precisa de 3 posiciones 0°, 90° y 180°. Por lo tanto, el modelo geométrico se desarrolló para un sistema de 3 GDL.

El método geométrico se suele utilizar para obtener los valores de las primeras variables articulares, que son las que posicionan el robot (prescindiendo de la orientación de su extremo). Además, se utilizan relaciones geométricas y trigonométricas sobre los elementos

del robot. Para el desarrollo y cálculo de los ángulos fue preciso hacer uso de las características principales del brazo Tabla 9.

**Tabla 9:** Datos característicos del brazo robótico.

| Especificaciones | Valor en (in) | Valor en (cm) |
|---|---|---|
| Distancia entre base y codo | 3,75 | 9,52 |
| Distancia entre codo y muñeca | 4,75 | 12,06 |
| Altura | 6 | 15,24 |
| Altura (estirado) | 14 | 35,56 |
| Distancia media de avance | 9,5 | 24,13 |

Tomada de [23].

Finalmente, se puede decir que el análisis cinemático inverso permite a través de una coordenada previamente definida, encontrar los ángulos de las articulaciones (hombro, codo y muñeca). Por consiguiente, para el desarrollo y análisis de la cinemática inversa se utilizó un software llamado (*Another method for Inverse kinematics of Lynxmotion robotic arms*) el cual a partir de una coordenada genera los ángulos. Con este análisis y la simulación en el software LynxTerm se logró realizar un proceso detallado y adecuado para la manipulación de los servos del brazo, esto es, cada servo fue identificado y sus rangos máximos de operación fueron previstos. En la Tabla 10 se muestran los rangos permitidos por cada uno de los servos.

**Tabla 10:** Rangos de operación de los servomotores.

| RANGOS DE OPERACIÓN | | |
|---|---|---|
| **Servo** | **Rango [PWM]** | **Grados** |
| Servo 0 “Base” | [500-2500] | 180º |
| Servo 1 “Hombro” | [1400-2100] | 45º |
| Servo 2 “Codo” | [1600-1900] | 41º |
| Servo 3 “Muñeca” | [700-1500] | 80º |
| Servo 4 “Gripper” | [500-1500] | 90º |

Cada uno de estos rangos fue delimitado en el software final de manera que el usuario o la persona que manipule el brazo no excedan los rangos de operación normales y cause un daño en el servomotor.

# 3. CARACTERÍSTICAS DE LA COMUNICACIÓN INALÁMBRICA CON MÓDULOS XBEE

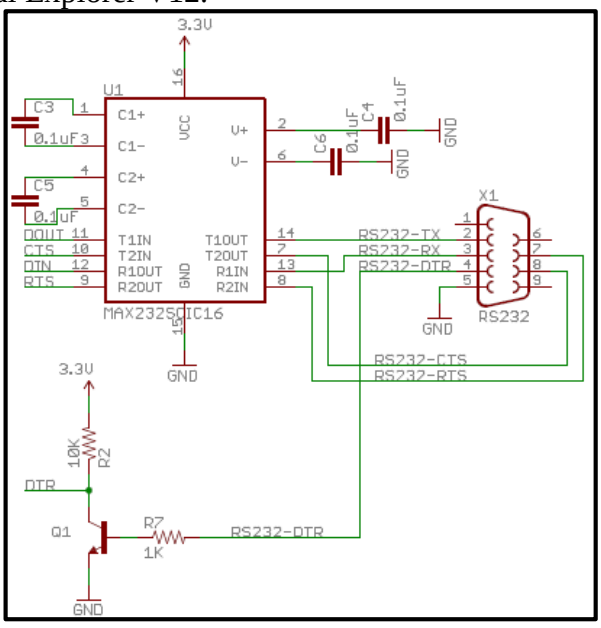
## 3.1 DISEÑO DEL SISTEMA DE COMUNICACIÓN INALÁMBRICA

Para este proyecto el sistema de comunicación implementado tiene una topología de red punto a punto, pero para observar las características, funcionalidades de estos módulos se realizaron pruebas utilizando dos módulos XBee Pro S1 y un XBee S1 creando diferentes topologías como árbol, malla y punto a punto, también se adquirió un circuito impreso de la empresa Sparkfun[1] para realizar pruebas con nuestros circuitos en protoboards cabe mencionar que los pines del módulo XBee no son compatibles con el tamaño de separación de las protoboards, para ello se requiere utilizar un adaptador a PDIP.

Para programar el firmware de estos módulos se debe utilizar un software facilitado por el fabricante **DIGI** llamado ***X-CTU***, esto también es posible realizarlo por medio del hyperterminal utilizando comandos AT, además del recurso del software se debe poseer una interfaz serial como medio físico para la sobre escritura de los parámetros del firmware del módulo XBee, en el mercado hay diferentes interfaces desde las USB hasta las mismas seriales. Para la configuración de los módulos inicialmente construimos una tarjeta serial utilizando el integrado max232 que permite la conversión de niveles RS232 a TTL.

En la Figura 33 se puede observar el esquemático con el cual se realizaron las pruebas iniciales y en la Figura 34 en circuito ensamblado para dicha prueba.

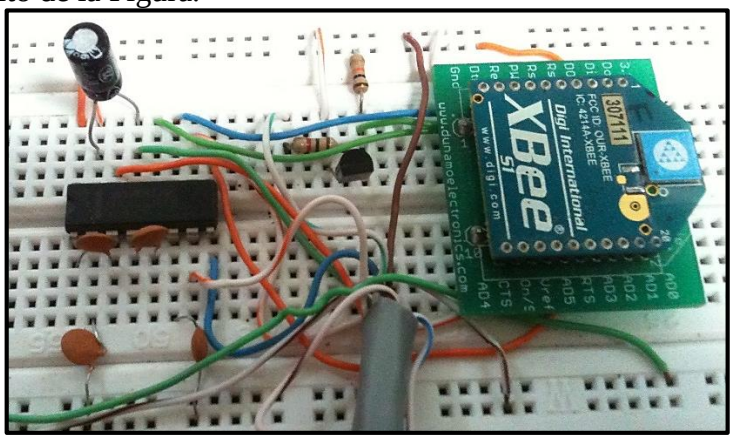
**Figura 33:** XBee Serial Explorer V12.




Tomado de [24].

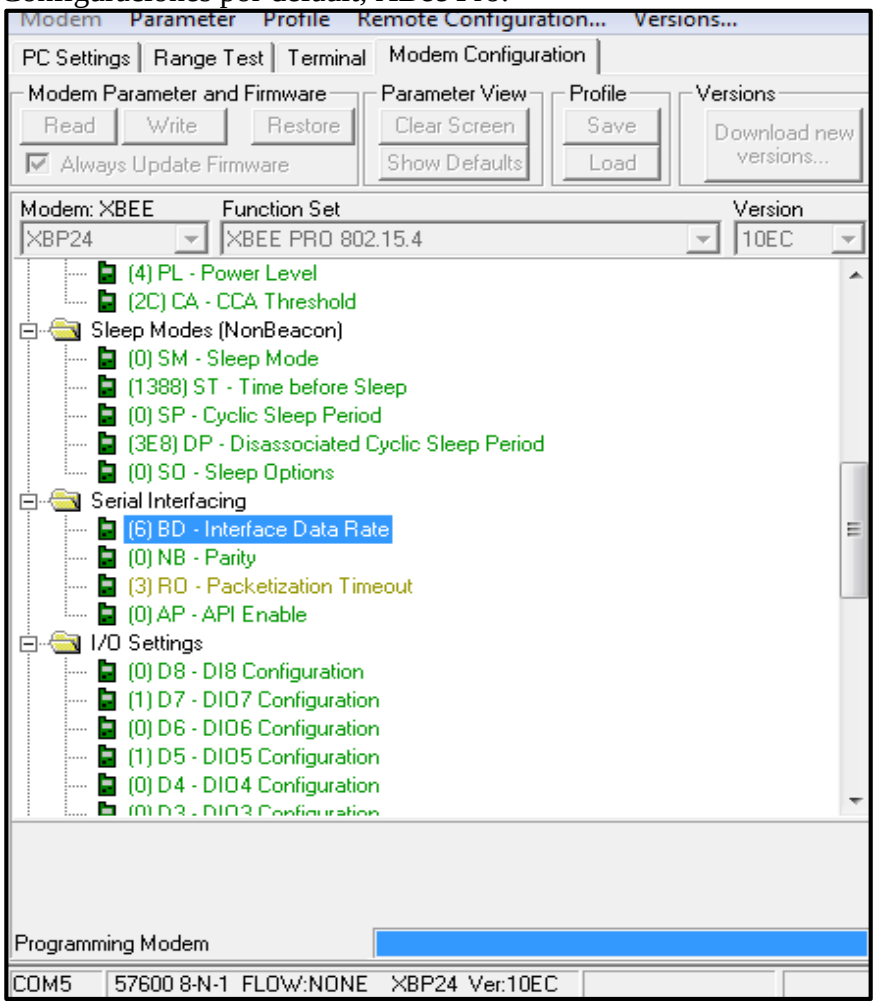
[1] XBee Explorer Serial, https://www.sparkfun.com/products/9111

**Figura 34**: Circuito de la Figura.




Teniendo en cuenta que la alimentación del XBee es de 3,3V se realizó la prueba con el software ***X-CTU*** ver Figura 35, adquiriendo la lectura de los parámetros actuales:

**Figura 35:** Configuraciones por default, XBee Pro.



## 3.2 PRUEBAS DE COMUNICACIÓN

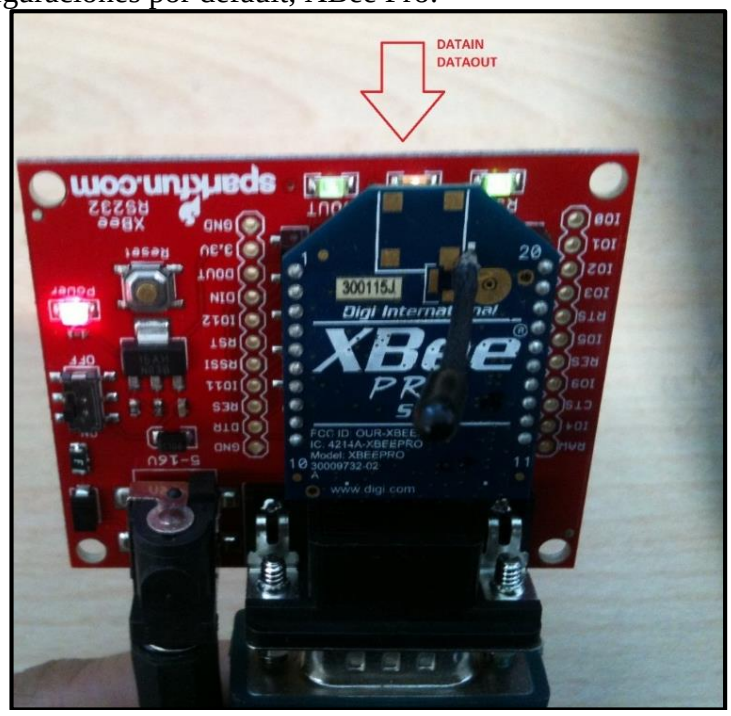
Las pruebas de comunicaciones realizadas, como se mencionó anteriormente se realizaron con la modificación de todos los parámetros (*Pan id, Key, Serial Interface, Api's .etc.*), después de esto se realizó la modificación de los diferentes modos de operación.

Las pruebas de comunicación se llevaron a cabo en tres etapas las cuales se realizaron con el fin de comprobar, como primera medida el estado del XBee, segundo para evidenciar la transmisión y recepción de datos y tercero para probar el envío de datos XBee-XBee (transmisión inalámbrica). A continuación se describen detalladamente cada una de estas.

### 3.2.1 Pc-XBee

Inicialmente, es necesario comprobar la conectividad del XBee con el Pc esto se realiza mediante el software X-CTU el cual identifica el XBee con un serial, de esta manera se puede decir que el modulo tiene una conexión establecida con el Pc. Adicionalmente, se realizar él envió de una cantidad limitada de tramas, para verificar que el dispositivo inalámbrico si estaba enviando la información adecuadamente, esto también se puede observar mediante los Led's indicadores de la tarjeta XBee Explorer, ver Figura 36.

**Figura 36:** Configuraciones por default, XBee Pro.



### 3.2.2 Pc XBee - XBee-Pc

Una vez comprobada la conectividad del XBee con el Pc, se procede a realizar una conexión entre los dos módulos XBee con el fin de establecer la configuración entre los dispositivos de la red punto a punto que se utilizó en el proyecto. Se desarrolló inicialmente una comunicación entre XBee utilizando el software del fabricante (X-CTU) enviando una cadena de caracteres como se puede apreciar en la Figura 37 de ambos lados (Transmisor y Receptor.). El montaje físico se puede observar en la Figura 38.

**Figura 37:** Prueba de Comandos AT entre XBee-XBee.

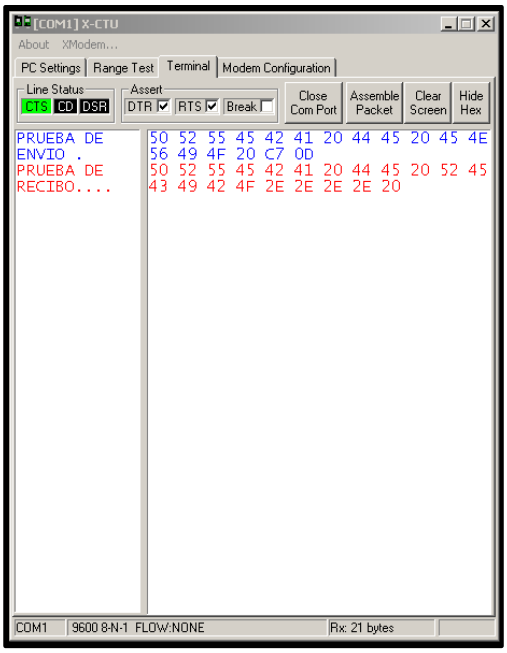
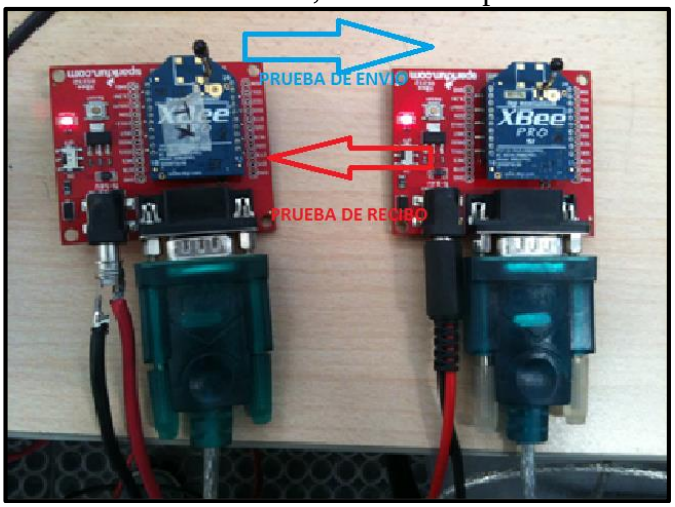


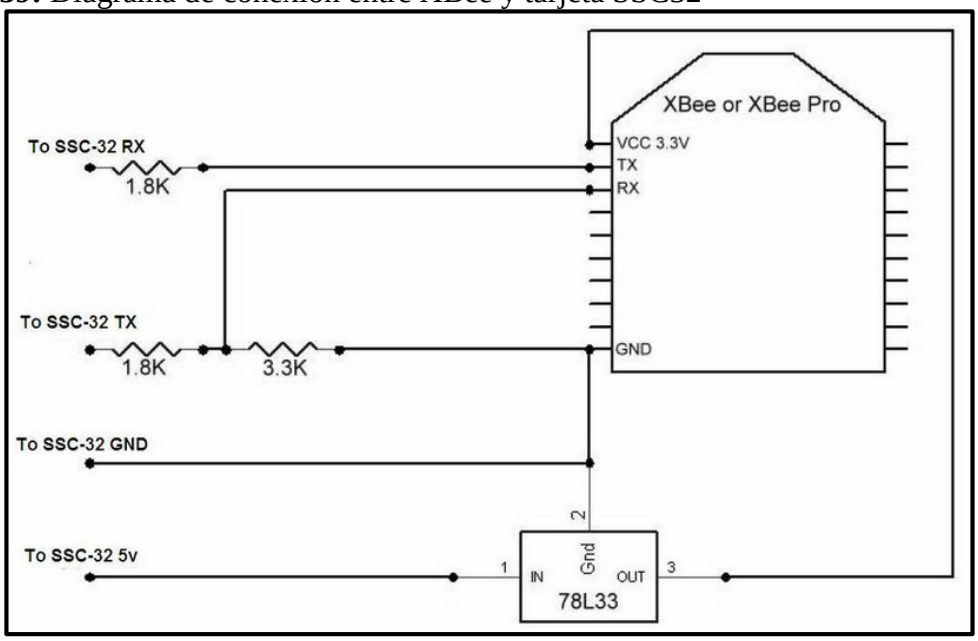

**Figura 38:** Prueba de comunicación exitosa, usando computador.



### 3.2.3 Pc-XBee- XBee-SSC32

En esta etapa se procede a examinar la conexión entre un módulo de comunicación y la tarjeta SSC-32, entonces, desde el software X-CTU se envía una trama específica y aceptable por la tarjeta del brazo de esta manera identificamos sí al enviar una posición determinada desde el Pc, la tarjeta lo reconocerá y convertirá en un pulso PWM para la ejecución de un movimiento en el brazo robótico. El funcionamiento de la tarjeta SSC-32, su forma de operación y la forma en la que recibe datos se especifica en el capítulo 4. Debido a que el nivel de tensión del XBee es de 3.3V y el de la tarjeta es de 5V es necesario hacer un divisor de tensión para obtener un nivel de tensión igual y no haya problemas de transmisión de datos. En la Figura 39 se puede observar el circuito necesario para realizar esta conexión.

**Figura 39:** Diagrama de conexión entre XBee y tarjeta SSC32




### 3.2.4 Microcontrolador-XBee-XBee-SSC32

Finalmente, y la etapa más importante de todas es la interacción entre el microcontrolador y la tarjeta SSC-32 a través de los dos módulos receptor y transmisor (XBee). Por lo tanto, una vez se comprueba el estado de los módulos y la transmisión y recepción correcta de datos, correcta en el sentido de que cada carácter enviado de forma serial es recibido exactamente igual, se puede entonces programar el microcontrolador para que a través de su módulo EUSART ó módulo de comunicación serial se envíen la tramas necesarias para la ejecución adecuada de diferentes movimientos por el brazo robótico.

### 3.3 MODO COMANDO

Como se mencionó anteriormente el módulo de comunicación inalámbrico puede recibir una serie de comandos y procesarlos, con el fin de probar este modo con el microcontrolador se implementa el código mostrado en la Figura 40, con el cual se verifió que este modo funciona correctamente.

**Figura 40:** Implementación del Modo Comandos AT en C18

```
void init_xbee(void)
{
char string[3],str[3]="OK";
delay100ms;
putcUSART('kX');
delay10ms;
putrsUSART("+++");
delay1s;delay1s;
getsUSART(string,3);
while (!strcmp(string,str)) {getsUSART(string,3);}
putrsUSART("ATDH 00\r");//destination address high
delay10ms;
putrsUSART("ATDL 02\r");// destination address low
delay10ms;
putrsUSART("ATID 10\r");//PAN ID
delay10ms;
//putrsUSART("ATBD 07\r");//set baud-rate 115200
//delay1ms;
putrsUSART("ATMY 01\r");//Source address
delay10ms;
putrsUSART("ATWR\r");//write to non-volatile memory
delay10ms;
//putrsUSART("ATAC\r");//write to non-volatile memory
//delay1ms;
putrsUSART("ATCN\r");//exit command mode
delay100ms;}
```

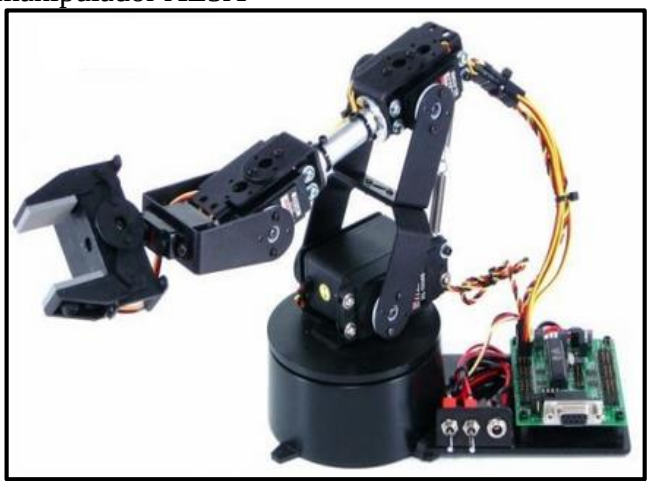
Finalmente se utiliza una configuración punto a punto, probando la teoría de las diferentes topologías de red soportadas por los dispositivos XBee con una tasa de transmisión de 115Kbps en modo transparente y comprobación de errores en la transmisión de los datos, también se exhibe la tecnología de los módulos y su protocolo base ZigBee , desde el punto de vista técnico estos permiten crear solución a la medida de bajo coste donde se requiera utilizar un ancho de banda reducido y bajo consumo de potencia.

Se utiliza un direccionamiento correcto para evitar la pérdida de datos, pues se observa que si se utiliza el modo transparente sin direccionamiento y con características por default cuando el modulo transmita en el momento requerido y reciba datos puede existir una posible colisión, el direccionamiento permite facilitar el intercambio de información entre los nodos de una red, verificando la correcta transmisión de datos, además de que el estándar 802.15.4 mejora la transmisión teniendo en cuenta la distancia entre los nodos.

# 4. CARACTERÍSTICAS DEL BRAZO ROBÓTICO

Inicialmente surgió la idea de desarrollar un prototipo industrial el cual simulara el desempeño de un brazo robótico capaz de obtener y trasladar materia prima de una banda a otra en una línea de producción. Este brazo debía tener la capacidad de girar en un punto fijo 180 grados, tener por lo menos 3 articulaciones y una pinza o gripper para la obtención de dicho material. Según esta idea y por el cumplimiento de esas especificaciones se seleccionó el brazo AL5A desarrollado por la compañía Lynxmotion, este fue adquirido en la tienda electrónica Dynamo Electronics ubicada en Bucaramanga, Colombia. La Figura 41 muestra el brazo AL5A. Adicionalmente, se diseñó la base para situar el brazo, para conocer las fichas técnicas dirigirse al Anexo 1.

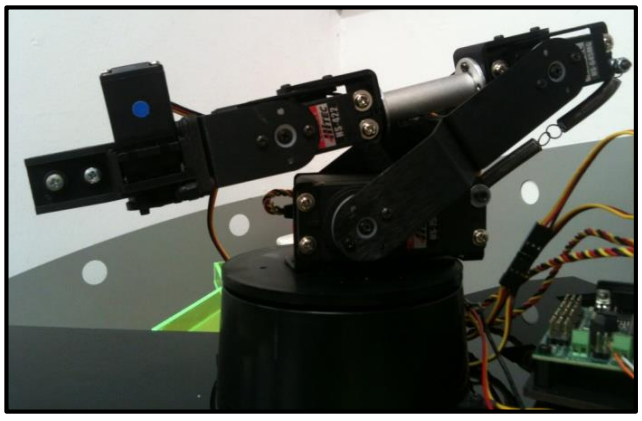
**Figura 41:** Brazo o manipulador AL5A



Tomada de [18]

Una vez se obtuvo este modelo, ensamblamos pieza por pieza hasta obtener el brazo completamente armado como se muestra en la Figura 42:

**Figura 42:** Brazo AL5A ensamblado:



Las características de este brazo son:

**Especificaciones:**

- No de Ejes = 4 + Gripper; rotación en la muñeca opcional
- Distancia entre base y codo = 3,75"
- Distancia entre codo y muñeca = 4,75"
- control = lazo cerrado local por medio de servomotores
- Altura= 6"
- Altura (estirado) = 14"
- Distancia media de avance= 9,5"
- Apertura del gripper = 1.25"
- Capacidad de carga (brazo extendido) = approx. 4 oz
- Peso = 24oz
- Rango de movimiento por eje = 180 grados
- Precisión de movimientos por eje = SSC32 0.09 grados
- Tensión de servos = 6 Vdc

Los servomotores utilizados en el brazo son los siguientes:

**Tabla 11:** Servomotores utilizados en el brazo AL5A

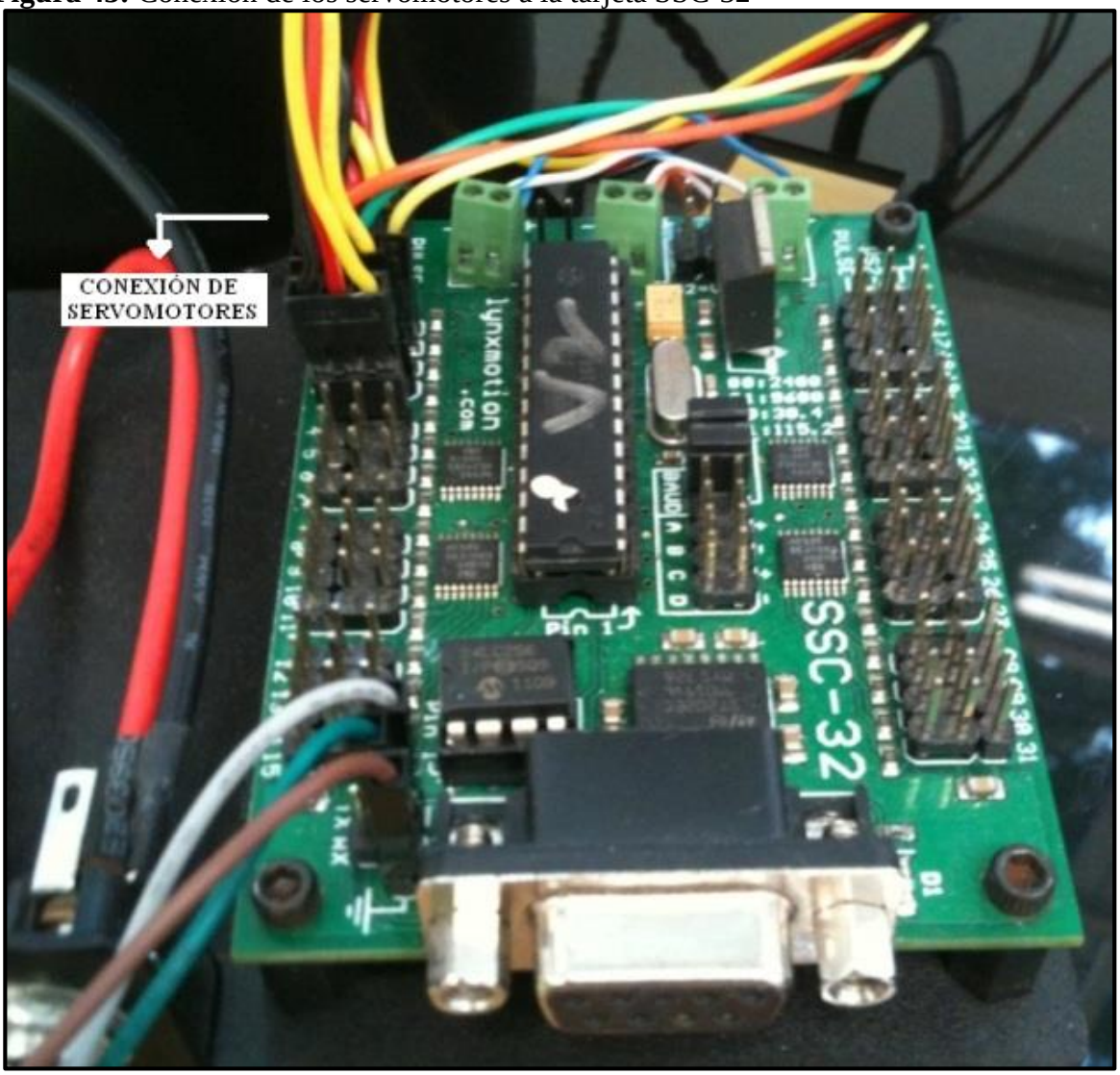
| **SERVOMOTORES** | | | |
|---|---|---|---|
| **1 x HS-755HB** | | **2 x HS-422** | |
|  | Hombro |  | Base y Muñeca |
| **1 x HS-645MG** | | **1 x HS-422** | |
|  | Codo |  | Gripper |

El brazo robótico de este proyecto es un brazo reprogramable con 4 GDL el cual está diseñado para la recolección ó toma de piezas u objetos en tres (3) posiciones previamente definidas las cuales alcanzan un rango de 180 grados. Este brazo robótico es controlado a través de Servomotores ubicados en sus articulaciones (Cintura o base, hombro, Codo, muñeca y

pinza) según estas características hace las veces de un brazo humano, sin embargo carece de conocimiento de lo que está en su entorno.

Cada servomotor es controlado por medio de la tarjeta de control SSC-32 la cual está encargada de enviar señales PWM. La tarjeta SSC32 tiene la posibilidad de conectar 32 servomotores cada uno de forma independiente, esta conexión se realiza a través de canales enumerados desde cero hasta treinta y uno [0-31], para el brazo de este proyecto el cual solo tiene cuatro grados de libertad 4GDL se utilizaron cuatro canales y sus articulaciones o servomotores están ubicados de la siguiente manera: base canal cero (0), hombro canal uno (1), codo canal dos (2), muñeca canal tres (3) y finalmente la pinza en el canal cuatro (4) así como se muestra en la Figura 43.

**Figura 43:** Conexión de los servomotores a la tarjeta SSC-32



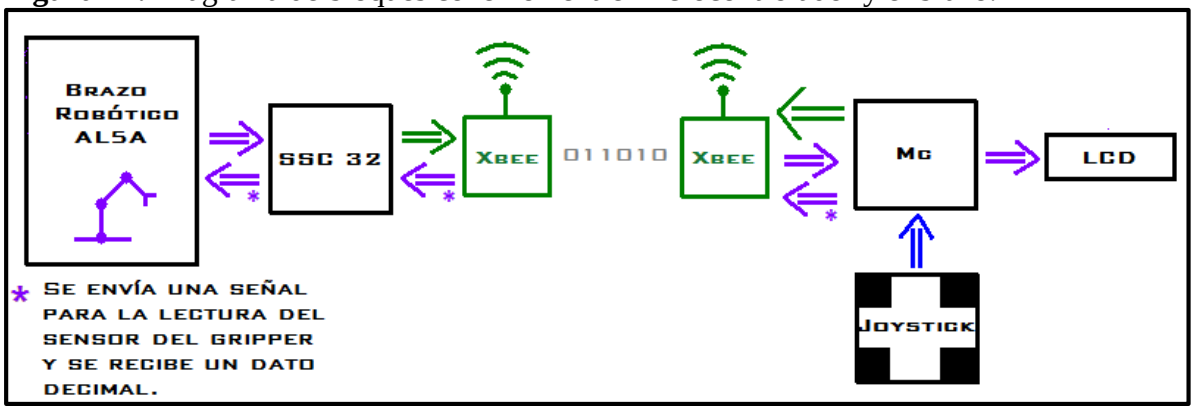
Adicionalmente, para el control de los movimientos del brazo es necesario enviar datos específicos los cuales contienen: el canal (servo), la posición y el tiempo del movimiento. Estos datos son enviados desde el microcontrolador PIC18F4550 a través del módulo USART (*Universal asynchronous receiver/transmitter*) que a su vez son traducidos por la tarjeta SSC32 a señales PWM y finalmente interpretadas por los servomotores para ejecutar el movimiento. La trama o código se muestra en la Tabla 12.

**Tabla 12:** trama para el envío de datos de la tarjeta SSC32

| TRAMA | DESCRIPCIÓN |
|---|---|
| **putrsUSART("#0P1500T1000\r")** | Envía la cadena de caracteres **("#0P1500T1000\r")** a través del módulo USART[2]. Con ese código se moverá el servo cero (base) a la posición 1500 y el movimiento se realizará en 1 segundo. |

En la Figura 44 se muestra un diagrama de bloques que da una idea general sobre la forma o la interfaz entre los elementos que permite la interacción con el brazo. Este esquema muestra las partes en las cuales se desarrolló el proceso de elaboración, no solo para programar el brazo sino para comunicarlo con el joystick inalámbricamente.

**Figura 44:** Diagrama de bloques conexión entre microcontrolador y el brazo.




Finalmente, es necesario pasar ese modelo a posiciones específicas, para luego ser programadas en el microcontrolador, además, las instrucciones deben ser enviadas a través de comunicación inalámbrica para finalmente ser recibidas por la tarjeta SSC 32 que traduce las tramas o posiciones que se nombran en el titulo 4.1 en movimientos específicos.

[2] La función putrsUSART es una función de la librería usart.h la cual está contenida en el compilador C18.

## 4.1 ANÁLISIS Y CONTROL DEL BRAZO ROBÓTICO

Una vez se tiene claro cómo se realizan los movimientos de cada servo es necesario analizar los rangos que cada uno de estos alcanza y como debe llevarse a cabo un movimiento grupal o de varios servos, para ello, fue necesario el uso de dos recursos o software principales desarrollados por la Lynxmotion los cuales son: el LynxTerm y (*Another method for Inverse kinematics of Lynxmotion robotic arms*).

Con la ayuda de estos métodos y conociendo los límites de los servos, se implementó un programa basado en intervalos dependientes del joystick, los valores de los potenciómetros de la palanca referidos a los movimientos frontal, lateral, y movimiento de circunducción son leídos por el microcontrolador, el cual se encarga de convertir la señal análoga en digital. Una vez se tiene la señal digital se adapta al rango aceptado por los servomotores, es decir, si un dato está entre [0,200] se pasa a [500, 1500]. Por lo tanto, el movimiento del servo es un movimiento dependiente de un dato variable entregado por el joystick.

Así pues, el brazo ejecuta un movimiento según el desplazamiento continuo de la palanca del joystick, entonces, cada servo se identifica o tiene una relación con un potenciómetro de la palanca, en la Tabla 13 se muestra esta relación. Sin embargo, en el siguiente capítulo se darán más detalles al respecto.

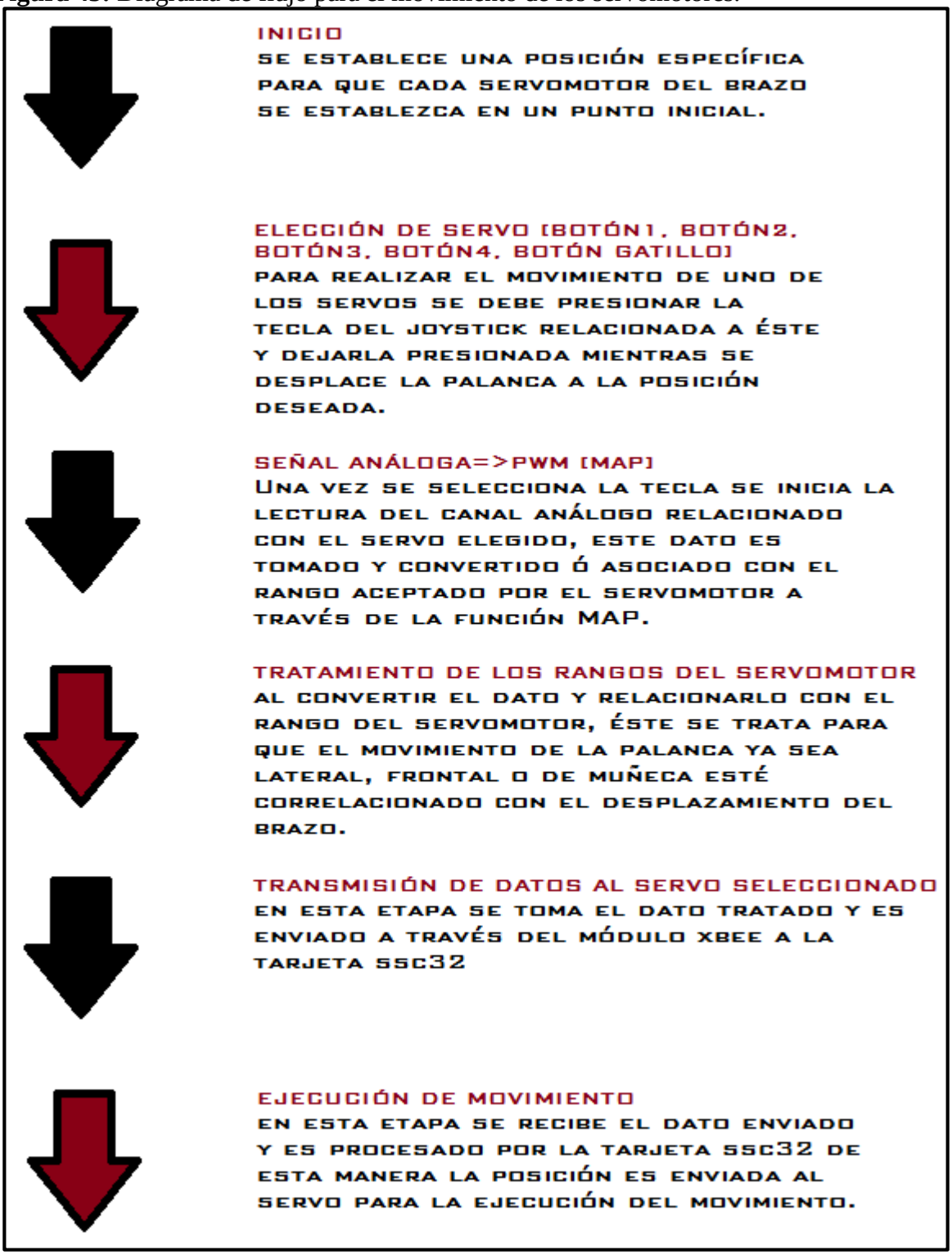
**Tabla 13:** Relación movimiento de servos por medio del joystick.

| **RELACIÓN SERVOMOTOR - JOYSTICK**. | |
|---|---|
| Servo 0 “base” | Potenciómetro Lateral + Botón1 |
| Servo 1 “Hombro” | Potenciómetro Frontal + Botón2 |
| Servo 2 “Codo” | Potenciómetro Muñeca + Botón3 |
| Servo 3 “Muñeca” | Potenciómetro Muñeca + Botón4 |
| Servo 4 “Gripper” | Botón Gatillo. |

Inicialmente, por defecto se inicia el brazo con una posición específica, llamada función de *inicio*, después se realiza una correlación entre la señal análogo-digital de los potenciómetros y los valores aceptables por los servomotores, esta correlación o mapeo se realiza a través de una función llamada Map[3], inmediatamente este dato se envía por el módulo de comunicación XBee a la tarjeta SSC32 y posteriormente se ejecuta el movimiento. En la Figura 45 se muestra un diagrama con el proceso descrito anteriormente.

[3] Arduino es el autor intelectual de la función Map, para ver detalles de esta función visitar la página web http://arduino.cc/en/pmwiki.php?n=Reference/Map.

**Figura 45:** Diagrama de flujo para el movimiento de los servomotores.



El envío del dato se realizó mediante una función llamada *sprintf* de la librería *stdlib.h*, esta función permite guardar en una variable una combinación de caracteres y una variable, en este caso la variable es la posición del servomotor. A continuación en la Tabla 14 se muestran unos fragmentos de código con el proceso desde la selección del servomotor hasta el envío de la posición deseada al servomotor, a manera de ejemplo se mostrará este proceso para el servo 0 ó "base", del mismo modo, este procedimiento se realiza con los demás servos.

**Tabla 14:** Código utilizado para él envió de posiciones al servo 0 "base".

```
/*SELECCIÓN DE TECLA "BASE"*/
```

```
if(SERV0==1)         //TECLA SERV0
{do{AdcConvert0();}  //IR AL CONVERTIDOR ANÁLOGO-DIGITAL.
while(SERV0==1);
}
```

```
/*OBTENCIÓN DE SEÑAL A TRAVÉS DEL CONVERTIDOR ANÁLOGO DIGITAL PARA EL MOVIMIENTO DEL SERVO #0 "BASE"
EJECUTADO POR EL MOVIMIENTO DEL POTENCIÓMETRO LATERAL*/
```

```
void AdcConvert0(void)
{
ADCON0bits.ADON=0x01;   //HABILITAR EL MODULO A/D.
SetChanADC(ADC_CH1);    //LATERAL
Delay10TCYx(1);
ConvertADC();
while(BusyADC()==1){}
adc_result1 = ADRESH;
SERVO0(adc_result1);
ADRESH=0;
Delay10TCYx(1);
}
```

```
/* RANGOS SERVO #0 "BASE"*/
/*Este servomotor solo puede tener movimientos en un rango de 500 a 2500. */
//Selección por encima y debajo de la posición central.
```

```
void SERVO0 (int BASE)
{
if(132<BASE&&BASE<191)
{
Rango0=map(BASE,38,190,500,2500);
if(Rango0<=2500&&Rango0>=500){Move0(Rango0);}
}
if(38<BASE&&BASE<126)
{
Rango0=map(BASE,38,190,500,2500);
if(Rango0<=2500&&Rango0>=500){Move0(Rango0);}
}}
```

```
/*ENVÍO DE LA POSICIÓN DEL SERVO #0 "BASE" POR LA USART*/
```

```
void Move0 (int M0)
{
        unsigned char Ma[13]="\0";
        sprintf(Ma,"#0P%iT1000\r",M0);
        putsUSART(Ma);
        M0=0;
}
```

En el último paso se observa como la trama requerida por la tarjeta SSC32 es combinada con una variable tipo string de 13 caracteres (Ma) la cual contiene la posición para el servomotor.

Estas posiciones se encargan de realizar el desplazamiento de los servos de la base, el hombro, el codo y la muñeca. Sin embargo, el servo para manejar el Gripper tiene un manejo especial ya que se adaptó un sensor de fuerza FSR-01. Este sensor está encargado de regular la presión ejercida por el Gripper. La fuerza a aplicar depende del objeto que se quiere sujetar, es importante elegir un material liviano debido a que el peso máximo soportado por el brazo es de 24Oz.

El código para realizar el movimiento de Gripper es similar al código para el movimiento de los servomotores, pero, en este caso el servomotor no depende de un potenciómetro del joystick sino del sensor FSR01, en la  Figura 46 se muestra un diagrama con el proceso que se lleva a cabo para el movimiento del Gripper.

**Figura 46:** Diagrama de flujo para el movimiento del Gripper.


Por medio del canal análogo de la tarjeta SSC32 es leída la señal del sensor FSR01, esta señal es solicitada por el microcontrolador por medio de una instrucción como esta: putrsUSART("VA\r"); , en este caso solitica la lectura del canal A . Una ves la tarjeta reciba este mensaje realiza la lectura y envia al micro una trama con el valor de la señal en ese instante.

El microcontrolador recibe esta señal y realiza un tratamiento para relacionar el valor del dato de la señal análoga con el movimiento del servomotor, esta relación se ejecuta asi como en el caso de los demás servos con la función MAP.

Finalmente, una vez es tratado el dato se envía la posicion al servomotor para que ejecute el movimiento, entonces, si la fuerza aplicada por el sensor es muy alta y supera el limite aceptado por el objeto el servomotor se detiene, o si por el contrario es muy baja este se cerrará progresivamente.

En conclusión, el servomotor asociado al Gripper tiene un tratamiento basado en la lectura de la señal del sensor FSR01, en este caso se evidencia una comunicación bidireccional entre el microcontrolador y la tarjeta SSC32 a través de los módulos XBee, ya que tanto el micro solicita el valor del sensor como la tarjeta lo envía para que este sea tratado. En la Tabla 15 se describe o se evidencia el código correspondiente al diagrama de la Figura 46.

**Tabla 15:** Código utilizado para él envió de posiciones al servo 4 "Gripper".

```
/*SELECCIÓN DE TECLA "GRIPPER"*/
```

```
if(PINZA==1)
{Sensor();}
```

```
/*FUNCIÓN PRINCIPAL SENSOR*/
/*Este servomotor solo puede tener movimientos en un rango de 500 a 1500. */
```

```
void Sensor(void)
{
unsigned char fsr;
unsigned int val=0;

    do{
    putrsUSART("VA\r");                 //Inicia captura de dato del FSR01
    while (!DataRdyUSART());
    fsr=ReadUSART();                    //se captura el dato en la variable FSR

    val=map(val,0,255,500,1500);
    Delay10KTCYx(255);

if(val<900)
    {
        val=val+100;
        GripperMove(val);
        Delay10KTCYx(255);
    }
    Delay10KTCYx(255);

}while(PINZA==1);

    if(PINZA==0)
    {
        if((val>900)&&(val<=1500))
        {
        val=val-100;
        GripperMove(val);
        Delay10KTCYx(255);
        }
    }
    val=0;
```

```
/*ENVÍO DE LA POSICIÓN DEL SERVO #0 "BASE" POR LA USART*/
```

```
void GripperMove(int var)
{
        unsigned char str[13]="\0";
        sprintf(str,"#4P%iT1000\r",var);
        putsUSART(str);
        var=0;
}
```

Por último, tenemos un brazo completamente comunicado inalámbricamente y enlazado al cerebro de todo el proceso que es el joystick. La forma como se programó o fue diseñado este sistema permite que la persona que lo controle tenga total manejo de cada uno de los servomotores. De esta manera, las rutas para la obtención del objeto son totalmente libres y dependientes de lo que el usuario desea.

# 5. CARACTERÍSTICAS DEL JOYSTICK

Para el desarrollo de este proyecto se utilizó el joystick marca Logitech®, modelo Extreme 3D Plus, debido a sus características ya que tiene una base sólida y estable, una muy buena comodidad de sujeción y por la cantidad de botones programables (12), esta última propiedad permitió el desarrollo de un diseño amplio y con libertad para la realización de diferentes funciones. Sin embargo, este joystick posee una conexión USB lo que genera una mayor dificultad en la lectura de sus elementos, por consiguiente, para mayor claridad se realizó un cambio en esta conexión, es decir, cada elemento del joystick ya sea tecla o potenciómetro fue tomado independientemente, de esta manera se logró discriminar cada elemento de forma eficiente y más importante aún clara y concisa.

En la Figura 47 se muestra el joystick cuando tenía la conexión USB, más adelante en la Figura 48, Figura 49, Figura 50, Figura 51, Figura 52, Figura 53 y Figura 54 se muestra el proceso de adaptación del joystick de conexión USB a conexión independiente de cada uno de sus componentes. Adicionalmente, se diseñó la base para situar el joystick, para conocer las fichas técnicas dirigirse al Anexo 1.
Como se nombró en el capítulo anterior se utilizaron tres potenciómetros y cinco teclas digitales incluyendo el botón de encendido y apagado, ver Tabla 13.

**Figura 47:** Joystick con conexión USB.

**Figura 48:** Identificación e independización de cada una de las teclas de la palanca, vista superior.

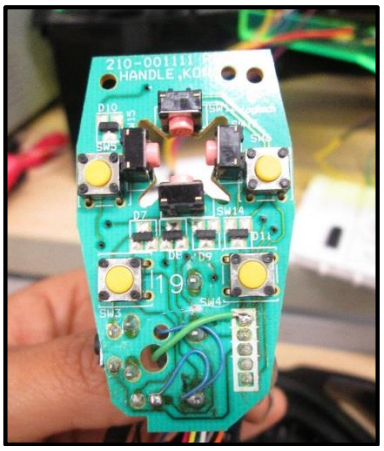


**Figura 49:** Identificación e independización de cada una de las teclas de la palanca vista inferior.

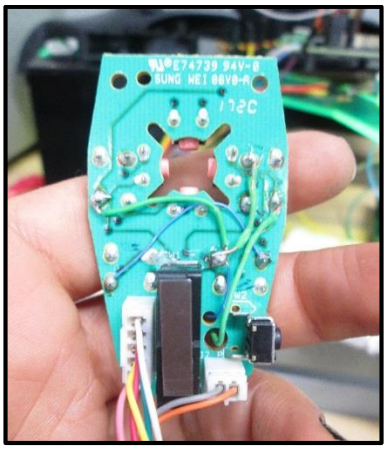


**Figura 50:** Identificación e independización de cada una de las teclas de la base.

**Figura 51:** Identificación e independización de cada uno de los potenciómetros.

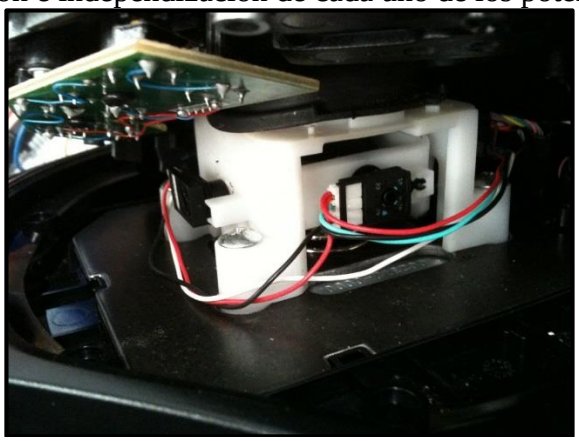


**Figura 52:** Identificación e independización de las teclas de la palanca y el potenciómetro de la muñeca.

**Figura 53:** Salida de cada una de las teclas de base y palanca más los potenciómetros

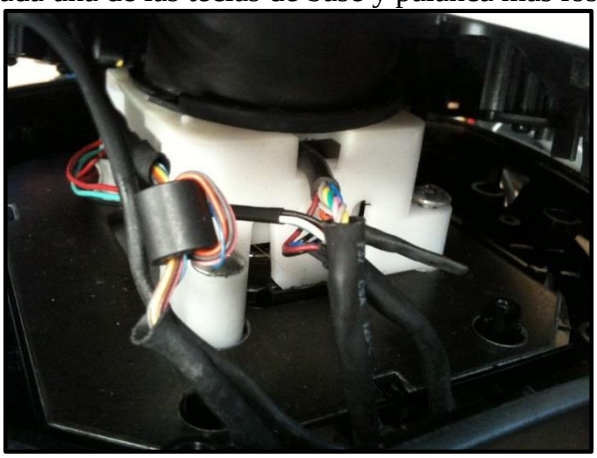


**Figura 54:** Joystick terminado.

## 5.1 INTERFAZ JOYSTICK-MICROCONTROLADOR

Como ya se tienen todos los elementos independientes entonces el microcontrolador recibe dos tipos de señal desde el joystick una análoga y otra digital, la análoga son los tres potenciómetros, uno lateral otro frontal y el movimiento de muñeca y las señales digitales que son los botones de encendido-apagado más los que son utilizados para la selección de un servo especifico. Es decir, debido a que el brazo robótico tiene 5 (GDL) y solo se cuenta con tres potenciómetros, entonces es necesario realizar una combinación potenciómetro- botón para el control de los servos, tal como se muestra en la Tabla 13. En la Figura 55 se muestran los botones seleccionados para el movimiento del brazo.

**Figura 55:** Botones utilizados para el movimiento del brazo.

Cada una de estas teclas debe controlar un servomotor del brazo, entonces al presionarse una de estas y accionar la palanca ya sea hacia adelante, atrás o ejecutarse un movimiento de muñeca el servomotor se desplaza a esta posición, para esto es necesario hacer un manejo especial tanto con las señales análogas como digitales, así como se muestra a continuación.

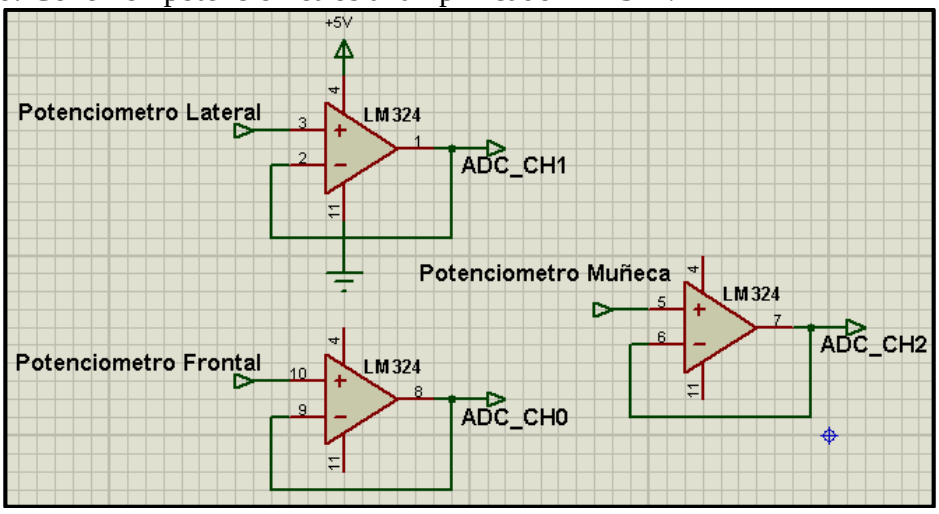
### 5.1.1 Manejo de las señales análogas.

Para el manejo de las señales de los potenciómetros se realizaron dos etapas:

- Toma de la señal con el amplificador operacional LM324 en configuración seguidor de tensión, esta configuración permite reducir la impedancia de los potenciómetros sin alterar la tensión. Este proceso se realizó debido a que los canales análogos del microcontrolador solo permiten una impedancia recomendada de 2.5kΩ y los potenciómetros tienen una impedancia mayor a este valor. En la Figura 56 se muestra el diagrama de conexión de los potenciómetros al amplificador operacional[4].

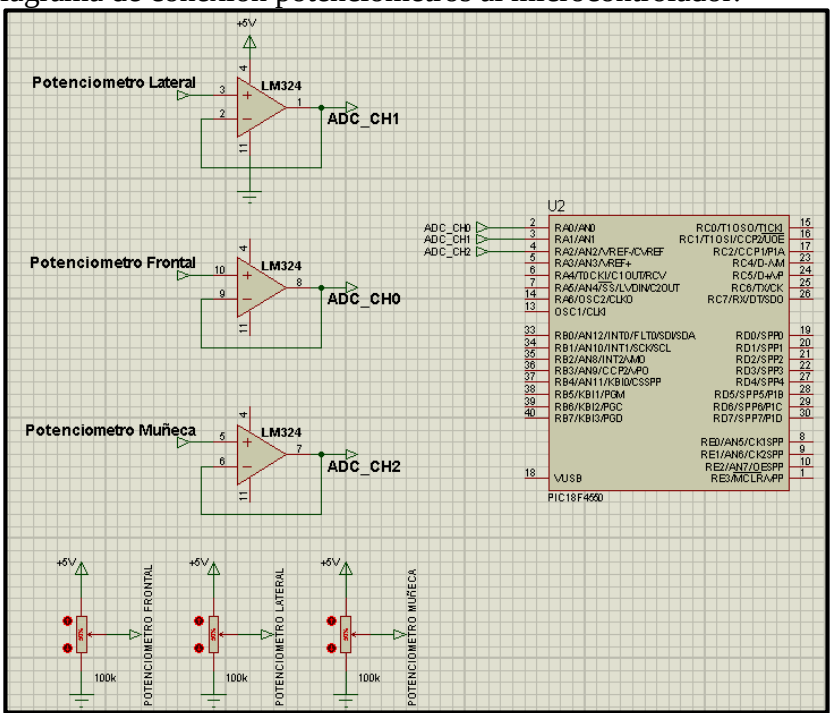
- Toma de la señal de salida del amplificador operacional en los canales análogos del microcontrolador, en esta etapa se configura por software todas las características necesarias para realizar una conversión análogo-digital de 8 bits, en la Figura 57 se muestra la conexión de los potenciómetros al microcontrolador.

**Figura 56:** Conexión potenciómetros al amplificador LM324.




[4] La simulación de esta configuración se desarrolló en el software Proteus

**Figura 57:** Diagrama de conexión potenciómetros al microcontrolador.




La configuración del módulo análogo en el software se realizó de la siguiente manera (Tabla 16):

**Tabla 16:** Código utilizado para la configuración del módulo A/D.

```
/*ADC CONFIG*/
```

```
ADCON1=0b00001011;
ADCON2=0b00011010;
ADCON0=0b00000000;
ADCON0bits.ADON = 0x01;    //Enable A/D module
ADRESH=0;
```

```
//  Canales AN0,AN1,AN2,AN3
//  ADCON1:|0|0|VSS(0)|VDD(0)|AN3(1)|AN2(0)|AN1(1)|AN0(1)|
//  VSS,VDD 5v / 2^8 = 20mV por numero osea que para 100 = 20mV*1.953125~1.96
//  ADCON2: ADFM Left justified(0)|-|6 TAD (0)|6 TAD(1)|6 TAD
(1)|FOSC32(0)|FOSC32(1)|FOSC32(0)|

    /*  . . .ADRESH . . : . . ADRESL. . .
        7 6 5 4 3 2 1 0 : 7 6 5 4 3 2 1 0
        X X X X X X X X . X X . . . . . . <-Left Justified
        . . . . . . X X . X X X X X X X X <-Right Justified
     */

//|-|-|CHS3(0)|CHS2(0)| CHS1 (0)| CHS0(0)|     GO/DONE | ADON (0) AD DISABLED.
//when adon=1 , go_done 1 = A/D conversion in progress 0 = A/D Idle.
```

En síntesis, se habilitan cuatro canales análogos AN0, AN1, AN2 y AN3, se realiza una conversión de 8 bits, no se utilizan tensiones de referencia diferentes a la de alimentación del microcontrolador, la resolución de la conversión es de 20mV por número y se elige justificación a la izquierda, por lo tanto, el dato de la conversión es guardado en el registro ADRESH.

### 5.1.2 Manejo de las señales digitales.

Como se observa en la Figura 55 se utilizaron 7 teclas digitales activación, desactivación, botón1, botón2, botón3, botón4 y gatillo las cuales tomaron el nombre de: ON, OFF, SERV0, SERV1, SERV2, SERV3 y GATILLO respectivamente en la Figura 58 se muestra el diagrama de conexión de estas teclas. El manejo de estas señales digitales se realizó de dos formas:

- Teclas de mayor prioridad activación y desactivación, se trataron como interrupciones de prioridad alta, esto es, aun cuando se esté llevando a cabo un proceso si se presiona una de las teclas el sistema se detiene y realiza o la activación o desactivación.

- Teclas de prioridad baja SERV0, SERV1, SERV2, SERV3 y GATILLO, se trataron como entradas digitales normales en el puerto b y c, las cuales son leídas durante el programa principal y en caso de ser presionadas se ejecuta su función.

La interrupción de prioridad alta se llevó a cabo mediante la interrupción externa INT0, en la Tabla 17 se muestra la configuración del módulo de interrupciones y en la Tabla 18 se evidencia una parte del código para la ejecución de esta interrupción.

**Tabla 17:** Código utilizado para la configuración del módulo de interrupciones.

```
/*INTERRUPT SETTINGS*/
```

```
INTCON =0b10010000;
INTCON2=0b11000000;
INTCON3=0b00000000;
INTCONbits.GIEH=1;
INTCONbits.RBIF=0;
INTCONbits.RBIE=0;
```

```
/*INTCON---->X X X X X X X X              |*INTCON2---->X X X X X X X X
 * --------->7 6 5 4 3 2 1 0              |* ---------->7 6 5 4 3 2 1 0
 * 7-> Global Interrupt GIEH=1            |* 7-> PORTB Pull Up Enable=1
 * 6-> Peripheral Interrupt PEIE=0        |* 6-> External Interrupt 0 Edge=1
 * 5-> TMR0IE Overflow Interrupt=0        |* 5-> External Interrupt 1 Edge=0
 * 4-> INT0IE External Interrupt=1        |* 4-> External Interrupt 2 Edge=0
 * 3-> RBIE Port Change Interrupt=0       |* 3-> Unimplemented=0
 * 2-> TMR0IF Overflow Flag=0             |* 2-> TMR0IP Overflow Interrupt=0
 * 1-> INT0IF External Interrupt Flag=0   |* 1-> Unimplemented=0
 * 0-> RBIF Port Interrupt Flag=0         |* 0-> RBIP Port Change Interrupt=0
     ***************************************************************************
* INTCONbits.RBIF=0;->
* INTCONbits.RBIE=0;->
* INTCON3---->INT1, INT2 ENABLE, PRIORITY AND FLAG.*/
```

**Tabla 18:** Código utilizado para la interrupción.

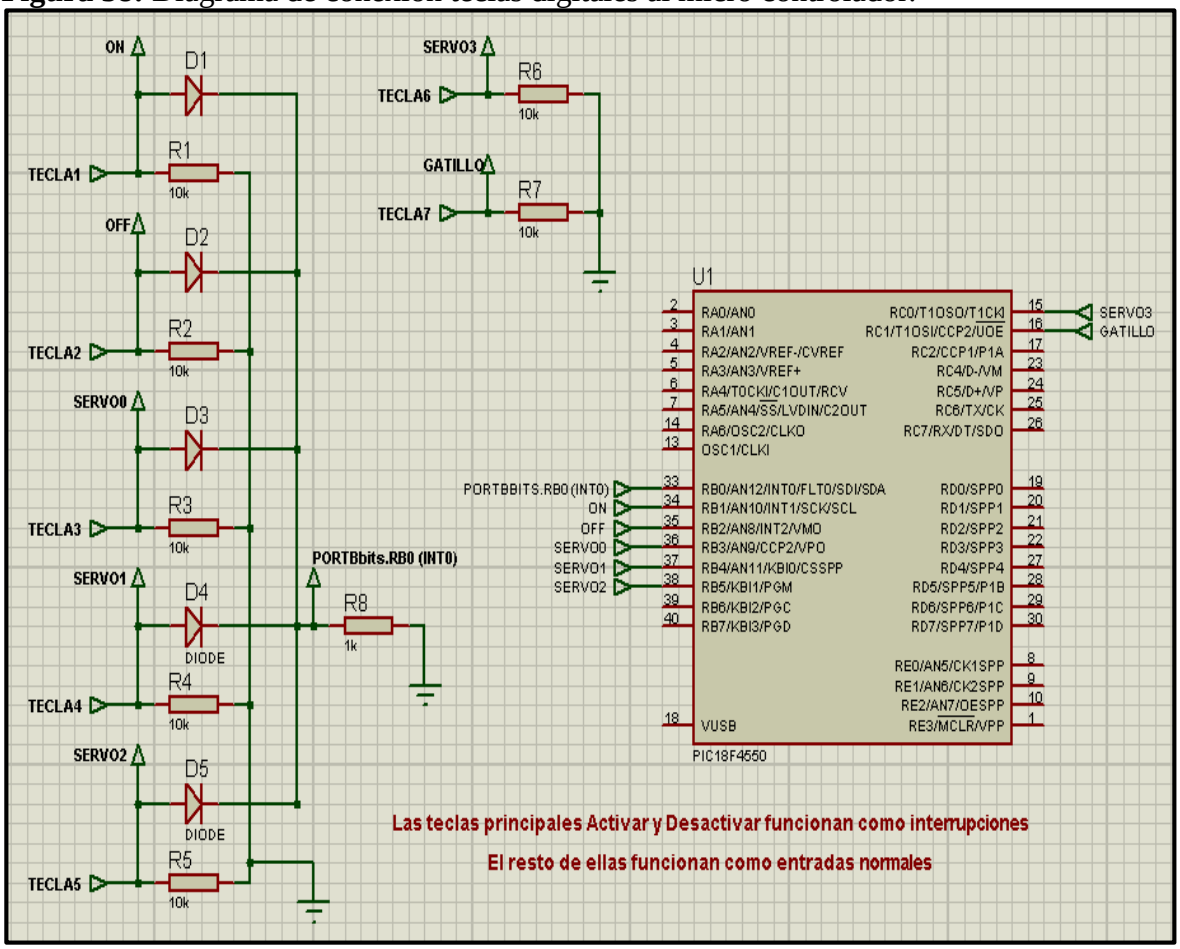
```
//HIGH INTERRUPTS
//*******************************************************
#pragma code high_vector=0x08          //HIGH PRIORITY
void high_interrupt (void)
{
 _asm GOTO high _endasm
}
#pragma code
```

```
//FUNCION 1: TECLAS DIGITALES----ACTIVAR; DESACTIVAR FLAG INT0
//*****************************************************
#pragma interrupt high
void high (void)
{
if (INTCONbits.INT0IF==1)               //Se activó la interrupción ?
    Time1;
{
    x=PORTB&0b00011111;                 //Máscara (Activar-Desactivar)

    switch (x)
    {
      case 3:
      Activate();                      //ON
      break;

      case 5:
      Desactivate();                   //OFF
      break;
     }
INTCONbits.INT0IF=0;                    //Se limpia la bandera INTIF
}}
```

**Figura 58:** Diagrama de conexión teclas digitales al micro controlador.




En el caso de las teclas relacionadas a los servos se deben ejecutar en el programa principal continuamente ya que a partir del accionamiento de una de ellas se debe realizar una lectura del canal análogo (potenciometros) y esta lectura debe ser continua debido al cambio permanente de posicion de la palanca. En la Tabla 19 se muestra esta parte del codigo con la lectura de cada una de las teclas y la relacion con los servomotores.

**Tabla 19:** Código utilizado para la el manejo de las teclas digitales del joystick.

```
//PROGRAMA PRINCIPAL (MAIN)
//*********************************************************
void main(void)
{
InitPorts();              //Configura Dispositivos Micro {USART,TRIS,ADC,PWM}
PORTB=0;                  //Inicializa el PORTB en 0.
INICIO();                 //Incio movimiento de brazo en reposo

while(ON==0)             //Mientras no se presione la tecla ON el sistema
    {                    //permanece en SLEEP MODE.
    Sleep();
    Nop();
    Nop();
    }

while(1)
{
    blinker();                //Parpadeo indicador de funcionamiento

    /*INICIO DE MOVIMIENTOS*/

    if(SERV0==1)              //Función movimiento "BASE"
    {do{AdcConvert0();}
    while(SERV0==1);}

    if(SERV1==1)              //Función movimiento "HOMBRO"
    {do{AdcConvert1();}
    while(SERV1==1);}

    if(SERV2==1)              //Función movimiento "CODO"
    {do{AdcConvert2();}
    while(SERV2==1);}

    if(SERV3==1)             //Función movimiento "MUÑECA"
    {do{AdcConvert3();}
    while(SERV3==1);}

    if(PINZA==1)             //Función movimiento "GRIPPER"
    {do{Sensor();}
    while(PINZA==1);}

}}
```

Como se puede observar en la Tabla 19 cada tecla llama una función identificada como *AdcConvert(),* esta función es la encargada de realizar la lectura del canal análogo y así leer el potenciómetro adecuado, adecuado se refiere al canal que se eligió para el manejo de ese servo, como ejemplo en la Tabla 20 se muestra que cuando se presiona la tecla SERV0 la palanca debe desplazarse hacia la izquierda o derecha para que el servo ejecute el movimiento, para conocer la relación entre cada tecla y el movimiento de la palanca ver la Tabla 13.

**Tabla 20:** Relación entre una tecla digital y un canal análogo.

```
//SELECCIÓN DE TECLA SERVO0
```

```
if(SERV0==1)              //Función movimiento "BASE"
    {do{AdcConvert0();}
    while(SERV0==1);}
```

```
/*************************************************************
/*MOVIMIENTO DEL SERVO #0 "BASE" EJECUTADO POR EL MOVIMIENTO DEL POTENCIÓMETRO
LATERAL*/
```

```
VOID ADCCONVERT0(VOID)
{
    ADCON0BITS.ADON=0X01;   //ENABLE A/D MODULE
    SETCHANADC(ADC_CH1);    //LATERAL
    DELAY10TCYX(1);
    CONVERTADC();
    WHILE(BUSYADC()==1){}
    ADC_RESULT1 = ADRESH;
    SERVO0(ADC_RESULT1);   //EL DATO ES ENVIADO A LA FUNCIÓN DE RANGOS Y
    ADRESH=0;              //EJECUCIÓN DE MOVIMIENTOS.
    DELAY10TCYX(1);
   }
```

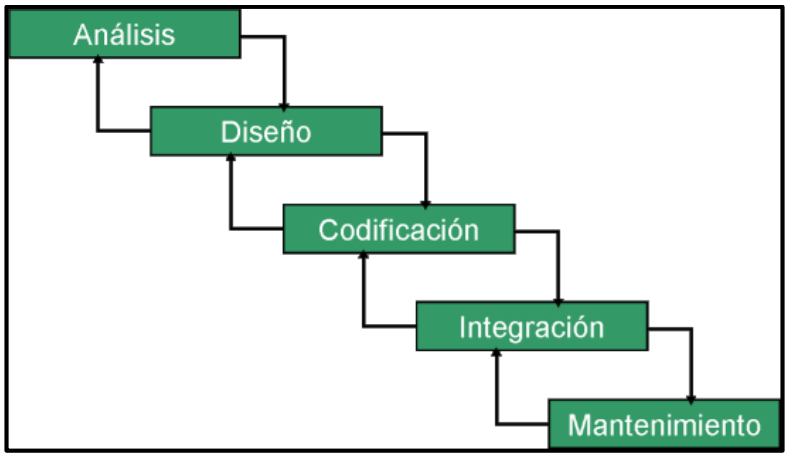
Finalmente, tenemos un sistema con señales análogas y digitales que relacionadas entre sí permiten un movimiento controlado de cada uno de los servomotores. Este sistema es limpio y confiable debido a que se programaron las posiciones máximas para cada uno de los servomotores, de esta manera sin importar si se adelanta más o menos la palanca el servo no se desplazara más allá de su límite.

# 6. IMPLEMENTACIÓN DE SOFTWARE Y LENGUAJE DE PROGRAMACIÓN

A la hora de ejercer actividades de desarrollo de software, se requiere fomentar tareas que vallan a la par con los avances del proyecto tanto en la escritura de código como en su implementación y etapas de prueba, cuantificar, priorizar, ejecutar son algunas de ellas.

El campo encargado de estudiar los principios y metodologías para el diseño, desarrollo, análisis, mantenimiento de software es conocido como Ingeniería de Software enmarcado en un ciclo de vida útil como se muestra a continuación ver Figura 59.

**Figura 59:** Desarrollo del Ciclo de Vida Útil.




Tomado de [25]

Para llevar a cabo el proceso de codificación se decidió utilizar un IDE (*integrated development environment*) que ya tuviera las herramientas fuertemente ligadas para el desarrollo con microcontroladores (Editor, Compilador, Debuger, Linker)

## 6.1 MPLAB X

Al momento de llevar a cabo el desarrollo de este proyecto, se decidió implementar las herramientas de desarrollo propias por la empresa fabricante de micro controladores Microchip ©, además de ello se seleccionó el compilador C18 en su versión 3.37, en la Figura 60 se puede observar el entorno de este software y en la Figura 61 se muestra un ejemplo de un call graph para un proyecto.

Algunos de los criterios para hacer esto fueron [26]:

- Soporte *call graph* para proyecto de gran envergadura.
- Opción para alojar múltiples proyectos.
- Integración de Diferentes versiones de compiladores.
- Soporte para parseo de sintaxis.
- Sangría y autocompletado.

- Facilidad para la creación de plantillas de código.
- Herramienta de historial local.
- Facilidad de integración con diferentes cvs[5].
- Ventana de tareas pendientes.

**Figura 60:** Entorno MPLABX

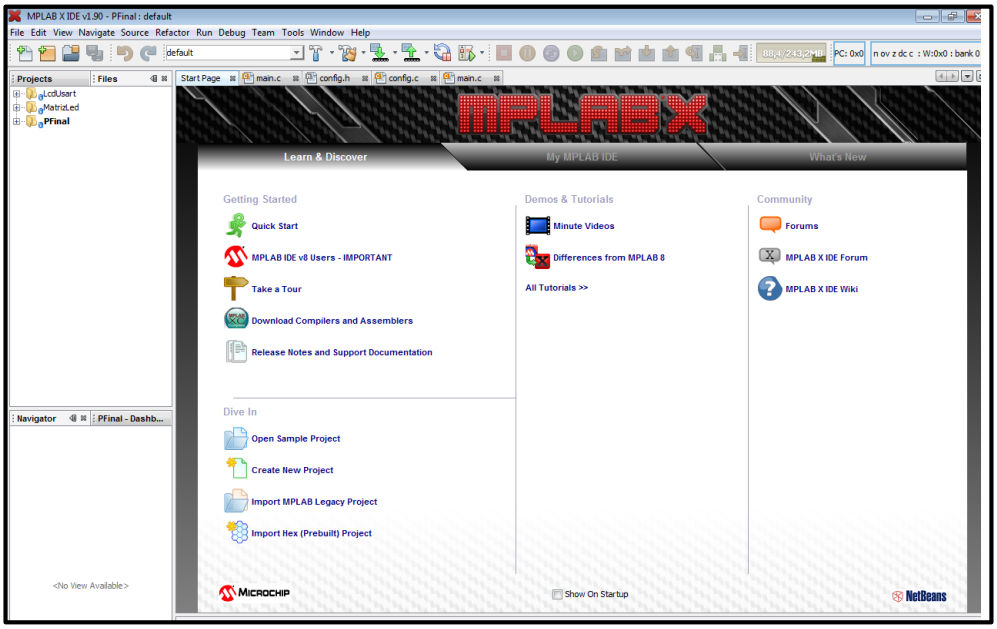
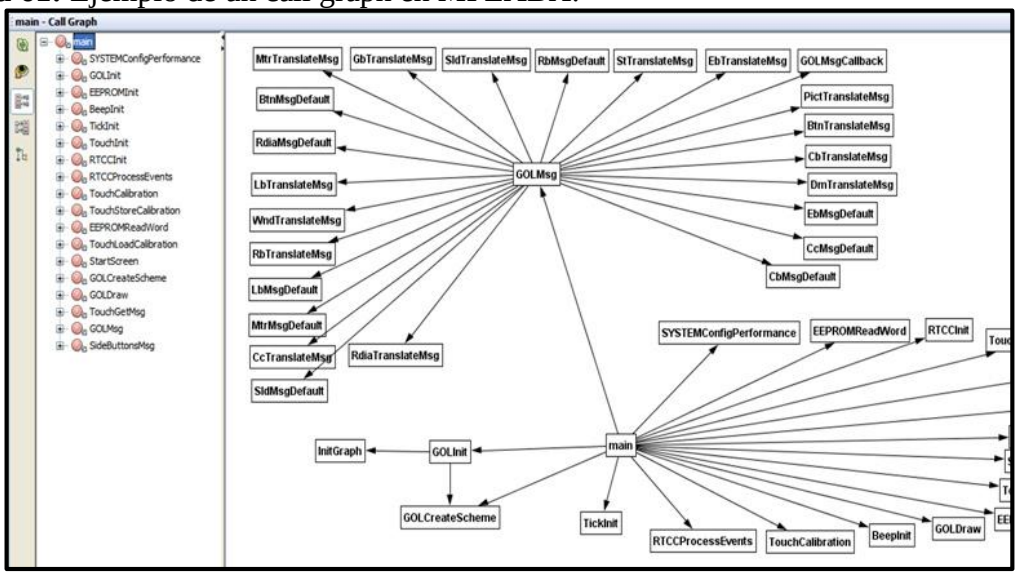



**Figura 61:** Ejemplo de un call graph en MPLABX.




[5] *Concurrent Versioning System*, Sistema de control de versiones, este mantiene el registro de todo el trabajo y los cambios en los archivos del proyecto, algunos son git, cvs, subversión.

Este IDE es basado en un proyecto open source desarrollado por Oracle llamado Netbeans IDE[6], con gran soporte comunitario, además de ello tiene una facilidad para la configuración del workspace como pueden ser los temas y colores, manejo de múltiples proyectos entre otros.

## 6.2 LENGUAJE C

El lenguaje C está constituido por tres elementos: el **compilador**, el **preprocesador** y la **librería estándar**. A continuación se explica brevemente en qué consiste cada uno de estos elementos. [27]

### 6.2.1 COMPILADOR

El compilador es el elemento más característico del lenguaje C. Su misión consiste en traducir a lenguaje de máquina el programa C contenido en uno o más ficheros fuente. El compilador es capaz de detectar ciertos errores durante el proceso de compilación, enviando al usuario el correspondiente mensaje de error. [27]

Seleccionamos el compilador C18 en su versión 3.40, por las siguientes razones:

- Mejor Integración con herramientas.
- Optimización para uso de memoria.
- Alta compatibilidad con ANSI C y portabilidad de código.
- Soporte oficial y de la comunidad.
- Librerías de código abierto.
- Versión de prueba para estudiantes.
- Portabilidad de código a otros a micros de gama media.
- Recursividad, entre overlays

### 6.2.2 PREPROCESADOR

El preprocesador es un componente característico de C, que no existe en otros lenguajes de programación. El preprocesador actúa sobre el programa fuente, antes de que empiece la compilación propiamente dicha, para realizar ciertas operaciones. Una de estas operaciones es, por ejemplo, la sustitución de *constantes simbólicas*. Así, es posible que un programa haga uso repetidas veces del valor 3.141592654, correspondiente al número pi. Es posible definir una constante simbólica llamada PI que se define como 3.141592654 al comienzo del programa y se introduce luego en el código cada vez que hace falta. En realidad PI no es una variable con un determinado valor: el preprocesador chequea todo el programa antes de comenzar la compilación y sustituye el texto PI por el texto 3.141592654 cada vez que lo encuentra. Las constantes simbólicas suelen escribirse completamente con mayúsculas, para distinguirlas de las variables. [27]

[6] https://netbeans.org/, NetBeans IDE is FREE, open source, and has a worldwide community of users and developers.

### 6.2.3 LIBRERÍA ESTÁNDAR

Con objeto de mantener el lenguaje lo más sencillo posible, muchas sentencias que existen en otros lenguajes, no tienen su correspondiente contrapartida en C. Por ejemplo, en C no hay sentencias para entrada y salida de datos. Es evidente, sin embargo, que ésta es una funcionalidad que hay que cubrir de alguna manera. El lenguaje C lo hace por medio de ***funciones*** pre programadas que se venden o se entregan junto con el compilador. Estas funciones *están agrupadas en un conjunto de librerías de código objeto*, que constituyen la llamada ***librería estándar*** del lenguaje. La llamada a dichas funciones se hace como a otras funciones cualesquiera, y *deben ser declaradas* antes de ser llamadas por el programa (se hace esto por medio de la directiva del preprocesador **#include**). [27]

## 6.3 LENGUAJE TARJETA SSC-32

El SSC-32 (controlador serie de servo) es un controlador de servos con electrónica pre-montada con algunas importantes características ver Tabla 21. Tiene una alta resolución (1uS) para el posicionamiento exacto, y los movimientos sumamente lisos. La gama es 0.50mS a 2.50mS para una gama de movimiento de aproximadamente 180°. El comando del movimiento puede tener inmediata respuesta, la velocidad controlada, el movimiento cronometrado, o una combinación de estos todos. [8]

**Tabla 21:** Características tarjeta SSC32

| ÍTEM | CARACTERÍSTICA |
|---|---|
| **Microcontrolador** | Atmel ATMEGA8-16PI |
| **EEPROM** | 24LC32P (No es soportado en esto modelo actual) |
| **Velocidad** | 14,75 MHz |
| **Secuenciador Interno** | Hexápodo de 12 Servos (Tripodo Alternante) |
| **Entrada Serie** | Verdadero RS-232 o TTL, 2400, 9600, 38,4k, 115,2k, N81 |
| **Salidas** | 32 (Servo o TTL) |
| **Entradas** | 4 (Estáticas o Latching) |
| **Requisitos de Corriente** | 31mA |
| **Interface del Ordenador** | DB9F |
| **Interface del Microcontrolador** | Enchufe tipo Header |
| **Control de Servos** | Hasta 32 servos plug-in directos |
| **Tipo de Servos Soportados** | Futaba o Hitec |
| **Percurso del Servo** | ~170° |
| **Resolución del Servo** | 1uS, .09° |
| **Resolución de la velocidad del Servo** | 1uS / Seg |
| **Control del movimiento del Servo** | Inmediato, Cronometrado, Acelerado o Sincronizado |
| **Dimensiones de la Tarjeta** | 7,62 x 5,84 cm |

Para el envío de un dato a la tarjeta SSC32 se debe tener en cuenta un formato específico el cual se puede observar en la Figura 62, con este formato la tarjeta identifica el servo, la posición enviada, el tiempo, la velocidad. Adicionalmente, es importante que al final de cada

trama se envíe el carácter de retorno <cr> que en C se escribe "\r", sin este carácter la tarjeta no ejecuta la instrucción.

**Figura 62:** Trama para el envío de movimientos a los servomotores.

| # <ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr> | |
|---|---|
| <ch> | Número de canal en formato decimal, 0-31 |
| <pw> | Ancho de pulso en microsegundos, 500-2500 |
| <spd> | Velocidad de movimiento en uS por segundos para cada canal (opcional) |
| <time> | Tiempo en mS para el movimiento completo, que afecta a todos a los canales, 65535 máx (opcional) |
| <cr> | Carácter de retorno de carro, ASCII 13 (necesario para iniciar la acción) |
| <esc> | Cancela la acción actual, ASCII 27 |

Tomado de [8].

### 6.4 PICKIT 3

El PICKit 3 es la herramienta de depuración y programación más sencilla y de menor coste de Microchip. Está totalmente soportado por el MPLABX IDE, y tiene una sencilla conexión USB *Full Speed* que permite programar y depurar, también es posible actualizar el firmware interno del PICKIT3. Tiene circuitos para protección de sobre tensión y de corto circuito, permite ejecución en tiempo real y soporta tensiones desde 2.0V. [7]Respectando la norma USB puede dar 100mA al circuito donde está conectado y tiene Led's de información rápida para el usuario hasta 5.5V, ver Figura 63. [28]

**Figura 63:** PICKit 3 Programador estándar, Microchip


Tomado de [28].

PICKit 3 puede trabajar con micros PIC de 8, 16-bit y 32 Bits. Algunas de las funciones que se pueden observar en el PICkit3 son:

[7] Producto PICkit3 descripción , http://www.sagitron.com/productos/activos/microchip/203-pickit3-programador-de-pic-sin-necesidad-de-ordenador

- Halt.
- Single step.
- Breakpoint sobre datos y direcciones.
- Stopwatch para mediciones y pruebas sobre la tarjeta.
- Análisis de Registros especiales, Variables, áreas de memorias.

Estos también poseen una nueva tecnológica conocida como “Power-To-Go”que da la posibilidad de una forma muy sencilla programar cualquier microcontrolador de las familias PIC12, PIC16, PIC18, PIC24, dsPIC33F y PIC32 de Microchip sin necesitar de un computador, este solo permite guardar un código de hasta 512KB en su Flash. [28]

En el Anexo2 se pueden evidenciar los diagramas de flujo y los call graph para este proyecto.

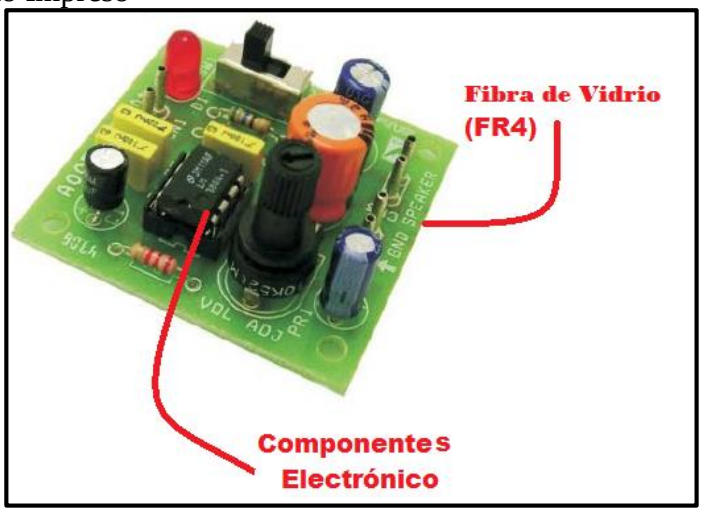
# 7. DISEÑO ELECTRÓNICO Y CIRCUITOS IMPRESOS

El diseño y desarrollo de circuitos impresos hoy en día es una de las etapas más importantes a la hora de realizar productos electrónicos, es una de las partes fundamentales de todo prototipo, pues se solucionan problemas de conectividad y malas conexiones que se podrían presentar en las protoboards (*BreadBoards*) es por ello se realizaron los circuitos impresos (*Printed Circuit Board*) para este proyecto y se implementaron varias técnicas descritas a continuación.

## 7.1 CIRCUITOS IMPRESOS

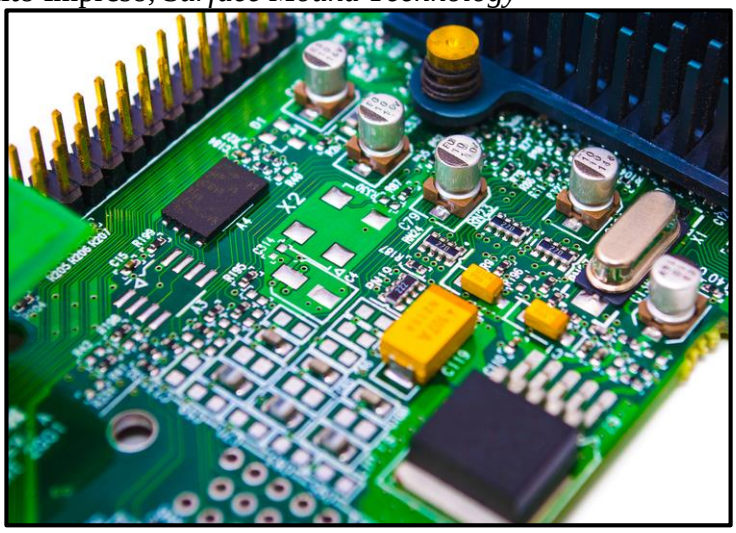
Un circuito impreso, ó PCB (del inglés *printed circuit board*), es una superficie constituida por dos caras una superior y otra inferior compuestas por caminos o pistas de material conductor típicamente platino, cobre, oro entre otros laminadas sobre una base que puede ser de material de fibra de vidrio, resina, cerámica, plástico, teflón o algún plástico resistentes a los ácidos férricos. El circuito impreso se utiliza para conectar eléctricamente a través de las pistas conductoras, y sostener mecánicamente, un conjunto de componentes electrónicos como en la Figura 64, en la Figura 65 se puede apreciar un ejemplo de un circuito impreso con montaje superficial.

**Figura 64:** Circuito Impreso




Tomado de [29]

**Figura 65:** Circuito Impreso, *Surface Mound Technology*



Tomado de [30].

Existen Actualmente organizaciones que poseen estándares de diseño para los circuitos impresos como La Organización IPC (*Institute for Printed Circuits*), que ha generado un conjunto de estándares que regulan el diseño, ensamblado y control de calidad de los circuitos impresos[8], para el caso de nuestros diseño hemos realizado los impresos utilizando uno de los estándares más conocidos en la industria el IPC2221[9].

### 7.1.1 Calculo IPC2221

Para realizar el ancho de una determinada pista o camino es necesario conocer tres datos como mínimo:

- Corriente Máxima a Circular por pista (***Imax***)[Amperios ]
- Incremento Máximo de Temperatura (***DTmax***) [°C]
- Grosor de Pista (***G***)[Onzas/Pie]

Cabe mencionar que ***G*** no se debe confundir con el ancho de pista, el grosor de pista se refiere a la altura de la pista referente al material que sirve de base en la placa de circuito impreso.

Para el cálculo se establecen los siguientes parámetros:

*Imax*=60mA
*DTmax*=10°C entiéndase como Temperatura ambiente 25 °C.
*G*=1 [Onza/pie]

[8] Printed Circuit Boards The basics, Sparkfun https://learn.sparkfun.com/tutorials/pcb-basics
[9] Association Connecting Electronics Industries, http://www.ipc.org/default.aspx

La fórmula a aplicar según el estándar para el ancho de una pista se calcula con la siguiente formula (14):

$$\boldsymbol{Ancho} = \frac{\boldsymbol{Area}}{\boldsymbol{G} * \mathbf{1.378}}[\mathbf{mils} * \mathbf{mils}] \quad \textbf{(14)}$$

Donde un mils es una milésima parte de una pulgada, evidentemente entre los datos necesarios para realizar el cálculo no figuraba el área, se debe utilizar la siguiente ecuación (15):

$$\acute{\boldsymbol{A}}\boldsymbol{rea} = \left[\frac{\boldsymbol{I}}{\boldsymbol{k1} * \boldsymbol{DTmaxDTmax^{k2}}}\right] * \mathbf{1}^{\left(\frac{\mathbf{1}}{\boldsymbol{k3}}\right)}[\text{mils} * \text{mils}] \quad \textbf{(15)}$$

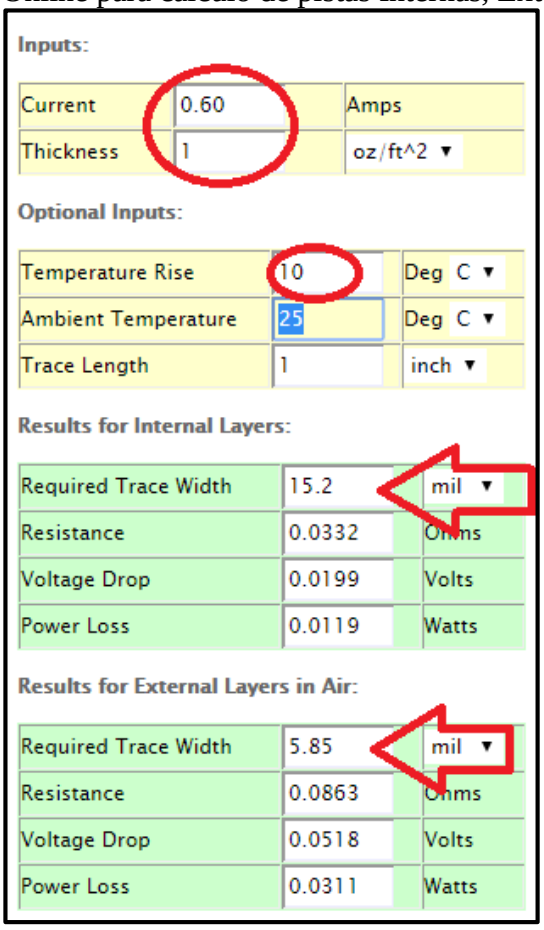
Donde *{k1... k3}* son constantes definidas por el estándar que estamos aplicando.
k1 =0.0150 cuando nuestra pista es interna
k2= 0.5453 cuando nuestra pista es interna ó 0.4281 cuando la pista es externa.
k3= 0.7349 cuando nuestra pista es interna ó 0.6732 cuando la pista es externa.

Ahora si se sustituye (14) en (15) se obtiene:

$$\boldsymbol{Ancho} = \frac{\left\{\left[\frac{\boldsymbol{I}}{\boldsymbol{k1} * \boldsymbol{DTmaxDTmax^{k2}}}\right] * \mathbf{1}^{\left(\frac{\mathbf{1}}{\boldsymbol{k3}}\right)}\right\}}{\boldsymbol{G} * \mathbf{1.378}[\text{mils} * \text{mils}]} \quad \textbf{(16)}$$

Existen algunas herramientas que facilitan más este proceso como es el caso de la calculadora de pistas online, en la cual solo se requiere ingresar los parámetros deseados, así como sse muestra en la Figura 66.

**Figura 66:** Herramienta Online para cálculo de pistas Internas, Externas.




Los valores resaltados en el círculo de color rojo representan los tres requisitos necesarios en orden de aplicar las ecuaciones mencionadas anteriormente, las flechas indican el resultado del grosor de pistas a aplicar.

## 7.2 SOFTWARE DE DISEÑO

Para el diseño de circuitos impresos se suele utilizar típicamente software del tipo CAD (*Computer Aided Design* ) ó CADD (*Computer-Aided Design And Drafting*) pues software de este tipo tiene un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos, diseñadores y todos aquellos que requieran precisión en el modelamiento geométrico de los componentes.

Algunas de las recomendaciones más conocidas a la hora de utilizar un software de este tipo son:

- **Soporte Comunitario:**

Se refiere al uso extendido de dicho software, entre más personas lo utilicen existirán más mejoras en él, ejemplo de ello es la existencia de más componentes, fácil detección de bugs, fácil detección de errores típicos de usuario, mejora y extensibilidad en las herramientas del software.

- **Fácil Utilización:**

Si es fácil de utilizar será mucho mejor para todos, la simplicidad en las herramientas mismas del software permiten que haya un uso mayor.

- **Capacidad:**

Algunos programas de este tipo ponen limitaciones en los diseño, como por ejemplo número de capas, número de componentes electrónicos, dimensiones del circuito impreso.

- **Portabilidad:**

Se refiere a la facilidad de que el software provea la opción de exportar el trabajo realizado a otras aplicaciones, también se refiere a el uso en diferentes sistemas operativos como puede ser Mac OS, Microsoft Windows, Gnu/Linux, Unix.

Para el desarrollo de los circuitos impresos se tuvo en cuenta las recomendaciones anteriores por ello se seleccionó **Eagle**[10] de la empresa **CADSOFT**[11] una empresa de los Estados Unidos. Particularmente ellos tiene una versión freeware que permite tranquilamente realizar los diseños de este prototipo, al utilizar una versión freeware se presentan algunas limitaciones las cuales son:

**Número de Capas**: 2
**Dimensiones**: 100 mm *80mm
**Esquemáticos**: 1

Inicialmente para el desarrollo del circuito se debe realizar el diseño del circuito esquemático agregando el componente y teniendo en cuenta que sus dimensiones concuerden con el tamaño de los objetos físicos como resistencia, diodos, pulsadores...etc.

Una vez se tiene el diseño del circuito esquemático se pasa a la vista de *board* que es donde se realiza el ruteo o trazado de las pistas, la facilidad de este software es que permite observar con claridad las conexiones entre componentes, además de ello utilizar la optimización de conexiones entre ellas conocida como ratnest.

En la Figura 67, Figura 68 y Figura 69 se puede observar el circuito esquemático, la vista de Board y el screen con todos los elementos del impreso principal de todo el sistema. Para ver los circuitos impresos de las fuentes del joystick y el brazo ver el Anexo 3.

---

[10] Para más información sobre este software visitar la página web , www.cadsoftusa.com
[11] La versión freeware se puede encontrar en el siguiente enlace: http://www.cadsoftusa.com/download-eagle/freeware/

**Figura 67:** Vista de Esquemático, Placa Principal Eagle Cadsoft.

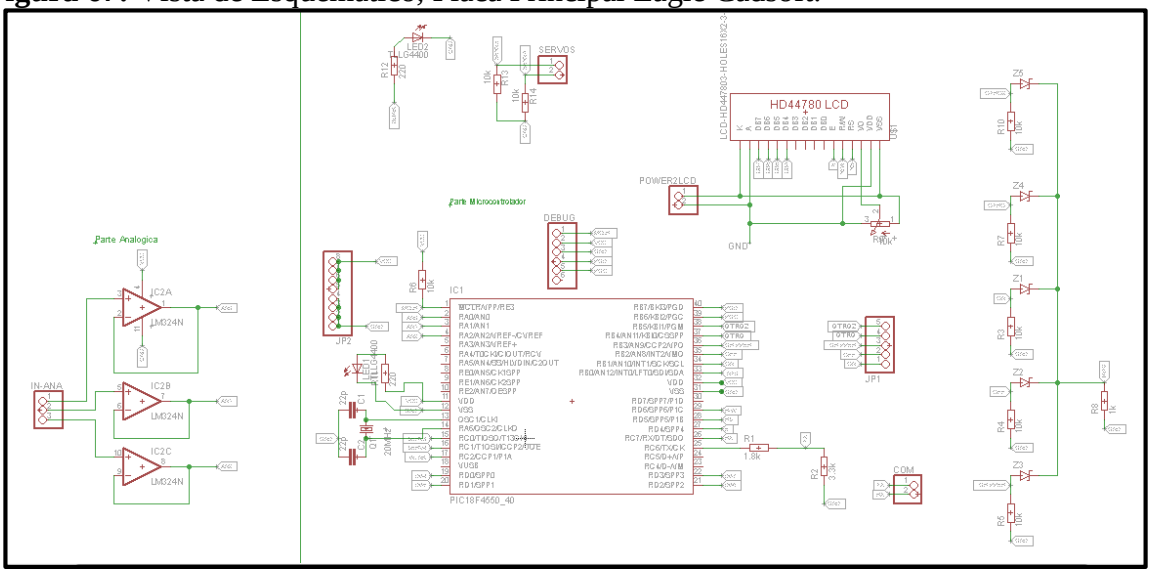



**Figura 68:** Vista de Board Placa Principal, Eagle Cadsoft.

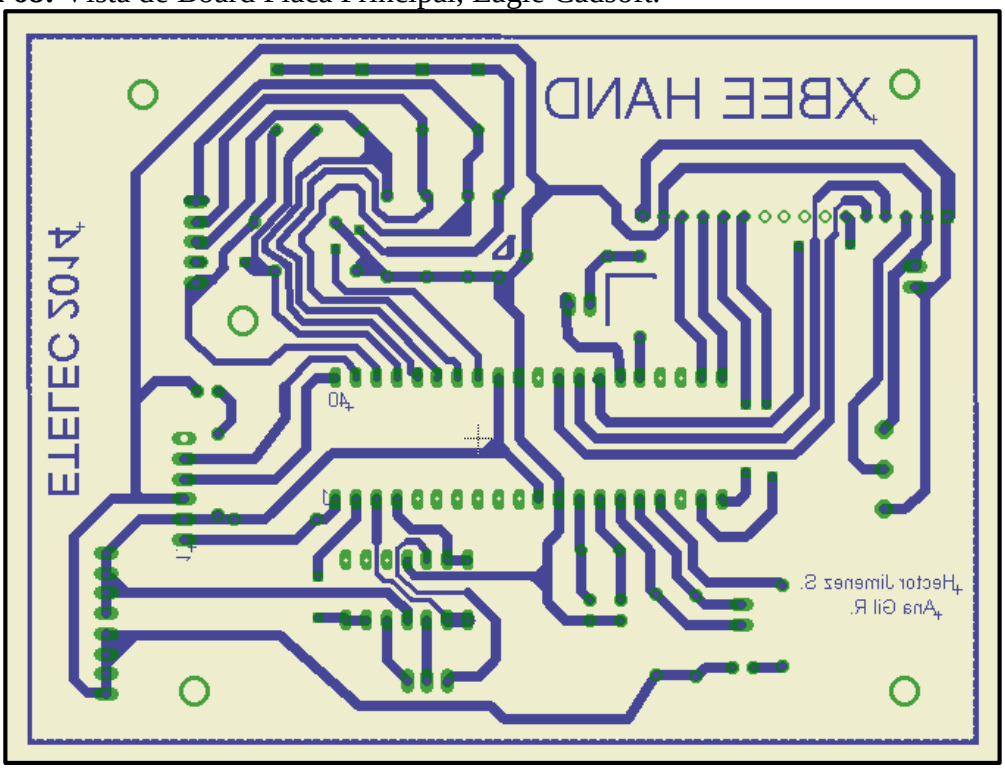
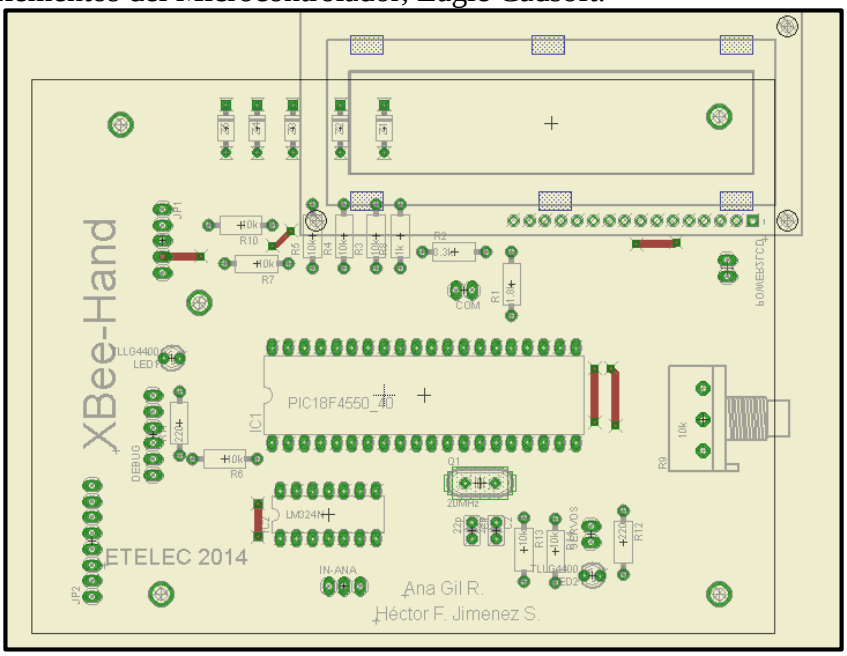


**Figura 69:** Elementos del Microcontrolador, Eagle Cadsoft.




## 7.3 REALIZACIÓN DE CIRCUITOS IMPRESOS

La realización de los circuitos impresos puede ser de tipo artesanal, semi profesional, profesionales. Para este prototipo se realiza pruebas de manera artesanal.

### 7.3.1 Método Artesanal:

El método artesanal para la fabricación de circuitos impresos es muy conocido tanto por amantes de la electrónica digital, estudiantes y profesionales del campo; en internet se encuentran una larga lista de cómo empezar desde cero en esta tarea [12] e incluso personas que hacen prueba con diferentes químicos.[13]

Para este caso se utilizó los tutoriales del gran maestro [14]**Todopic** [15] utilizando el método del planchado que consiste en los siguientes pasos:

- Imprimir el diseño de las pistas y los pads en papel termotransferible.
- Con una esponja de brillo tomar la baquela de FR4 y lijar la parte del cobre.
- Utilizar una plancha para pasar el calor del papel termotransferible a la cara de cobre de la baquela, este proceso se debe realizar como mínimo por 14 minutos.

---

[12] Pcb Etching, http://www.instructables.com/id/PCB-etching-using-laser-printer/

[13] Stop using Ferric Chloride etchant! (A better etching solution.) by Elliot Graham, http://www.instructables.com/id/Stop-using-Ferric-Chloride-etchant!--A-better-etc/

[14] Como fabricar circuitos impresos en forma artesanal, http://www.todopic.com.ar/apuntes/impresos/impresos.htm

[15] Como hacer circuitos impresos, http://www.pablin.com.ar/electron/cursos/pcb/index.htm

- Una vez se tiene certeza de que la tinta del papel termotransferible ha pasado a la cara de cobre, se retira la plancha y se sumerge este en agua fría para retirar el papel innecesario.
- Se lava con agua abundante y jabón.
- Finalmente se sumerge en cloruro férrico, que es el ácido encargado de consumir el cobre que no se utilizará.

Se debe realizar el mismo procedimiento para las partes de la capa superior, pero en lugar de utilizar papel termotransferible se utiliza papel fotográfico o litográfico.

En la Figura 70, Figura 71, Figura 72, Figura 73, Figura 74, Figura 75, Figura 76, Figura 77 y Figura 78 se evidencian los resultados obtenidos al realizar todos los circuitos impresos con el proceso descrito anteriormente.

**Figura 70:** Pistas impreso principal.

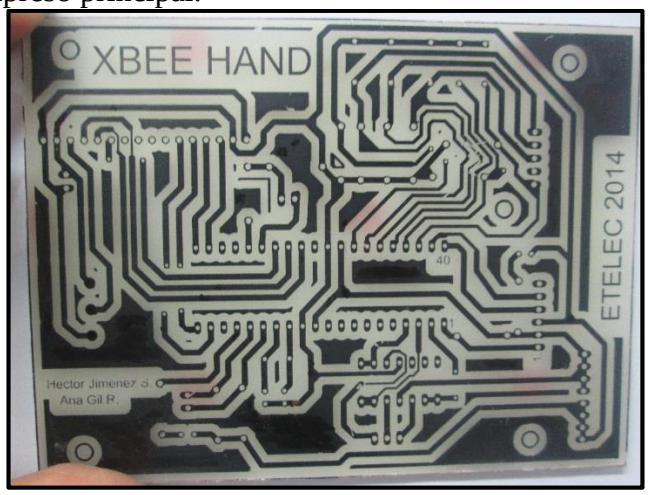



**Figura 71:** Screen del circuito principal.

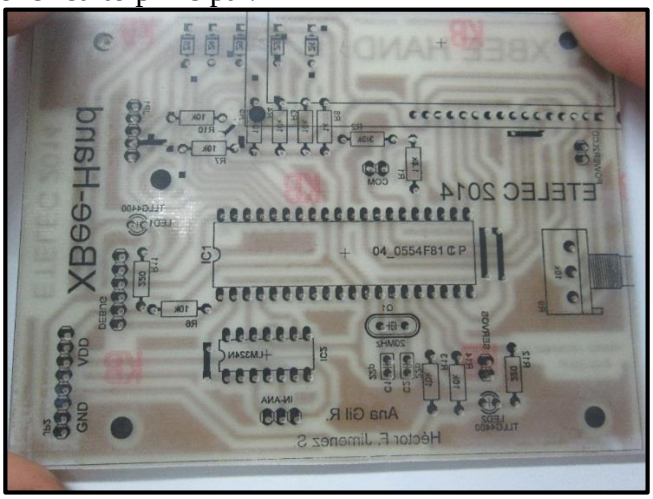


**Figura 72:** Estación de soldadura, pasta para soldar, estallo y des-soldador de vacío.


**Figura 73:** Proceso de soldadura de los elementos al circuito impreso.

**Figura 74:** Circuito principal terminado vista inferior.

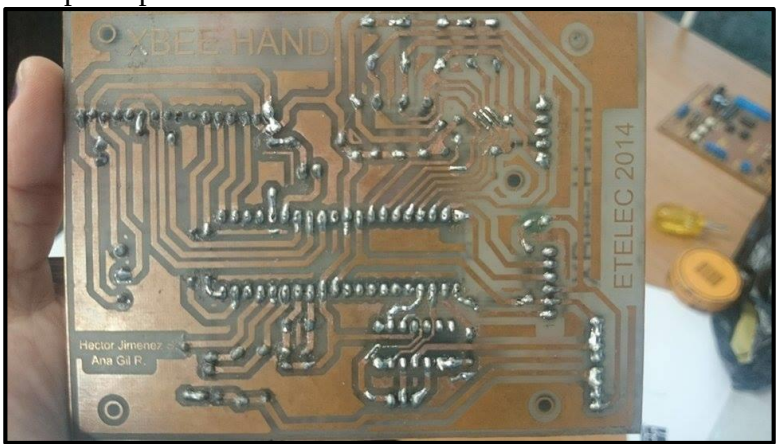


**Figura 75:** Circuito principal terminado vista superior.

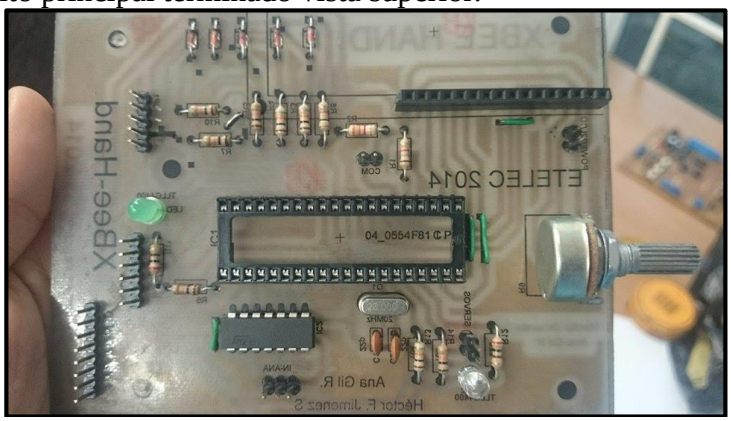



**Figura 76:** Fuente de poder para el joystick.

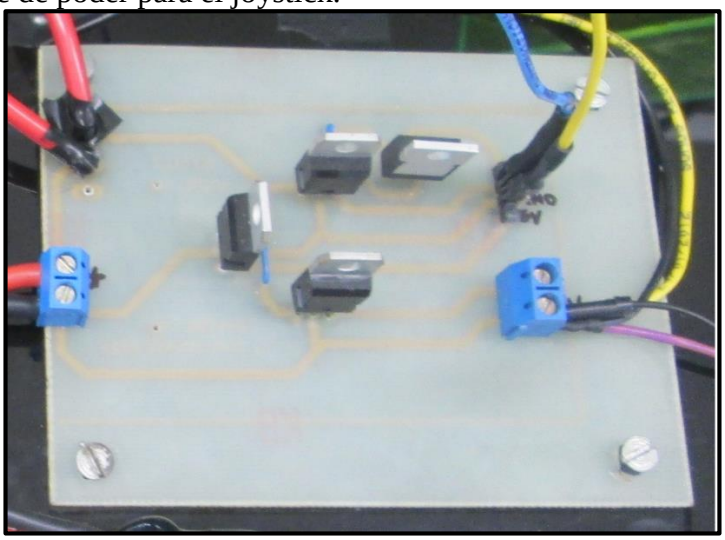


**Figura 77:** Fuente de poder para el brazo robótico.

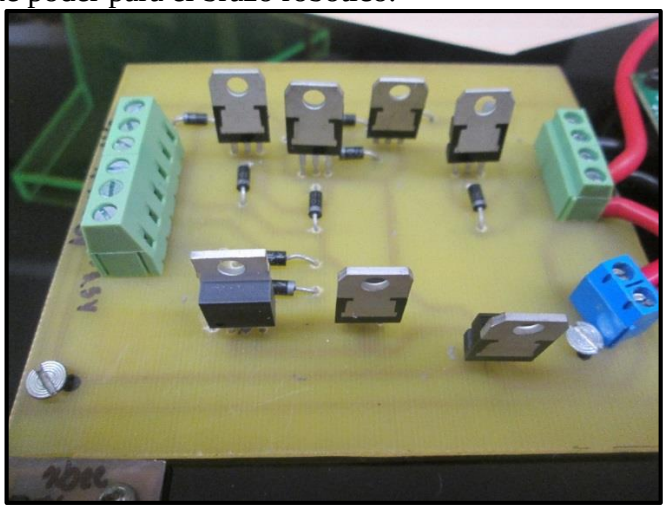
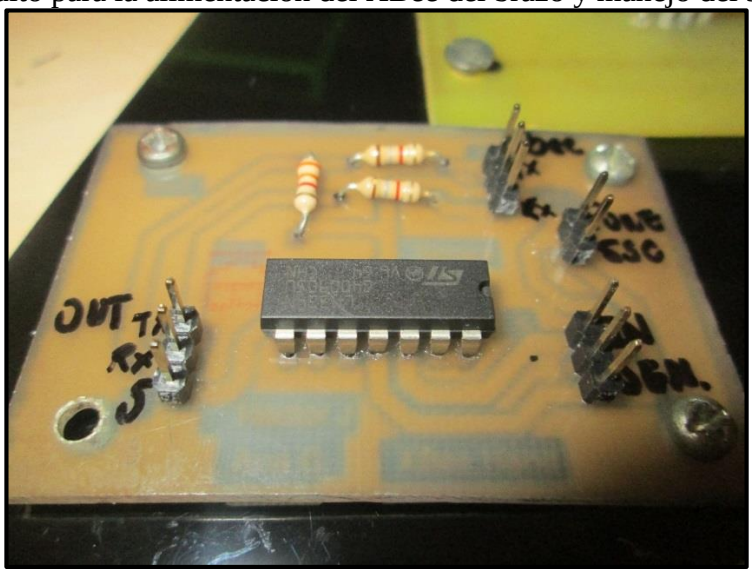


**Figura 78:** Circuito para la alimentación del XBee del brazo y manejo del sensor FSR01.




A pesar que este procesos es extenso si se realiza con cuidado cada paso los resultados son positivos, por ello es importante tener muy buen manejo del software para el diseño del circuito ya sea Eagle ó traxmaker y como se dijo anteriormente seguir cada paso como es indicado.

# 8. RESULTADOS

Se obtuvo un sistema muy dinámico y adaptable a un espacio educativo e inclusive de carácter industrial, es multifuncional y depende esencialmente de la programación, cada movimiento, cada instrucción depende del usuario que finalmente era el objetivo inicial en el proyecto un sistema abierto y muy funcional para el sistema educativo. En el área industrial como se ha visto en los últimos años ha sido de gran aplicación sistemas inteligentes que permitan elaborar tareas que en definitiva son peligrosas para el ser humano.

Finalmente, se logró implementar, manejar y obtener un alto grado de conocimiento en el manejo de los módulos de comunicación XBee, protocolo ZigBee, diferentes recursos del microcontrolador PIC18F4550, la interfaz SSC32 para el control y posicionamiento de los servomotores que componen el brazo robótico y los algoritmos utilizando diferentes estructuras de control y su codificación en un lenguaje de alto nivel.

Durante todo el proceso se comprobó que los módulos XBee pro son dispositivos de comunicación confiables, de bajo costo y alto desempeño en aplicaciones como esta, además permite utilizar todo el protocolo ZigBee y soportar todas las capas de red del estándar 802.15.4 utilizada en estos dispositivos.

## 8.1 PROBLEMAS ENCONTRADOS

En el transcurso del proyecto se presentaron diferentes situaciones que de alguna manera dificultaron mucho más el proceso, algunos de estos problemas fueron:

- **Imprecisión de los potenciómetros del joystick:** el joystick cuenta con tres potenciómetros y al accionar uno de ellos el otro varía significativamente, lo cual genera una imprecisión notable en la lectura de la señal analógica, de ahí la idea de independizar cada señal por medio del uso de teclas digitales.

- **Manejo de la tarjeta SSC32:** a pesar de contar con buena información acerca de esta tarjeta, alguna información (conexión entre la tarjeta SSC32 y el módulo XBee) no se encontró propiamente documentada, lo cual, dificultó el desarrollo con esta interfaz, por ende tocó realizar una consulta propia para poner en marcha la tarjeta.

- **Tramas de Datos erróneas:** inicialmente al enviar los datos de un pc a otro por medio de los módulos XBee los datos de recepción no eran los transmitidos solo eran caracteres incoherentes. Sin embargo, al revisar cada configuración del hyperterminal del pc y del XBee había un problema en la sincronización debido a que las velocidades de transmisión eran diferentes.

- **Conversión análogo digital incorrecta:** en este caso al realizar un proceso de debugging con el IDE MPLABX se encontró que los datos no eran codificados de forma coherente, según la resolución del convertidor, además se confirmó mediante

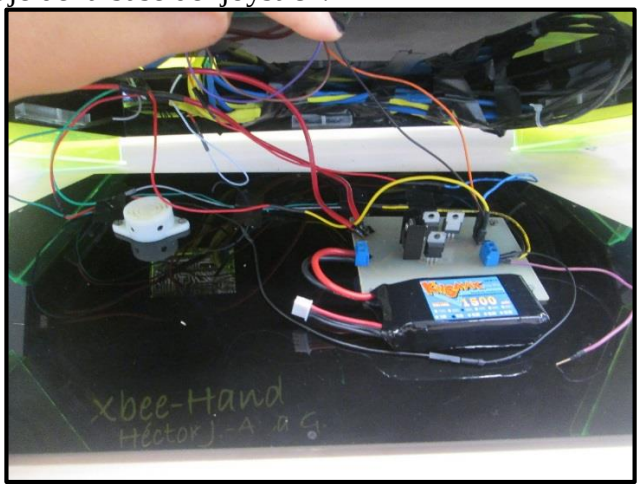
una lectura física de una señal de tensión proveniente de uno de los potenciómetros del joystick. Por ende, fue necesario implementar un amplificador operacional en configuración seguidor de tensión para eliminar el ruido en la señal y disminuir la impedancia de entrada en el puerto análogo del microcontrolador, adicionalmente, se configuro el modulo A/D con una justificación de 8 bits a la izquierda.

## 8.2 RESULTADO FINAL

Finalmente, se tiene un brazo robótico controlado por un joystick el cual envía sus instrucciones por comunicación inalámbrica, para mostrar este prototipo final se diseñaron dos bases adecuadas para soportar cada elemento, estas bases fueron diseñadas en CorelDRAW[16] y simuladas en Rhinoceros 3D[17] ver Anexo 1 y posteriormente construidas físicamente en acrílico negro y verde neón.

En la Figura 79, Figura 80, Figura 81, Figura 82, Figura 83 , Figura 84, Figura 85, Figura 86 y Figura 87 se muestra el proceso de ensamblaje y terminado del prototipo.

**Figura 79:** Ensamblaje de la base del joystick.




[16] Software de diseño gráfico, aplicado el dibujo, la maquetación de páginas para impresión y/o la publicación web, etc. Para mayor información visitar la página web http://www.coreldraw.com/us/.
[17] Software de diseño gráfico para el modelado en 3D. Para mayor información visitar la página web http://www.rhino3d.com/.

**Figura 80:** Base del joystick terminada vista superior.


**Figura 81:** Base del joystick terminada vista superior.

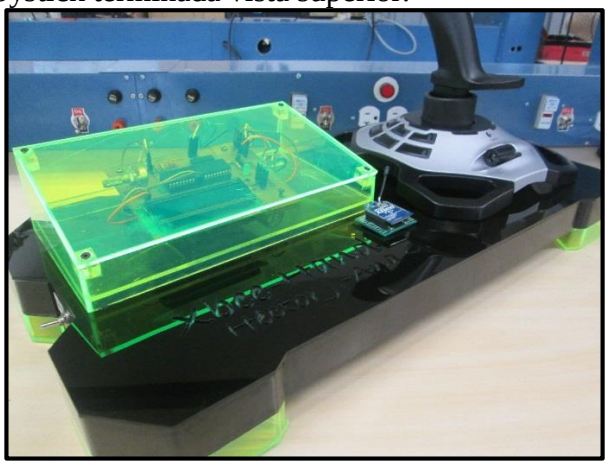


**Figura 82:** Base del joystick terminada vista lateral.

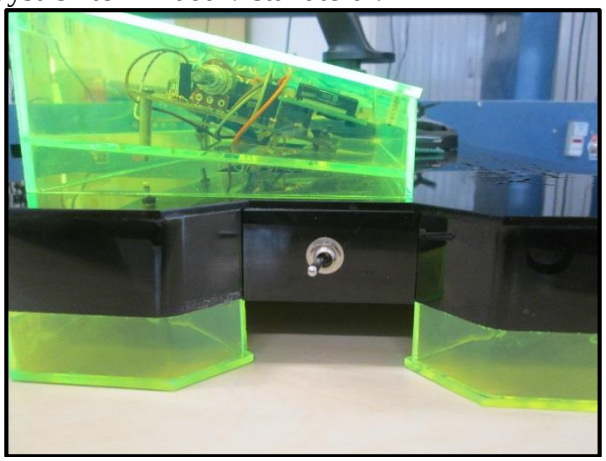


**Figura 83:** Ensamblaje de la base del brazo robótico.

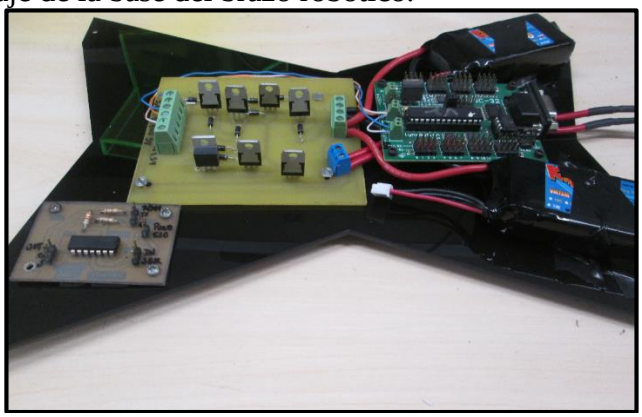


**Figura 84:** Base del brazo terminada vista inferior.

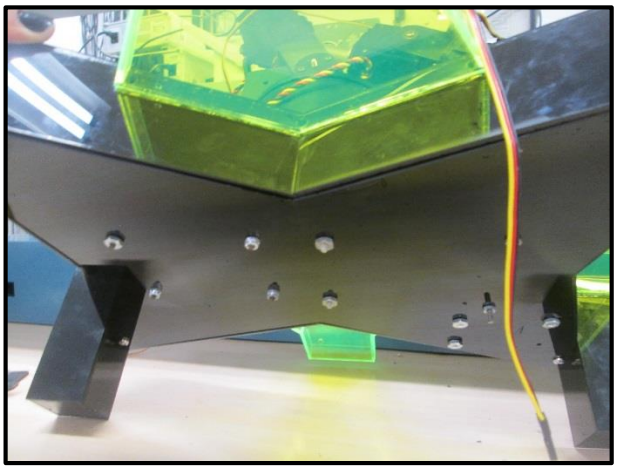


**Figura 85:** Base del brazo terminada vista lateral.

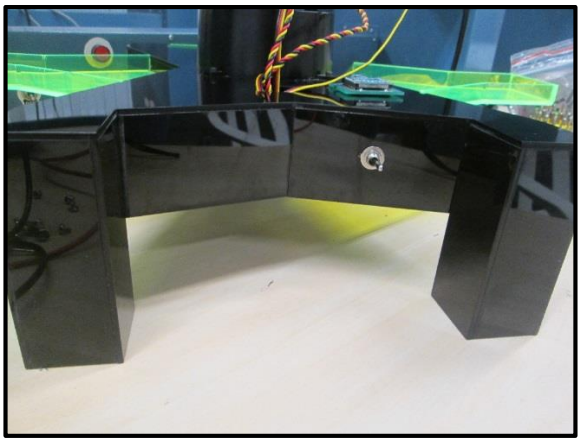


**Figura 86:** Base del brazo terminada vista superior.

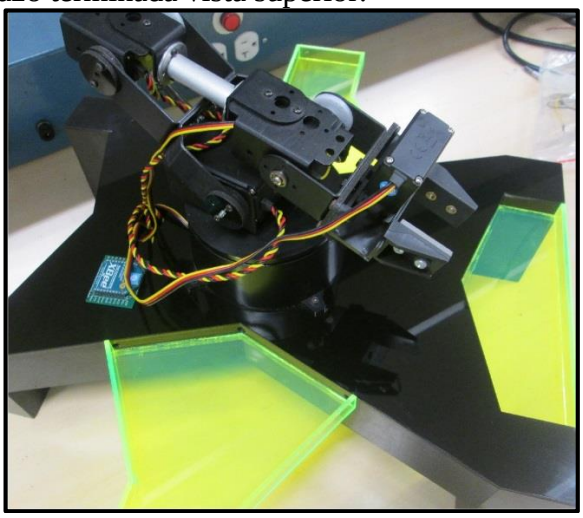
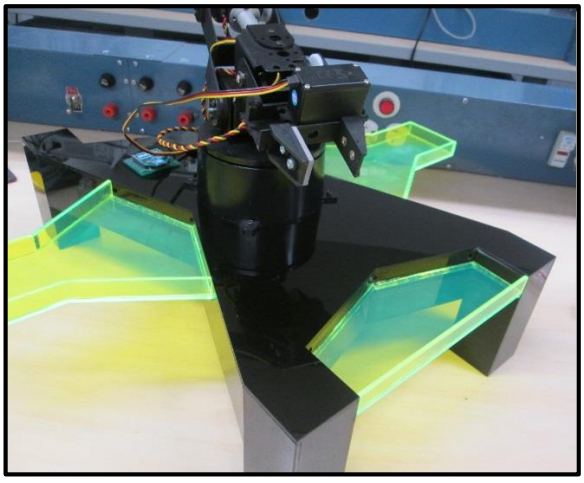


**Figura 87:** Base del brazo terminada vista lateral.



# 9. CONCLUSIONES

- Durante las pruebas de comunicación se pudo comprobar que el uso de dispositivos inalámbricos externos a la comunicación del proyecto no afectan el desempeño del prototipo y el envío de datos, pues realizando el direccionamiento adecuado entre módulos no se presentan pérdidas de datos ó interferencias electromagnéticas en la señal.

- Es trascendental identificar los aspectos más relevantes dentro de un sistema de comunicación como el implementado dentro de este proyecto, algunos de estos pueden ser una velocidad de transmisión óptima en el medio, comprobación de datos CRC, de esta manera se garantiza una comunicación limpia y mucho más segura.

- La caracterización de los servomotores juega un papel muy significativo en cualquier sistema robótico, ya que no todos se comportan igual más aun cuando cada uno realiza un movimiento específico y diferente en el sistema. Por ejemplo, el hombro no realiza el mismo movimiento que la muñeca y así sucesivamente, por ende si no se realiza esta caracterización en cualquier momento el sistema puede fallar y hasta colapsar.

- En todo el proceso del proyecto se logró utilizar varios de los recursos del microcontrolador 18F4550 como su convertidor análogo-digital, comunicación serial asíncrona, manejo de entradas o salidas, manejo de interrupciones, los cuales son de gran importancia ya que todas están dentro de un solo circuito integrado, permitiendo realizar el desarrollo del algoritmo.

- La tarjeta SSC32 fue de gran beneficio para este proyecto, es una interfaz muy útil, eficiente, y amigable con el usuario. Presenta la oportunidad de diseñar un sistema más amplio con un gran número de servos, con una resolución en la posición de 0.09°/unidad.

- A pesar del error en la señal análoga enviada por el joystick, este es un dispositivo con muchas posibles combinaciones debido a su número de teclas, lo cual permitió implementar la idea de utilizar teclas para discriminar cada lectura de los potenciómetros. Asimismo es ergonómico y cómodo para el usuario.

- La programación estructurada fue muy importante en este proyecto ya que permite una mayor organización, ubicación de fallas humanas, eficiencia en el tiempo de compilación, ejecución y disminución en el tamaño del código objeto.

- El sensor de presión en los gripper o pinzas robóticas es de gran aplicación en la robótica ya que ayuda a generar una retroalimentación mecánica de la misma, para controlar la fuerza de agarre, de esta manera el objeto a sujetar no será destruido por el mismo al intentar cerrarse más de lo que el objeto soporta.

## 10. BIBLIOGRAFÍA


[1] Wikipedia, «Robot,» 2011. [En línea]. Available: http://es.wikipedia.org/wiki/Robot. [Último acceso: 20 01 2014].

[2] I. M. Ltda, «Xbee Chile,» Ingenieria MCI Ltda, 15 01 2012. [En línea]. Available: http://www.xbee.c1/caracteristicas.html.

[3] R. Faludi, Building Wireless Sensor Networks:A Practical Guide to the Zigbee Mesh Networking protocol., Sebastoopol,CA: Oreilly, 2011.

[4] M. F. Soed, «ZigBee Application : Autonomous Robot,» Saturday, August 29, 2009, 29 08 2009. [En línea]. Available: http://vr-mission.blogspot.com/2009_08_01_archive.html. [Último acceso: 11 10 2014].

[5] I. Digi International, XBee®/XBee-PRO® DigiMesh™ 2.4 RF Modules, 2010.

[6] F. A. Candelas Herías y J. A. Corrales Ramón, «Servomotores,» Alicante, 2007.

[7] Lynxmotion, «Users Manual SSC-32 Ver 2.0,» 2005.

[8] F. Jim, «Manual de SSC-32,» 2009 .

[9] Monografias, Microcontroladores , S.F. [En línea]. Available: http://www.monografias.com/trabajos12/microco/microco.shtml.

[10] S. Sánchez., «Microcontroladores,» 2013. [En línea]. Available: http://microcontroladoressesv.wordpress.com/arquitectura-de-los-microcontroladores/.

[11] R. P. A. Fernando E. Valdés Pérez, «Microcontroladores: fundamentos y aplicaciones con PIC,» España, Carles Parcerisas Civit (3Q Editorial), 2007.

[12] A. P. A. Eduardo P. Domínguez de la Cruz, «Paquete didáctico para la implementación del PIC 18F4550.,» Mexico D.F, 2012.

[13] Microchip, *PIC18F2455/2550/4455/4550 Data Sheet*, U.S.A, 2006.

[14] U. P. d. Valencia, «Microcontrolador PIC18F4550,» [En línea]. Available: http://www.joseapicon.com.ve/descargas/pic/Manual%20PIC%2018F4550.pdf.

[15] I. Servicios, «Informatica moderna,» 2014. [En línea]. Available: http://www.informaticamoderna.com/Joystick.htm.

[16] Wikipedia, «Batería Eléctrica,» 21 05 2014. [En línea]. Available: http://es.wikipedia.org/wiki/Bater%C3%ADa_el%C3%A9ctrica#Bater.C3.ADas_de_pol.C3.ADmero_de_litio_.28LiPo.29.

[17] D. Electronics, «Dynamo Electronics,» [En línea]. Available: http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=974&category_id=87&option=com_virtuemart&Itemid=58. [Último acceso: 07 2014].

[18] D. Electronics, «Dynamo Electronics,» 2007. [En línea]. Available: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/IMAX-B6.pdf.

[19] J. M. A. Usategui, Robotica practica: tecnología y aplicaciones., Madrid: Ediciones Paraninfo, S.A , 1999.

[20] «Creando el futuro. Net,» [En línea]. Available: http://creandoelfuturo.net/es/morfologia-del-robot/estructura-mecanica-robot. [Último acceso: 08 06 2014].

[21] F. H. R. Leyva, «Robótica, Modelado Cinemática de Robots,» 2012.

[22] «SlideShare,» [En línea]. Available: http://es.slideshare.net/ohckang/robotica-edinsoncs-ockangplcautomatas-cinematicaautomatizacion-y-robotica. [Último acceso: 15 08 2014].

[23] E. Dynamo, «Brazo Robotico AL5A,» Bucaramanga.

[24] SparkFun, «SparkFun XBee Explorer Serial,» 2009. [En línea]. Available: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Serial-Explorer-v12.pdf. [Último acceso: 12 09 2014].

[25] U. P. d. Madrid, «Entornos de programación,» [En línea]. Available: http://lml.ls.fi.upm.es/ep/0708/entornos.html.

[26] Microchip, «MPLAB C18 Compiler,» 2014. [En línea]. Available: http://www.microchip.com/forums/m82318.aspx.

[27] J. G. d. J. d. l. Fuente, Aprenda lenguaje ANSI C, como si estuviera en primero, San Sebastian, 1998.

[28] S. G. d. I. Electrónicas, «Sagitron,» 2012. [En línea]. Available: http://www.sagitron.com/productos/activos/microchip/203-pickit3-programador-de-pic-sin-necesidad-de-ordenador. [Último acceso: 13 11 2014].

[29] R. Sharma, «ELECTRONICS LAB,» [En línea]. Available: http://electronics-lab.com/projects/audio/010/. [Último acceso: 13 11 2014].

[30] Colourbox, «Colourbox,» 2011. [En línea]. Available: https://www.colourbox.com/image/repair-and-diagnostic-electronics-isolated-on-white-background-image-2728995. [Último acceso: 13 11 2014].

[31] C. M. C. S. Alfonso Zapata Guarín, Brazo robotizado para alimentación y descarga de un sistema de producción de piezas de madera, Pereira: Universidad Tecnológica de Pereira Colombia , 2010.

## ANEXO 1

## PLANOS TÉCNICOS DE LAS BASES PARA EL PROTOTIPO

**Figura 1:** Ficha técnica de la base del joystick vista superior

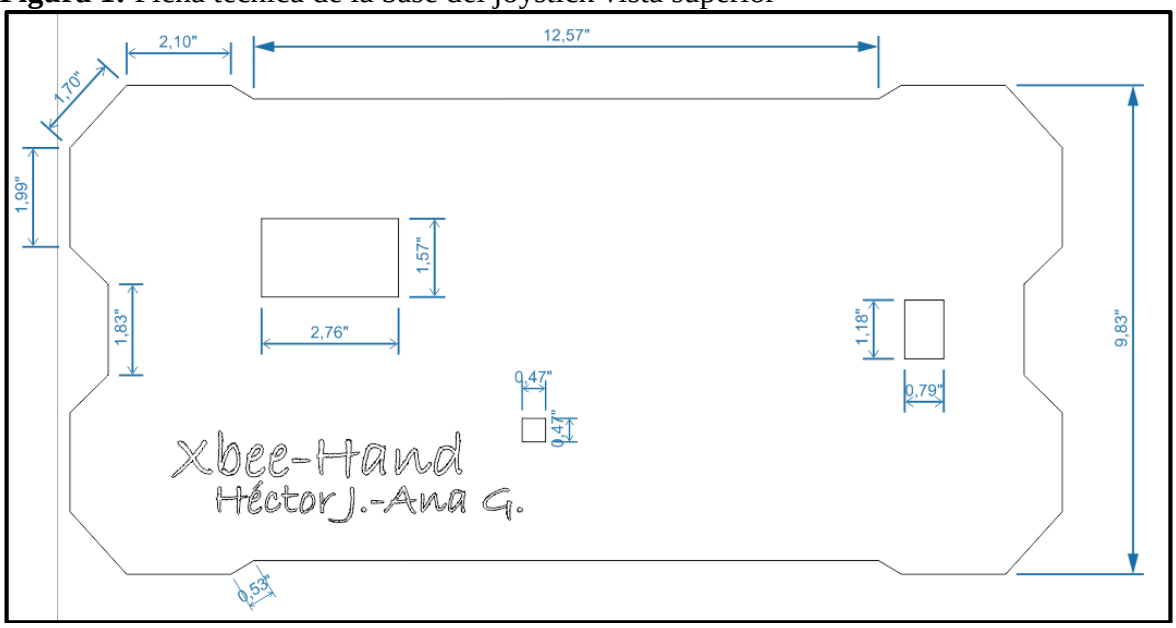



**Figura 2:** Ficha técnica de la base del joystick vista superior con la base para el impreso.

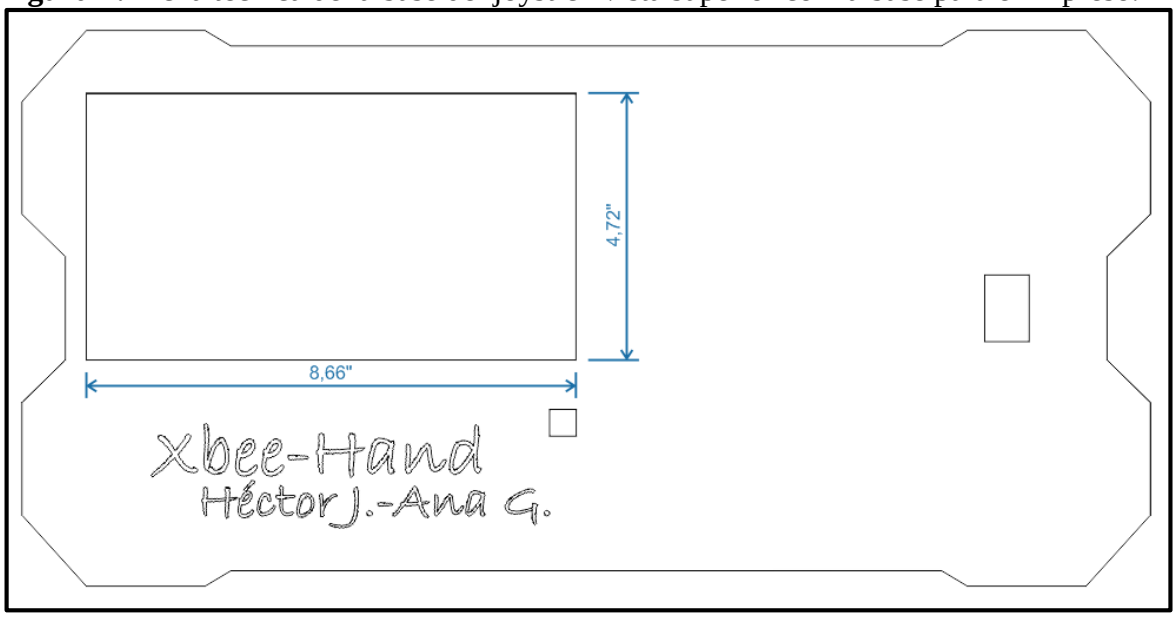


**Figura 3:** Ficha técnica de la base del joystick.

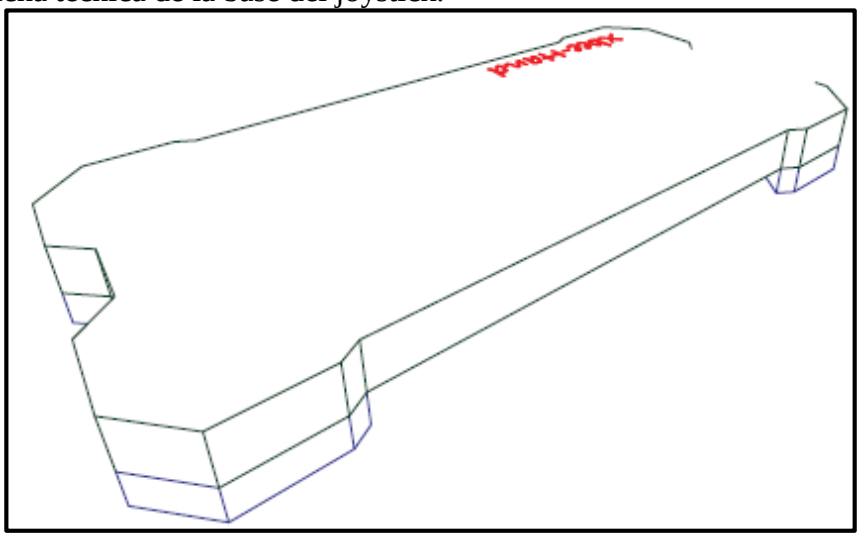


**Figura 4:** Ficha técnica de la base del joystick vista frontal.

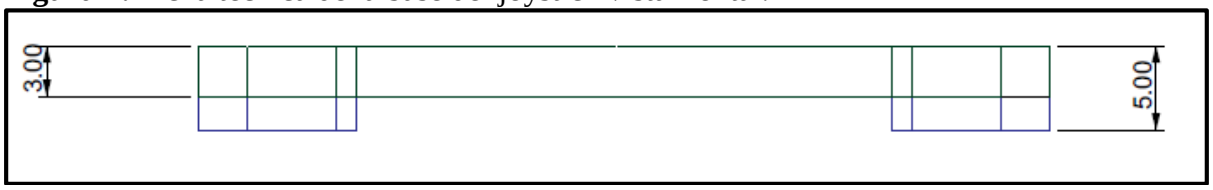



**Figura 5:** Ficha técnica de la base del joystick vista lateral.

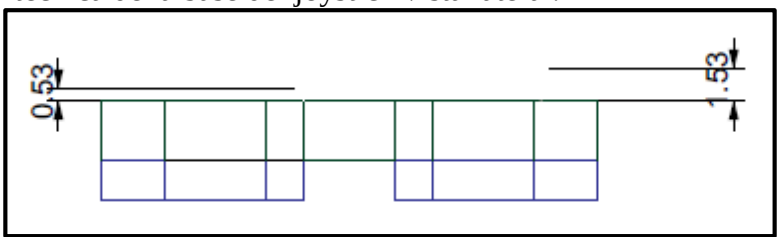


**Figura 6:** Ficha técnica de la base del brazo vista superior

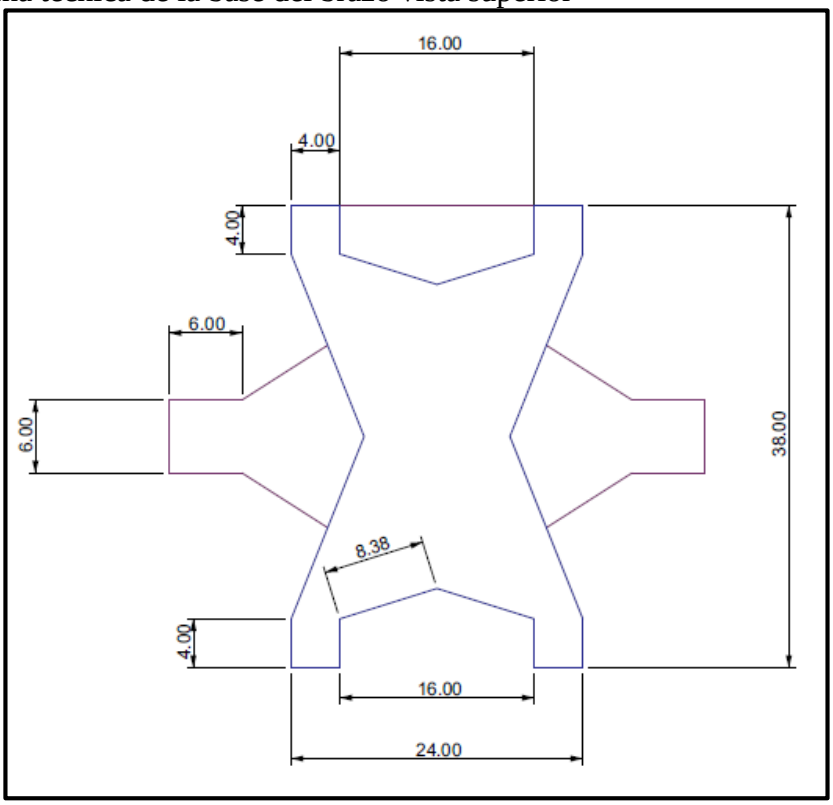



**Figura 7:** Ficha técnica de la base del brazo vista lateral.

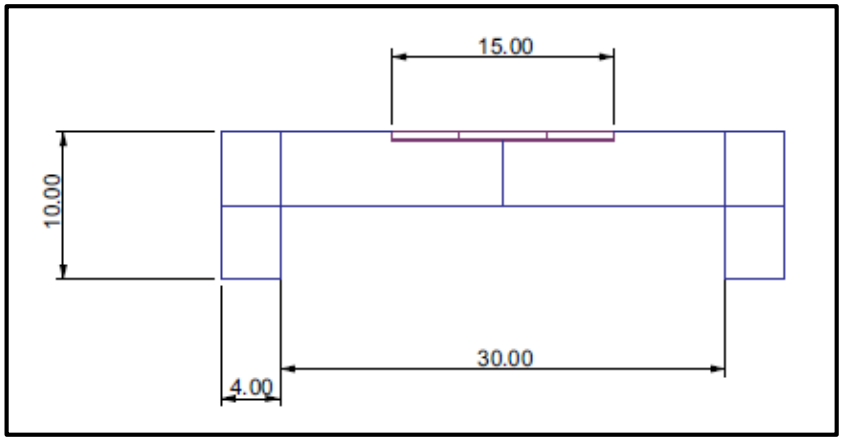


**Figura 8:** Ficha técnica de la base del brazo vista frontal.

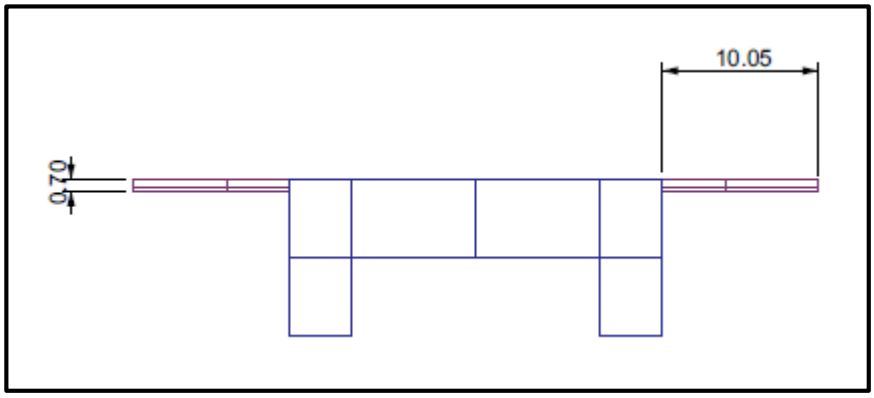
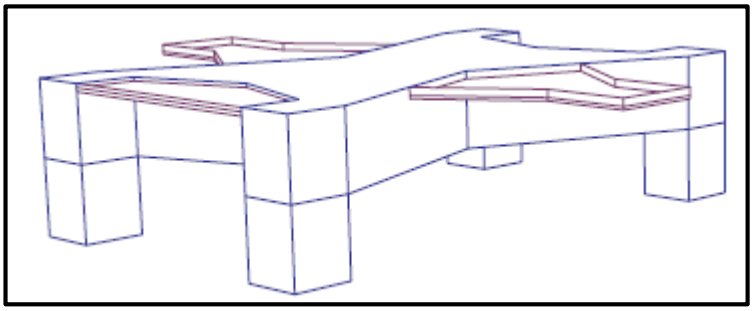



**Figura 9:** Ficha técnica de la base del brazo.



**SIMULACIÓN DE LAS BASES EN RHINOCEROS 3D**

**Figura 10:** Simulación 3D de la base del joystick.

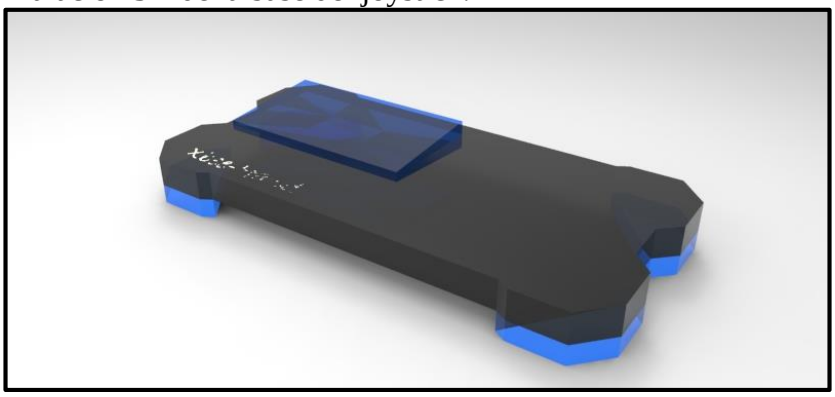
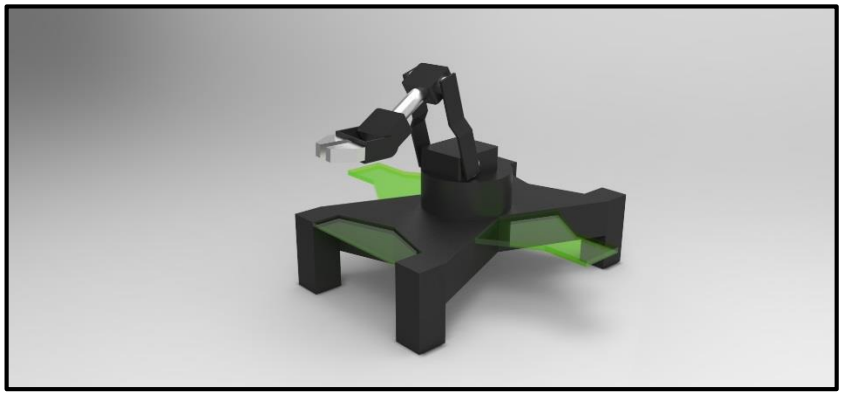


**Figura 11:** Simulación 3D de la base del brazo.



## ANEXO 2

## DIAGRAMAS DE FLUJO

**Figura 12:** Diagrama de flujo convertidor A/D

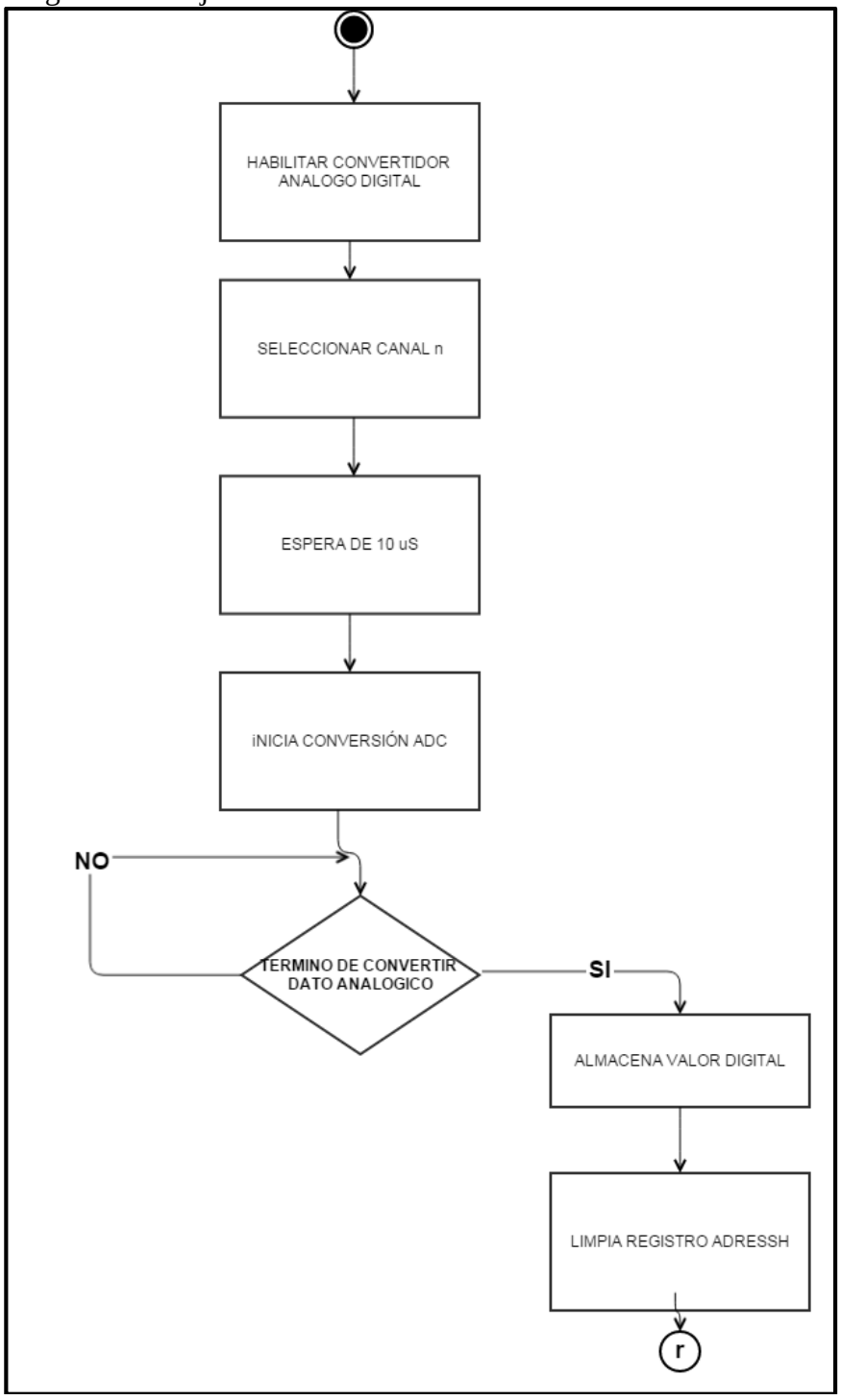
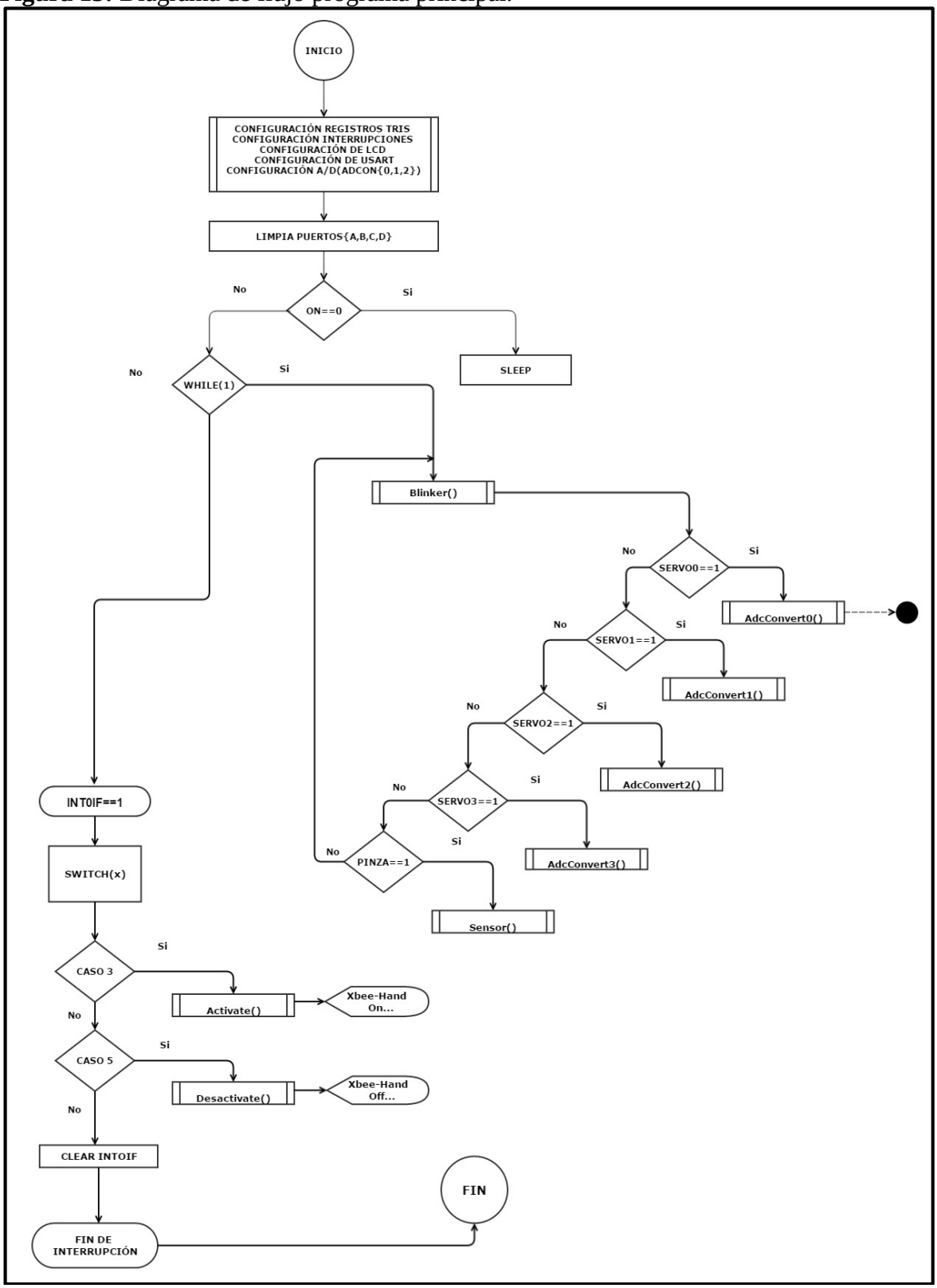


**Figura 13:** Diagrama de flujo programa principal.



**CALL GRAPHS PARA DIAGRAMAS DE FLUJO:**

La herramienta call graph permite visualizar las llamadas de todo el programa, al ser este proyecto estructurado el programa principal fue subdividido en programas más pequeños.

**Figura 14:** Funciones del programa

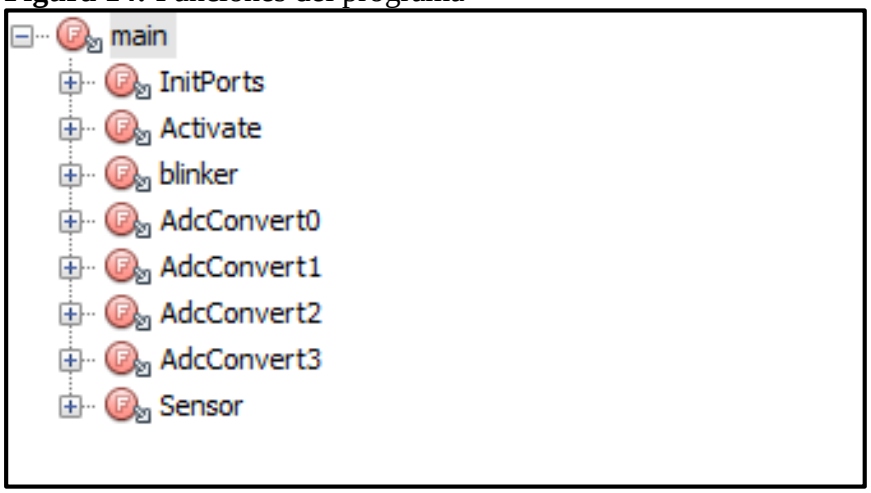



**Figura 15:** Llamadas a funciones del bloque principal (*main)*.

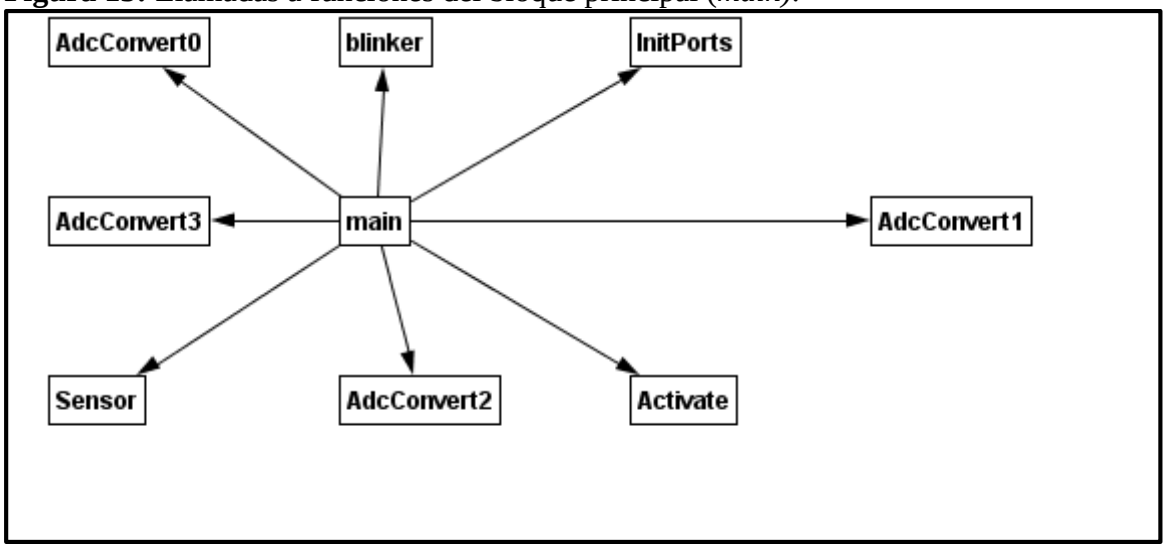
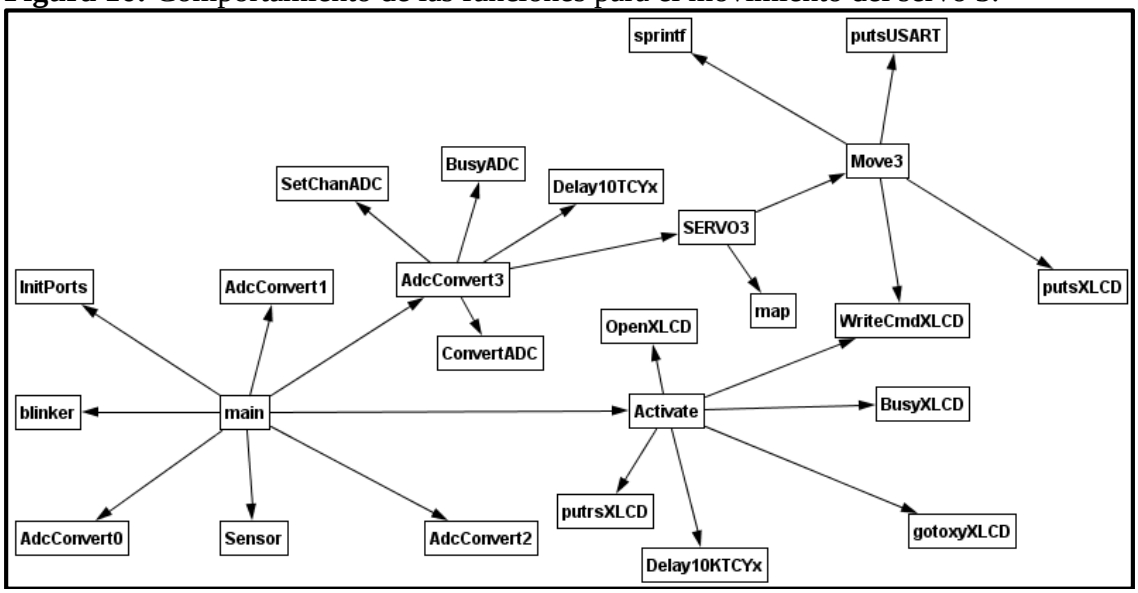


**Figura 16:** Comportamiento de las funciones para el movimiento del servo 3.



## ANEXO 3

### ESQUEMÁTICOS DE LAS FUENTES DISEÑADAS PARA REGULAR LA TENSIÓN DE LAS BATERÍAS

Debido a que las baterías utilizadas tienen una tensión de 11.1V fue necesario diseñar dos fuentes una para el brazo y otra para el joystick de manera que se logrará reducir este valor a las tensiones requeridas a través de los reguladores de tensión 7809,7805 y LM1117 para obtener tensiones de: 9V, 5V y 3.3V respectivamente.

**Figura 17:** Fuente final base del joystick, tensiones: 9V, 5V, 3.3V.

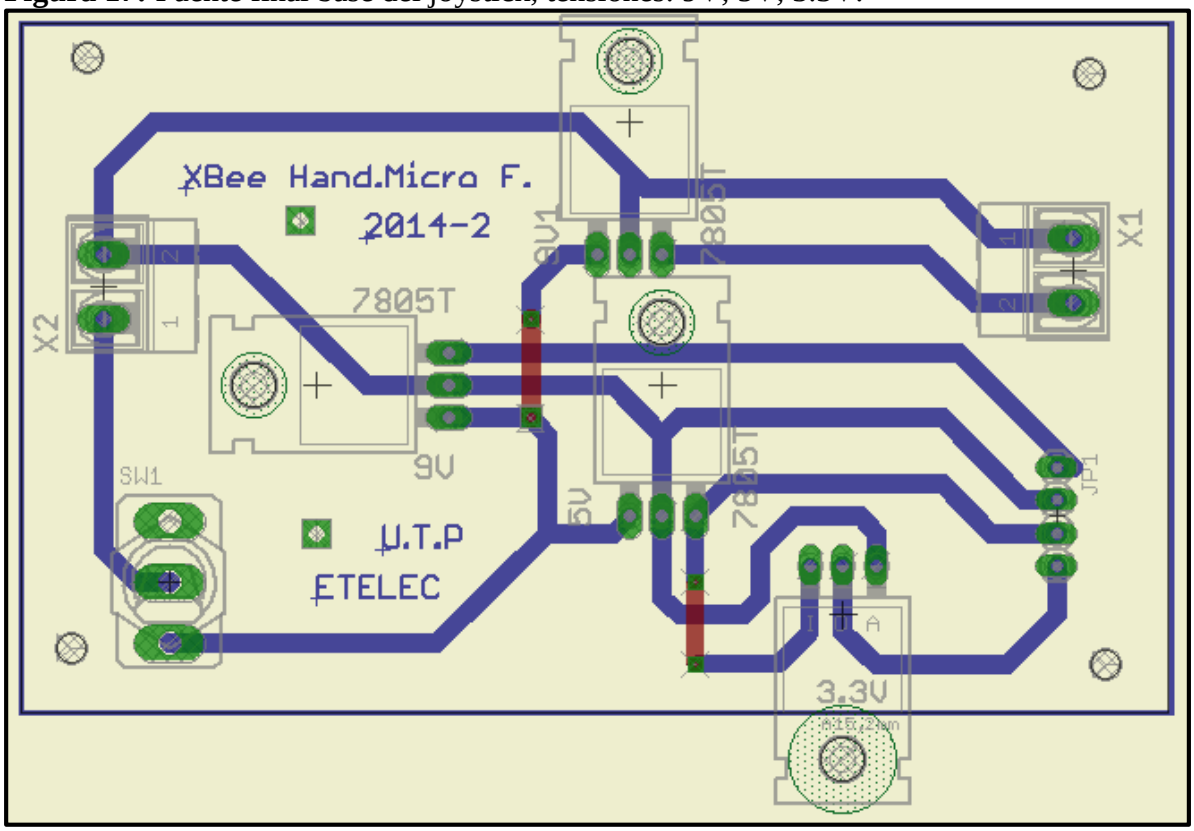


**Figura 18:** Fuente final base del brazo, tensiones: 9V, 6V, 5V, 3.3V.