UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERÍA ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA: DISEÑO E IMPLEMENTACIÓN DE UN ROBOT HEXÁPODO PARA INSPECCIÓN DE CÁMARAS DE REGISTRO DE FIBRA ÓPTICA PARA LA EMPRESA TELCONET EN LA CIUDAD DE QUITO.

AUTOR: DARWIN ANDRES CAMPOVERDE ORDOÑEZ

TUTOR: MG. RENÉ CORTIJO LEYVA

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR:

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación ”TEMA: DISEÑO E IMPLEMENTACIÓN DE UN ROBOT HEXÁPODO PARA INSPECCIÓN DE CÁMARAS DE REGISTRO DE FIBRA ÓPTICA PARA LA EMPRESA TELCONET EN LA CIUDAD DE QUITO.”, presentado por la Sr. Darwin Andres Campoverde Ordoñez, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. 24 Agosto del 2017

TUTOR

.....

Ing. Rene Ernesto Cortijo Leyva, Mg

Dedicatoria

Este documento lo dedico con mucho aprecio a mis padres que me han dado su apoyo incondicional todo el tiempo de vida de estudiante. Y las personas que han aportado de alguna manera a lograr este paso en mi vida.

Índice general

Índice de figuras

Indice de Tablas

Resumen

El presente proyecto consiste en un diseño electromecánico que pretende solventar incidencias que ocurren en las cámaras de registro de la empresa Telconet.

Para el efecto, inicialmente se realizó un estudio de campo que permitió conocer los procedimientos que realizan los técnicos para el ingreso a las cámaras de registro, actividades que realizan dentro de las cámaras de registro tales como; revisión de cables de fibra óptica, acometidas, instalación y desmontaje de cables de fibra, creación de mangas y fusiones. También se realiza revisiones de la infraestructura civil. Dentro de todos los procedimientos descritos en la cámara de registro se evidenciaron inconvenientes, como riesgos de seguridad al momento del ingreso y la salida de los técnicos de Telconet e incomodad al realizar algún trabajo dentro de la cámara. Condiciones ambientales adversas dentro de la cámara de registro principalmente como presencia de gas, baja iluminación, altas temperaturas y humedad.

Con la construcción de un robot zoomórfico con transmisión de video y control remoto mediante comunicación inalámbrica, a través de una interfaz gráfica en un Smartphone con sistema operativo Android, se pretende evitar o minimizar los inconvenientes y posibles riesgos identificados para los técnicos que trabajan diariamente en la revisión física de las cámaras de registro. Dando la alternativa de revisar la cámara de registro desde afuera, manteniendo seguro a la persona que va a ingresar. De esta manera se podrán identificar los factores nocivos para el ser humano y también se realizará la revisión de la infraestructura de la cámara de registro.

Toda la información referente a la programación o código fuente del hardware estará adjuntada en Anexos al final de la documentación del proyecto.
Palabras Clave: Diseño, Robot, Hexápodo, Inspección, Cámaras, Fibra, Telconet.

Abstract

The present project consists of an electromechanical design that seeks to solve incidents that occur in the registration chambers of Telconet. For this purpose, a field study was initially carried out, which allowed the technicians to know the procedures for entering the registration chambers, activities carried out within the registration chambers such as; Revision of fiber optic cables, connections, installation and disassembly of fiber cables, creation of sleeves and fusions. Civil infrastructure reviews are also carried out. Within all the procedures described in the registry chamber, there were problems such as security risks at the time of entry and exit of Telconet technicians and inconvenience when performing some work inside the camera. Adverse environmental conditions within the recording chamber mainly as gas presence, low illumination, high temperatures and humidity.

With the construction of a zoomorphic robot with video transmission and remote control through wireless communication, through a graphical interface on a Smartphone with Android operating system, it is intended to avoid or minimize the disadvantages and possible risks identified for technicians working daily in physical review of the registration chambers. Giving the alternative to check the registration chamber from the outside, keeping the person who is entering. In this way, it will be possible to identify the factors harmful to the human being and also the revision of the infrastructure of the registration chamber will be carried out.

This document is structured as follows: in point 1, the objectives of the project are described, so that the proposed goals can be understood. In section 2, an introduction to the topic is made explaining how mechanical design, electronic design, communication, robot control and implementation will be performed. Point 3 will detail the tests, results of the functionality and operation of the robot. And as the last point will see the conclusions and recommendations of the Project. All information regarding the programming or source code of the hardware will be attached in Attachments at the end of the project documentation.

INTRODUCCIÓN

1.1. Antecedentes de la situación objeto de estudio

La palabra robot fue creada por el checoslovaco de nombre Karel Capek, para una obra teatral. La palabra ”robota” en checoslovaco significa ”Trabajador de servicio obligatorio”. Dentro de las nuevas tendencias de trabajo automatizado, la robótica es una herramienta indispensable en varios campos de la tecnología y ciencias. Como por ejemplo en la exploración, medicina, industria y en general procesos automáticos. Dentro de la ciencia robótica existen varias divisiones o distribuciones que se clasifican según su semejanza. Normalmente los robots son creados para suplir algún tipo trabajo que realiza el ser humano en cuanto a su necesidad de entorno. Por eso se han creado brazos robóticos, que realizan tareas repetitivas, con capacidad de carga y con precisión. Algo que el ser humano no lo podría hacer de forma continua por horas debido a sus limitaciones. También se han creado robots móviles por su facilidad de transporte y traslado. Hay otros tipos de robots que ayudan al ser humano a realizar tareas en donde el ambiente es nocivo para la salud. Los robots zoomórficos son robots que asemejan su forma y motricidad a los animales. Entre este tipo de robot se encuentran los hexápodos que son de forma parecida a los insectos artrópodos como las hormigas, moscas, mariposas. Los robots hexápodos cuentan con 6 articulaciones para realizar su movimiento de caminar o desplazarse. Sirven para la comprobación de la biología, y debido a sus 6 extremidades tiene una gran estabilidad. Algo que ayudaría a realizar varias actividades en cualquier terreno. A este robot se le puede acoplar varios tipos de sensores que sirven para el monitoreo del ambiente en el que se encuentran y también se le puede instalar una cámara de video para que nos permita revisar visualmente en donde se encuentra. De esta manera facilita las revisiones de cualquier ambiente sin tener que el ser humano tenga que ingresar a estos lugares.

1.2. Planteamiento del problema

El personal técnico de la empresa Telconet realiza inspecciones periódicas de la infraestructura civil de sus cámaras de registro que se encuentran en la ciudad de Quito, con el fin de dar mantenimiento preventivo y correctivo a las mismas. El procedimiento

para la inspección se realiza con un técnico que debe ingresar dentro de la cámara de registro mediante el uso de una escalera de metal que le permitirá bajar y realizar la inspección. Este trabajo conlleva tiempo considerable y es poco cómodo para el operador, además de que puede acarrear problemas de seguridad, ya que el técnico encargado puede lesionarse al momento de ingresar y salir del mismo. Además, en las cámaras de registro existen condiciones ambientales contraproducentes para los seres humanos, por ejemplo, al ser un lugar cerrado, existe la acumulación de gases tóxicos como existen en las construcciones bajo tierra y altas temperaturas en días soleados lo cual puede ocasionar problemas de salud a la persona que realiza trabajos en las cámaras de registro.

1.3. Formulación del problema

En las inspecciones rutinarias que realizan el personal de Telconet, dentro de las cámaras de registro, pueden presentarse varios inconvenientes al momento de realizarlas como: la manera de acceder a la cámara ya que pueden sufrir lesiones al momento de descender porque la entrada es angosta. La cámara puede contener concentraciones peligrosas de gas que pueden ser nocivas para el personal técnico encargado. También no se cuenta con un procedimiento técnico para hacer una inspección de la cámara de registro y obtener datos de la misma.
Para poder mejorar el procedimiento y minimizar inconvenientes, el robot hexápodo será el encargado realizar el análisis de las cámaras de registro operado por una persona remotamente, con esta nueva herramienta a disposición, las inspecciones a las cámaras de registro pueden realizarse de mejor manera, ya que el robot será el encargado de proveer a la persona que lo manipula, imágenes de video, datos de los sensores de gas, temperatura y humedad. Y se las presentará por medio de una aplicación de interfaz gráfica para un teléfono inteligente Android, donde el operador podrá tener esta información de manera fácil y desde una postura cómoda.

1.4. Justificación

El robot tipo hexápodo ayudará a realizar maniobras para explorar las cajas de distribución de la empresa Telconet. Ya que normalmente dentro de los procesos del área de operaciones Urbanas y Fibra óptica se realizan las inspecciones de las cámaras de registro. Este tipo de inspecciones lo realizan una vez al mes por motivos de seguridad y mantenimiento de las mismas. La empresa Telconet al contar con muchas cámaras de registro dentro de la ciudad de Quito, les lleva bastante tiempo realizar este tipo de tarea. Con las comunicaciones dedicadas a la fibra óptica se debe tener mucho cuidado ya que el tráfico de comunicaciones es elevado debido a los clientes con los que cuenta. Mediante la toma de datos de los sensores instalados en el robot, se podrá analizar la temperatura y los gases peligrosos que existen en las cajas de distribución. Con la

cámara instalada al robot se podrá realizar una revisión visual más cómoda y completa. El control de las extremidades del robot facilitará el posicionamiento del mismo; De esta manera se reducirá los riesgos que existen para los técnicos de Telconet que realizan este trabajo. Y se reducirán los tiempos de operaciones para las inspecciones de las cámaras de registro.

1.5. Objetivo General

Implementar un robot hexápodo para la inspección física de cámaras de registro de fibra óptica para la empresa Telconet en la ciudad de Quito.

1.6. Objetivos Específicos

- Diseñar el esqueleto del robot para un correcto funcionamiento de articulaciones.
- Desarrollar el software que permita controlar los servomotores para la motricidad del robot.
- Desarrollar una interfaz gráfica para el control remoto del robot y el monitoreo de los distintos parámetros.
- Realizar pruebas de funcionamiento y correcciones necesarias para la correcta operatividad del robot.

1.7. Descripción de los capítulos

En el capítulo 1 se incluye toda la información relacionada que ha sido objeto de partida para comenzar el proyecto. Estudio de robots, funcionamiento, tipos. La teoría de servomotores y su funcionamiento, Tecnologías de control automáticos entre los que están el microcontrolador, tarjeta Arduino, sensores. Tecnologías de comunicación inalámbrica que se utiliza hoy en día Wifi, Bluetooth. Software de programación para el desarrollo de la comunicación del smartphone y control del robot.
En el capítulo 2 se presentara los diagramas de diseño mecánico, electrónico, etapa de potencia, Cálculos de consumo de energía del robot. Lógica de programación para el funcionamiento del robot, programación del microcontrolador para la etapa de control y administración de todo el sistema según la tarjeta Arduino. Diseño de las tarjetas para el acople de los periféricos pines. Interfaz gráfica para el control desarrollado con el software para Android. Análisis de los sensores a utilizar para la comunicación con la interfaz.
Finalmente en el capítulo 3 se explicará el desarrollo de todas las partes del sistema de funcionamiento del robot. Como se realizará el montaje del robot desde la creación de

las extremidades hasta la integración final del robot. Se realizaran pruebas de funcionamiento del robot para la comprobación de la motricidad del robot. Se presentará una tabla de resultados de los datos obtenidos en el campo.

CAPÍTULO I FUNDAMENTACION TEÓRICA

2.1. Robótica.

El término robot engloba a un gran nombre de dispositivos y mecanismos, lo que hace que sea muy difícil establecer una clasificación. El siguiente apartado se basa en la arquitectura del robot, es decir, su configuración general. De esta forma, se ecnuentran los siguientes grupos: poliarticulados, móviles, androides, zoomórficos e híbridos.(Almeida Hernández and Ochoa Urgilés(2013)).
La historia de la automatización industrial está caracterizada por períodos de constantes innovaciones tecnológicas. Esto se debe a que las técnicas de automatización están muy ligadas a los sucesos económicos mundiales. El uso de robots industriales junto con los sistemas de diseño asistidos por computadora (CAD), y los sistemas de fabricación asistidos por computadora (CAM), son la última tendencia en automatización de los procesos de fabricación. Éstas tecnologías conducen a la automatización industrial a otra transición, de alcances aún desconocidos. Que poco a poco van adquiriendo más terreno dentro de las demás ciencias. Aunque el crecimiento del mercado de la industria Robótica ha sido lento en comparación con los primeros años de la década de los 80's, de acuerdo a algunas predicciones, la industria de la robótica está en su infancia. Ya sea que éstas predicciones se realicen completamente, o no, es claro que la industria robótica, en una forma o en otra, permanecerá.(Molina(2015))

Figura 2.1: Brazo Robótico
Fuente:(Association(2017))

2.2. Robot Hexápodo.

El robot hexápodo es un vehículo que se desplaza o camina sobre seis extremidades, es un robot que tiene buena estabilidad ya que su centro de gravedad se encuentra cerca del suelo. En el caso de que una de las extremidades no funcionara el robot es capaz de seguir caminando. Este tipo de robot se basa en la biología de animales, en este caso del insecto hormiga que es de la familia de las Himenópteras que incluyen avispas y abejas. Los modelos que existen realizados a nivel mundial pueden tener varias aplicaciones para simular la biología de los seres vivos, en los que se basa el proyecto realizado esta el ANT 1 HEX. Uno de los proyectos destacados es el Mihro este robot controla 21 servomotores. Este robot incluye la capacidad de adaptarse al suelo debido a sensores de contacto que tiene en cada una de las articulaciones, también posee la capacidad de detectar los obstáculos.

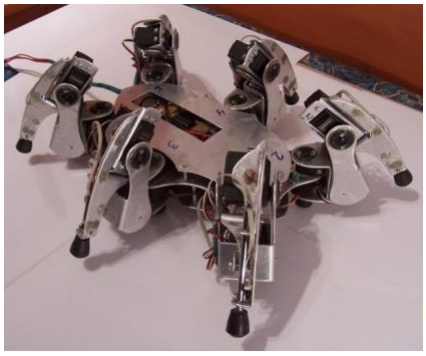


Figura 2.2: Robót Mihro
Fuente:(Pedrosa Lojo(2008))

2.3. Microcontrolador.

Los microcontroladores son dispositivos electrónicos que permiten realizar y ejecutar procesos como el de una computadora, porque contienen CPU, RAM y ROM, puertos de entrada y salida como periféricos. Existen varios tipos de microcontroladores, los más comunes son los Microchip y Atmel.
Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador. Un microcontrolador dispone normalmente de los siguientes componentes:

- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM para Contener los datos.
- Memoria para el programa tipo ROM/PROM/EPROM.
- Líneas de E/S para comunicarse con el exterior.

- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

Figura 2.3: Microcontrolador
Fuente:(Alvarez(2016))

Los productos que para su regulación incorporan un microcontrolador disponen de las siguientes ventajas:

- Aumento de prestaciones: un mayor control sobre un determinado elemento representa una mejora considerable en el mismo.
- Aumento de la fiabilidad: al reemplazar el microcontrolador por un elevado número de elementos disminuye el riesgo de averías y se precisan menos ajustes.
- Reducción del tamaño en el producto acabado: La integración del microcontrolador en un chip disminuye el volumen, la mano de obra y los stocks.
- Mayor flexibilidad: las características de control están programadas por lo que su modificación sólo necesita cambios en el programa de instrucciones.

El microcontrolador es en definitiva un circuito integrado que incluye todos los componentes de un computador. Debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna. En este caso el controlador recibe el nombre de controlador empotrado (embedded controller).(J. M. Angulo Usategui(1999))
Aplicaciones de Microcontrolador:

- Robótica. Controladores de robot, comunicaciones, control se sensores, etc.
- Informática.- Impresoras, discos duros, lectores de CD.
- Industria. Plantas de procesamiento.
- Automóviles. Control de alarmas, luces, control de velocidad.

En el mercado hay varios tipos de microcontroladores, dependiendo del uso inclusive el mismo micro puede ser encontrado en varios presentaciones. A esto se le llama tipos de encapsulados. Como SOIC (Plastic Small Outline), SSOP (Plastic Shrink Small Outline), TQFP (Plastic Thin Quad FlatPack).

Figura 2.4: Microcontrolador
Fuente:(Alvarez(2016))

2.4. Arduino UNO.

Es una placa de entrenamiento para poder conectar los periféricos de manera más fácil. Es una placa de microcontrolador basada en la ATmega328P. Tiene 14 pines digitales de entrada / salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio.

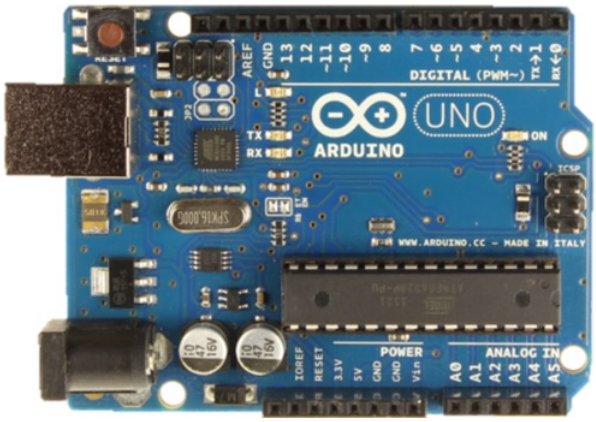


Figura 2.5: Arduino Uno
Fuente:(Arduino et al.(2014)Arduino, Ethernet, and Android)

Fue diseñado para gente que quiere hacer una introducción a la electrónica sin

necesidad de tener conocimientos previos en ciencias. Se creó esta plataforma para que estudiantes en diseño industrial pudieran tener acceso a unos conocimientos cerrados a ellos hasta entonces. Cabe decir que todo el sistema de desarrollo, el software, los circuitos y la documentación son abiertos. Esto permite a cualquier persona reproducir el sistema y usarlo en beneficio propio.

Las plataformas Arduino están basadas en los microcontroladores AVR en sus diferentes presentaciones como ser Atmega168, Atmega328, Atmega1280, ATmega8 y otros similares, chips sencillos y de bajo coste que permiten el desarrollo de múltiples diseños. (Arduino et al.(2014)Arduino, Ethernet, and Android)

TIPOS DE ARDUINO.

Las placas de Arduinos más conocidos:

Arduino UNO rev.3 Esta es la última revisión de la placa Arduino USB básica con cambios en el diseño pero conservando la misma funcionalidad que los modelos anteriores. Se conecta al ordenador con un cable USB estándar y contiene todo lo necesario para programar la placa. Se puede ampliar con gran variedad de shields: placas de extensión con funcionalidades específicas.

Arduino Ethernet Esta revisión de la placa Arduino básica. Se conecta al ordenador con un cable RJ-45 es decir a una red Ethernet (para poder usarse también INTERNET) contiene todo lo necesario para programar la placa. Se puede ampliar con gran variedad de shields: placas de extensión con funcionalidades específicas.

Arduino Android Esta revisión de la placa Arduino USB básica. Se conecta al ordenador con un cable USB estándar y contiene todo lo necesario para programar la placa, diseñada para usar como entorno de programación dispositivos con android (celulares y tabletas), de igual manera como la mayoría de los modelos, se puede ampliar con gran variedad de shields: placas de extensión con funcionalidades específicas.

Duemilanove Esta revisión de la placa Arduino USB básica. Se conecta al ordenador con un cable USB estándar y contiene todo lo necesario para programar la placa. Se puede ampliar con gran variedad de shields: placas de extensión con funcionalidades específicas.

Diecimila Esta es la revisión anterior de la placa USB básica.

Nano Una placa compacta diseñada para usar directamente en placas de desarrollo, el Nano se conecta al ordenador con un cable Mini-B USB.

Mega Mas grande y potente placa Arduino, compatible con los shields de Duemilanove y Diecimila.

Bluetooth El Arduino BT contiene un módulo bluetooth que permite comunicarse y programarse sin cables. Es compatible con los shields de Arduino.

Características del Arduino UNO:

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

2.5. Servomotores.

Los servomotores son dispositivos electromecánicos que consisten en un motor eléctrico, un juego de engranes y una tarjeta de control, todo implementado dentro de una carcasa de plástico. La característica principal de estos motores es que la gran mayoría no están hechos para dar rotaciones continuas ya que principalmente son dispositivos de posicionamiento en un intervalo de operación. Estos motores tienen un elevado torque por el mismo motivo de que giran en un ángulo restringido.

Funcionamiento del Servomotor

Para poder controlar el movimiento del servomotor se debe alimentar la señal de control mediante una señal PWM. Que se lo puede generar a través de un hardware. Para realizar el giro, la frecuencia del PWM debe estar constante. Esto depende de las especificaciones del fabricante, normalmente los motores trabajan como se muestra en la figura.

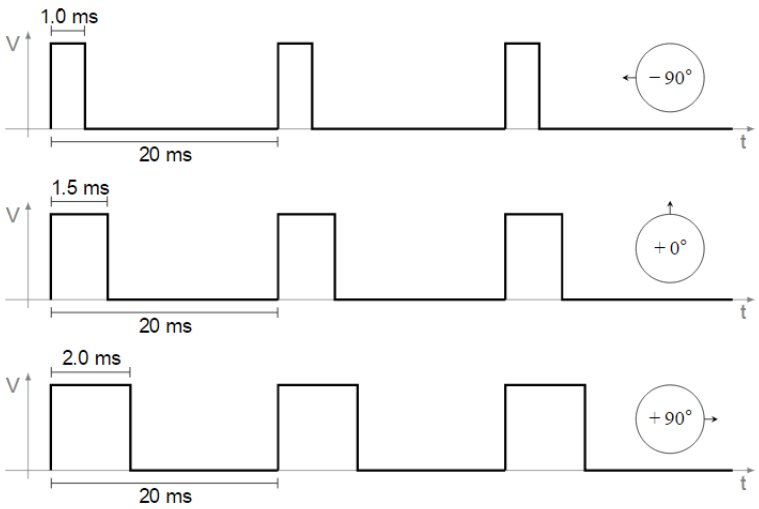


Figura 2.6: Señal PWM Servo
Fuente:(Daware(2015))

Tipos de Servomotores.

Los Principales son:

- **Servomotores de corriente continua (cc).**
  Son los servomotores que son más utilizados . funcionan con un pequeño motor de corriente continua. El servomotor se controla a través de una señal PWM (modulación por ancho de pulso).

Figura 2.7: Señal PWM Servo
Fuente:(Hitec(2017))

- **Servomotores de corriente alterna (AC).**
  Pueden utilizar corrientes más potentes y por lo tanto se usan para mover grandes fuerzas. En este tipo de motor se puede alimentar con circuitos monofásicos y trifásicos. Para poder variar la velocidad, inversión de giro y el torque de los motores AC se debe utilizar variadores de frecuencia

- **Servomotores de imanes permanentes o Brushless.**
  Se llama motor brushless por que es un motor de corriente alterna sin escobillas. Se utilizan para grandes torques o fuerzas y para altas velocidades. Son los más usados en la industria. Están basados en los motores síncronos.

- **Motor Paso a Paso.**
  Este es un tipo de motor eléctrico, sino que gira un paso a la vez. No giran de manera continua sino por pasos, es decir, giran un numero determinado de grados dependiendo del motor. La característica principal de estos motores es que se puede controlar la velocidad, inversiones de giro y el freno. Su control se basa en polarizar las bobinas que llevan incluidas de manera adecuada para que giren correctamente. Son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. Como por ejemplo en brazos robóticos de alta presición.

2.6. Wifi.

Es un tipo de tecnología inalámbrica mediante la cual se puede transmitir voz y datos. En el cual se puede enviar mucha cantidad de tráfico. Dicha tecnología se la puede utilizar en ambientes cerrados o en exteriores debido a su amplio alcance. Uno de los

problemas es la cantidad de usuarios que utilizan este tipo de tecnología y la saturación de espectro radioeléctrico, por tanto esta reducido a distancias cortas ya que al aumentar la distancia de conexión está expuesto a interferencias. Otro de sus problemas es la falta de se seguridad en las conexiones y los constantes intentos de acceder a la red por parte de terceros.
Las tecnologías específicas utilizadas por los equipos WiFi incluyen 802.11a, b, g, y n.
802.11n fue ratificado por IEEE en septiembre 2009, es un estándar muy reciente.
802.11g es compatible con 802.11b, y 802.11n es compatible con 802.11a cuando opera a 5 GHz, y con b/g en la banda de 2.4 GHz.
802.11n puede utilizar dos canales adyacentes de 20 MHz, para un total de 40MHz lo que no está contemplado en los estándares anteriores, y de esta manera puede alcanzar rendimientos reales superiores a 100 Mbps. El estándar permite inclusive mejorar esta cifra usando múltiples flujos de datos y ya existen equipos que utilizan esta modalidad.
802.11a,b, y g son ahora parte del estándar IEEE 802.11-2007 que comprende todas las enmiendas ratificadas hasta ese año, incluyendo 802.11e que permite QoS (calidad de Servicio). (Ermanno(2010)).

2.7. Cámaras Wifi Direct

El Wi-Fi Direct se puede ver como un Bluetooth actualizado y más rápido. Es un tipo de conexión limitada en espacio (no es para enviar fotos a nuestros familiares que viven en otra ciudad) y pensada para compartir cosas sencillas.

Por ejemplo, imprimir una archivo desde el portátil ó el smartphone con una impresora inalámbrica, enviar una imagen al móvil de una persona que está en la misma habitación, o por ejemplo, reproducir en la pantalla del televisor un vídeo que tenemos guardado en el móvil. Todo esto sin cables y sin internet.

Es útil cuando se quiere recibir o enviar alguna información a alguien pero se desea dar datos personales. Por ejemplo, cuando se está en un lugar donde hay varias personas, se hace un pequeño grupo entre desconocidos y se envía fotos o videos. Este tipo de conexión es muy útil cuando se desea compartir información. El WiFi Direct permite compartir esos archivos mucho más rápido que por Bluetooth, sin gastar datos de internet y sin exponer datos privados.

El Wi-Fi Direct está presente en muchos dispositivos. Teléfonos, cámaras, impresoras, ordenadores, consolas de videojuegos y televisores se pueden conectar con este protocolo, que a su vez es muy seguro, ya que está protegido con el último cifrado para redes.

Un dispositivo con WiFi Direct activado emite una señal hacia otros artefactos en la sala, haciendo saber que está disponible para una conexión. Los usuarios pueden enviar una invitación o recibir una, para efectuar dicha conexión. Cuando ambos o más

dispositivos está conectados (pueda que pida una clave, disponible en el mismo menú) se puede empezar a compartir archivos. (abc tecnoloía Madrid(2015))

Tecnología Wi-Fi Direct, tal como se describe en ”Especificación técnica Wi-Fi Peer-to-Peer (P2P)” Adopta un enfoque diferente, para mejorar la conectividad de dispositivo a dispositivo. En lugar de aprovechar el modo de operación ad-hoc, Wi-Fi Direct se basa en el modo de infraestructura IEEE 802.11 y permite que los dispositivos negocien quién asumirá el control de las funcionalidades similares a AP. Por lo tanto, permite que los dispositivos Wi-Fi heredados se conecten a la red Wi-Fi Direct que no hubiera sido posible de otra manera.(abc tecnoloía Madrid(2015))

Esta cámara se configura y maneja desde un dispositivo móvil o ordenador. Se puede conectar directamente por WIFI a un teléfono, o configurarla como P2P para que se conecte a la red WIFI de un router y después configurarla desde esa red. Es importante recordar que en el modo P2P o punto a punto la red WIFI del a cámara no es visible y se debe manejar desde un dispositivo conectado al mismo router que la cámara.(abc tecnoloía Madrid(2015))

Figura 2.8: Cámara P2P MD81S
Fuente: Aliexpress.com2015

Características

Nombre de la marca: JIUSION
Tipo: Mínimo
Soporte de alta definición: 480P
Tipo de Tarjeta de Memoria: Micro SD / TF
Tecnología del sensor: CMOS
Paquete: No
Tamaño de memoria integrada: No
Número de modelo: MD81 MD99S
Size: 55mm*28mm*20mm
Camera Shape: Pen
Retail Packages: None

Optical Zoom: Fixed Focus

2.8. Bluetooth.

La tecnología inalámbrica Bluetooth (BWT) fue desarrollada en 1994 en Ericsson en Suecia. El propósito original de BWT era eliminar la necesidad de conexiones de cables propietarias entre dispositivos tales como PDAs y PCs portátiles. Aunque la comunicación por infrarrojos existió en ese momento, esto requería línea de contacto de señal directa. Por lo tanto Ericson optó por una solución más barata y de bajo consumo incorporado a los dispositivos, haciendo posible que se pueda conectar dispositivos a través de paredes y otros materiales no metálicos. Ericsson empezó a trabajar en BWT, el concepto se convirtió en una tecnología de radio que simultáneamente conecta varios dispositivos en una red de Área personal. Debido al potencial ilimitado de BWT, el Grupo de Interés Especial de Bluetooth (SIG) se formó en 1998 para desarrollar la Especificación Bluetooth IEEE 802.15. La especificación normalizó el desarrollo de dispositivos compatibles con BWT para que los dispositivos de diferentes fabricantes puedan trabajar juntos.

Los dispositivos compatibles con BWT operan en la banda industrial, científica, médica (ISM) de 2,4 GHz (GHz) sin restricciones. La banda ISM oscila entre 2.400 GHz y 2.483 GHz. Los dispositivos habilitados para BWT usan setenta y nueve frecuencias de 1 megahertz (de 2.402 a 2.480 GHz) en la banda ISM como se muestra en la Figura 2.9. Los dispositivos habilitados para BWT usan una técnica llamada salto de frecuencia para minimizar el espionaje e interferencia de otras redes que usan el Banda ISM. Con salto de frecuencia, los datos se dividen en pequeños pedazos llamados paquetes. El transmisor y el receptor intercambian un paquete de datos a una frecuencia, y luego saltan a otra frecuencia para intercambiar otro paquete. Repiten este proceso hasta que se transmitan todos los datos.(Company(2004))

Topología de Red Bluetooth.

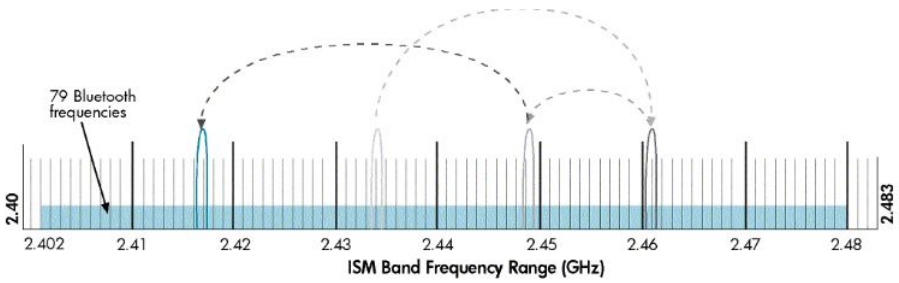


Figura 2.9: Tecnología Bluetooth
Fuente:(Company(2004))

Los dispositivos habilitados para BWT forman topologías de red llamadas piconets y scatternets. Un piconet consta de hasta ocho dispositivos habilitados para BWT (Figura

2.10). Cuando se establece una piconet, un dispositivo establece el patrón de salto de frecuencia y los otros dispositivos sincronizan sus señales con el mismo patrón. El dispositivo que establece el patrón de salto de frecuencia se denomina dispositivo primario y los otros dispositivos se denominan dispositivos secundarios. Cada piconet tiene un patrón de salto de frecuencia diferente para diferenciar sus señales de las señales de otras piconets. La comunicación bluetooth tienen todos los dispositivos smarthphone pa-

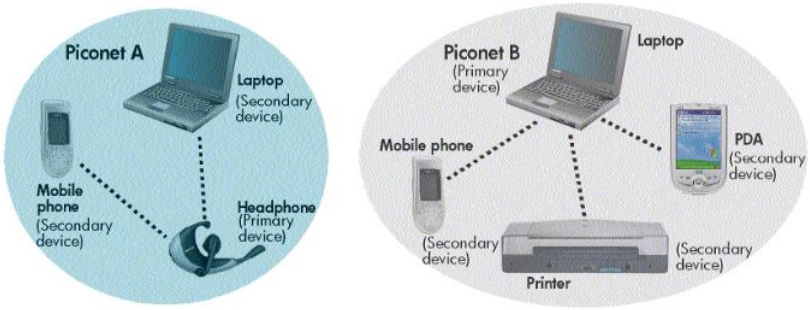


Figura 2.10: Tecnología Bluetooth
Fuente:(Company(2004))

ra poder recibir y transmitir información con otros equipos que están equipados con esta tecnología. El bluetooth es una comunicación de corta distancia y maneja una velocidad de comunicación media, dependiendo de la versión que utiliza que va desde los rangos 1Mbit/s a 32 Mbit/s.

2.8.1. Módulo Bluetooth

El módulo HC-05 es un módulo Bluetooth SPP (protocolo de puerto serie) fácil de usar, diseñado para la configuración de la conexión serial inalámbrica. El módulo Bluetooth de puerto serie está especificado como Bluetooth V2.0 + EDR (Enhanced Data Rate) Modulación de 3Mbps con transceptor de radio 2.4GHz y banda base completos. Utiliza CSR Bluecore04 Sistema de chip único externo de chip con tecnología CMOS y con AFH (función de salto de frecuencia adaptable). Tiene un encapsulado de 12.7mmx27mm. HC-05 Es un módulo de comunicación en serie Bluetooth tiene dos modos de trabajo: modo de trabajo de orden de respuesta y modo de trabajo de conexión automática. Y hay tres funciones de trabajo (Master, Slave y Loopback) en el modo de trabajo de conexión automática. Cuando el módulo está en el modo de trabajo de conexión automática, seguirá el modo predeterminado de configuración para transmitir los datos automáticamente. Cuando el módulo está en el modo de trabajo orden-respuesta, el usuario puede enviar el comando AT al módulo para establecer los parámetros de control y orden de control enviado. El modo de trabajo del módulo puede conmutarse controlando el nivel de entrada del PIN del módulo (PIO11).

Especificaciones de Hardware

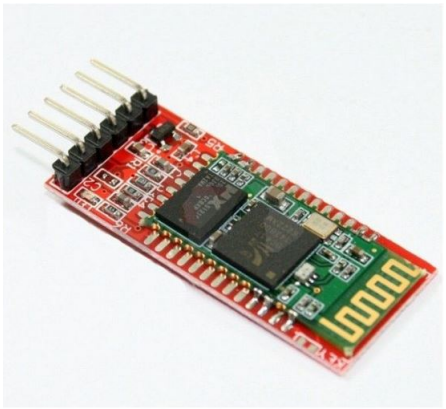


Figura 2.11: Módulo Bluetooth HC-05
Fuente: ElectrónicosCaldas.com

- Sensibilidad -80dBm
- Hasta +4dBm RF poder de transmisión
- Bajo voltaje de operación 1.8V a 3.6V I/O
- Control PIO
- Interface UART con velocidad de baudios programable
- Antena Integrada
- Conector edge

Especificaciones de Software

- Configuración de Fabrica: 38400 b, Data bits:8, Stop bit:1,Parity:No parity
- Tipos de Baudios 9600,19200,38400,57600,115200,230400,460800.
- Chip de radio: CSR BC417143
- Potencia de emisión: menor que 4 dBm, Clase 2
- Seguridad: Autenticación y encriptación (Password por defecto: 1234)
- Módulo montado en tarjeta con regulador de voltaje y 6 pines suministrando acceso a VCC, GND, TXD,
- RXD, KEY y status LED (STATE)
- Consumo de corriente: 50 mA
- Temperatura de operación: -20 °C a +75 °C
- Alcance 5m a 10m

2.9. Sensores Gas

Estos sensores son electroquímicos y varían su resistencia cuando se exponen a determinados gases, internamente posee un calentador encargado de aumentar la temperatura interna y con esto el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia. El calentador dependiendo del modelo puede necesitar un voltaje entre 5 y 2 voltios, el sensor se comporta como una resistencia y necesita una resistencia de carga (RL) para cerrar el circuito y con este hacer un divisor de tensión.

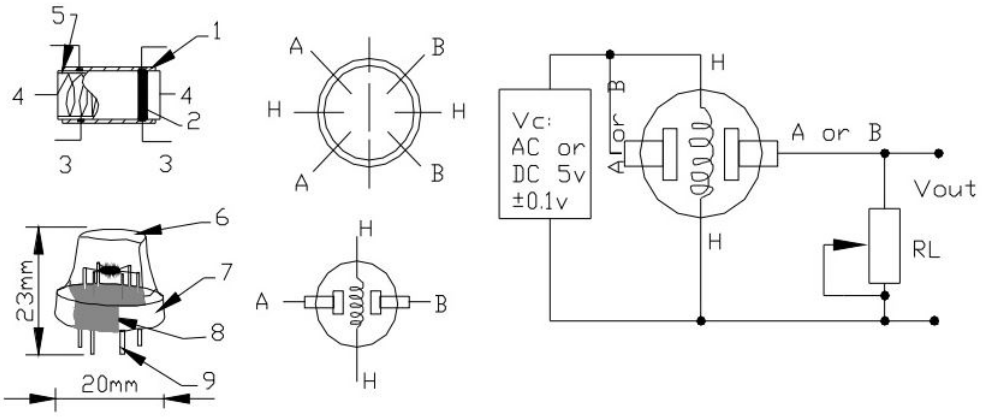


Figura 2.12: Diagrama Conexión MQ2
Fuente:(Mechatronics(2016))

Sensor de gas combustible y humo MQ-2. Este tipo de sensores son adecuados para detectar GLP, propano, metano, alcohol, hidrógeno, humo. Siendo más sensible al GLP y propano.

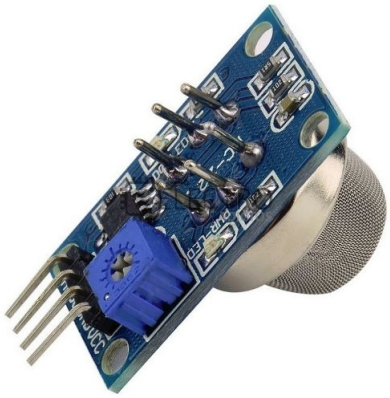


Figura 2.13: Sensor MQ2
Fuente:(Mechatronics(2016))

Escalamiento de los sensores MQ.

Si en la aplicación que se está implementando se necesita los valores en unidades correspondientes a la medición del gas, se debe escalar el valor leído, el problema de supone la relación entre la lectura analógica y el valor real no es lineal. Por lo que se requiere estimar la curva que da el datasheet.

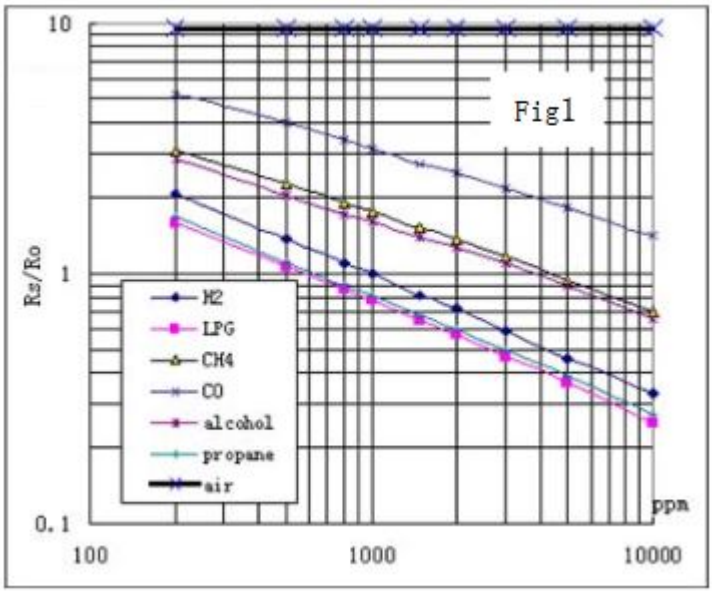


Figura 2.14: Curva Especificación MQ2
Fuente:(Mechatronics(2016))

2.10. Sensores Temperatura y Humedad

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Es bastante simple de usar tanto en hardware como software. El único inconveniente de este sensor es que sólo se puede obtener nuevos datos una vez cada 2 segundos.

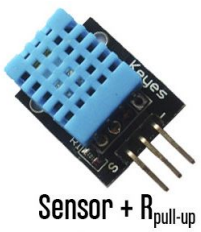


Figura 2.15: Sensor DHT-11 MQ2
Fuente:(Mechatronics(2016))

El sensor DHT11 se caracteriza por tener la señal digital calibrada, asegurando alta estabilidad y fiabilidad a lo largo del tiempo. El sensor integra un sensores resistivos pa-

ra temperatura (termistor) y otro para humedad. Puede medir la humedad en un rango desde 20 % hasta 90 % y temperatura en el rango de 0 °C a 50 °C.

Cada sensor DHT11 está estrictamente calibrado en laboratorio, presentando una extrema precisión en la calibración. Los coeficientes de calibración se almacenan como programas en la memoria OTP, que son empleados por el proceso de detección de señal interna del sensor.

El protocolo de comunicación emplea un único hilo o cable, por lo tanto hace que la integración de este sensor en nuestros proyectos sea rápida y sencilla.
En comparación con el DHT22, este sensor es menos preciso, menos exacto y funciona en un rango más pequeño de temperatura / humedad, pero su empaque es más pequeño y de menor costo

2.11. Regulador de Voltaje

DC-DC regulador tension 5A

El regulador tension sirve para alimentar a los dispositivos con el voltaje se desea, de una manera estable. Se necesita una fuente de alimentación 2v por encima del voltaje que se desea regular. Este tipo de regulador funciona hasta 5 amperios de carga.

DC-DC STEP DOWN Adjustable Power Supply Module Converter 0.8V-24V 5A Max.

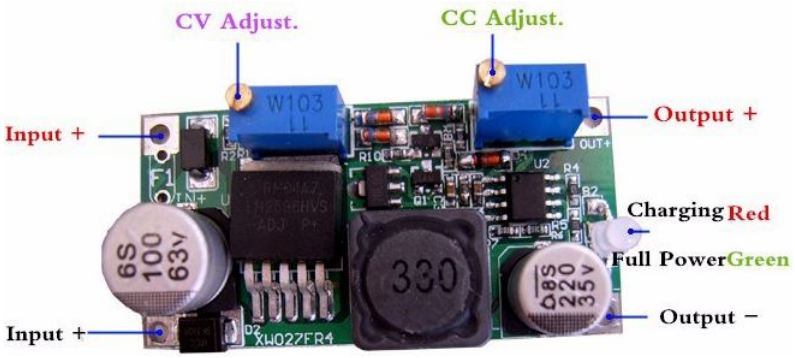


Figura 2.16: Regulador de Voltaje 5a
Fuente:(Mechatronics(2016))

Especificaciones:

- Voltaje de entrada: DC 5V - 30V

- Entrada a 6v >Salida máxima 12v
- Entrada a 12v >Salida máxima 24v
- Tensión de salida: DC 0.8V-24V (ajustable, salida <Entrada)
- Corriente de salida: corriente nominal es de 2,5 A,
- (5AMax,Si la corriente es mayor que 2,5, se requiere la mejora de la disipación de calor.)
- Ondulación de la salida: 30mV (máximo)
- Regulación de la carga: $\pm$ 0.5 %
- Regulación de voltaje: $\pm$ 2.5 %
- Temperatura de funcionamiento:-40 °C 85 °C

2.12. Baterias de Litio

El plomo es un metal muy pesado y por lo tanto la investigación se hizo hace muchos años para hacer una mejor batería que sea más liviana en peso, así como que ha mejorado la densidad de potencia. Entonces el litio se convirtió en la elección lógica para reemplazar el plomo, ya que es el metal más ligero disponible en el mundo. El litio no sólo es ligero sino también altamente reactivo y por esta razón el litio puro nunca se encuentra en la naturaleza. El metal de litio se fabrica a partir de sales de litio que se extraen de las actividades mineras. En el mundo de hoy las baterías de iones de litio se utilizan en casi todos los aparatos incluyendo portátil, teléfono celular, cámara, iPod y muchos más dispositivos y por lo tanto son muy populares en estos días. Son una de las baterías más enérgicas disponibles hoy que las hacen tan populares. Nuestra concentración principal está en las baterías del fosfato del hierro del litio y del cobalto del litio Las pilas. Las baterías del fosfato del hierro del litio fueron desarrolladas por una persona en la universidad de Tejas. Aunque primero fue comercializado por otros, las baterías de iones de litio fueron fabricadas por primera vez por A123, mediante la preparación de baterías de litio nanofosfato dopado en el 2006. Hoy la empresa A123 ha hecho un buen progreso en su tecnología de baterías y fabrica baterías de amplia gama que sirve a bicicletas eléctricas a servidores de computadoras

Batería recargable de Li-Ion de 5V 9000mAh y 12V 4500mAh. La batería mide unos 33x85x25mm y pesa unos 140gr. Dispone de interruptor de encendido/apagado y led de actividad para saber cuando esta en marcha. Conector de carga standard de 2.1mm, salida simultanea de 5V (conector hembra) y de 12V por el mismo conector de carga de 2.1m (macho). Modelo YSD-12-5, Input 12.6VDC, Output 5VDC 9000mAh y 12VDC 4500mAh. Securame.com2016

Figura 2.17: Bateria de Litio 12V-5V
Fuente:(Securame.com,2016)

2.13. SOFTWARE

2.13.1. MPLAB ASSEMBLER

Cada computadora personal tiene un microprocesador que gestiona las actividades aritméticas, lógicas y de control de la computadora. Cada familia de procesadores tiene su propio conjunto de instrucciones para manejar varias operaciones como obtener entrada desde el teclado, mostrar información en la pantalla y realizar varios otros trabajos. Este conjunto de instrucciones se denominan ”instrucción de lenguaje de máquina”. El procesador entiende sólo instrucciones de lenguaje de máquina que son cadenas de 1s y 0s. Sin embargo, el lenguaje de máquina es demasiado oscuro y complejo para su uso en el desarrollo de software. Así, el lenguaje de ensamblaje de bajo nivel está diseñado para una familia específica de procesadores que representa varias instrucciones en código simbólico y una forma más comprensible.

Ventajas del lenguaje de Assembler

Una comprensión del lenguaje ensamblador proporciona el conocimiento de:

- Interfaz de programas con OS, procesador y BIOS;
- Representación de datos en memoria y otros dispositivos externos;
- Cómo el procesador accede y ejecuta la instrucción;
- Cómo las instrucciones acceden y procesan los datos;
- Cómo un programa accede a dispositivos externos. Otras ventajas del uso del lenguaje ensamblador son:
- Requiere menos memoria y tiempo de ejecución;
- Permite trabajos complejos específicos de hardware de una manera más fácil;

- Es adecuado para trabajos de tiempo crítico;

Figura 2.18: MPLAB IDE Microchip
Fuente: (Programa MPLAB,V8.92)

MPLAB IDE es un software que permite el desarrollo de proyectos de aplicación para los PIC o microcontroladores de Microchip, Puede funcionar en multiplataforma como Linux, WIndows y Mac OS. Este Software permite realizar simulaciones y compilaciones completas y también paso a paso, lo cual es de gran ayuda para los desarrolladores de programas para PIC.

El programa incluye un editor de texto, macro-ensamblador, compilador ANSI C, y simulador para trabajar con cualquier microcontrolador PIC. El simulador puede operarse tanto en programas desarrollados en lenguaje ensamblador o ANSI C. Al realizar la simulación paso por paso, el usuario elige de un menú de opciones, cuáles registros y funciones desea observar. Por ejemplo se puede verificar el valor que contienen las variables en determinado momento del código de programación. Se pueden colocar puntos

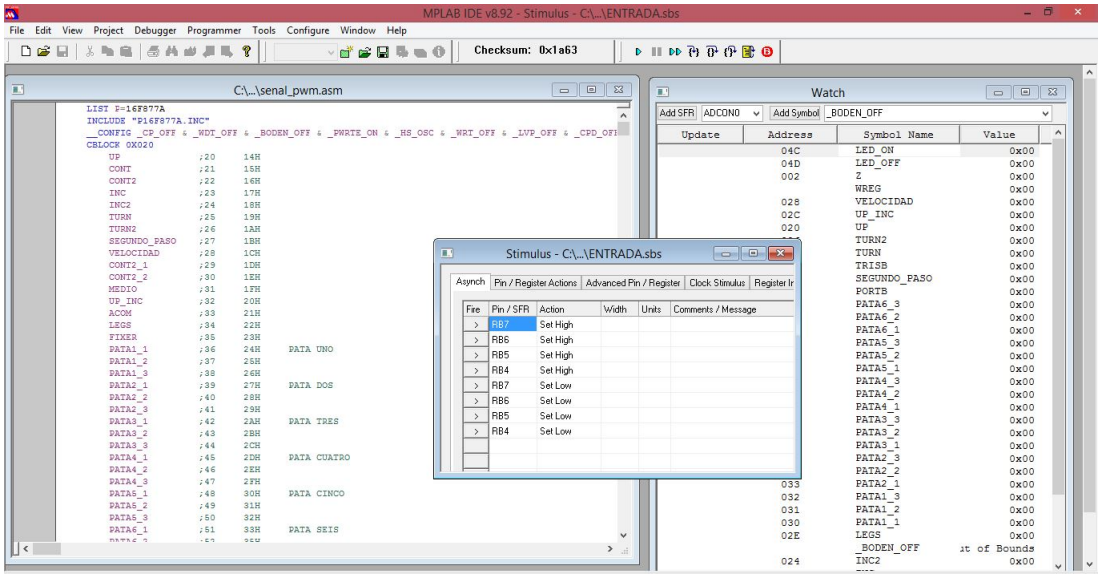


Figura 2.19: Ventana de MPLAB IDE Microchip
Fuente: (Programa MPLAB,V8.92)

de parada durante la ejecución del simulador. En una ventana se observa el contenido de la memoria de datos, en otra, el contenido de los registros especiales y en una última, un

cronómetro "stopwatch.[en] la cual se marca el tiempo de ejecución durante la simulación.

Con la tecla F7, se avanza hacia la siguiente instrucción y automáticamente se actualiza el contenido de las ventanas. Existe también una opción (seleccionar: "debugger", "stimulus") para generar entradas digitales durante la simulación en cualquier bit de los puertos. Para poder simular ingreso de datos periféricos.

2.13.2. Android Studio

Android es un sitema operativo de código abierto para dispositivos móviles, se programa principalmente en Java, y su núcleo está basado en Linux.
Android - tanto el sitema operativo, como la plataforma de desarrollo - están liberados bajo la licencia de Apache. Esta licencia permite a los fabricantes añadir sus propias extensiones propietarias, sin tener que ponerlas en manos de la comunidad de software libre. Al ser de open source, Android hace posible:

Figura 2.20: Logo Android
Fuente:(CCIA(2014))

- Una comunidad de desarrollo, gracias a sus completas APIs y documentación ofrecida.
- Desarrollo desde cualquier plataforma (Linux, Mac, Windows, etc).
- Un sistema operativo para cualquier tipo de dispositivo móvil, al no estar diseñado para un sólo tipo de móvil.
- Posibilidad para cualquier fabricante de diseñar un dispositivo que trabaje con Android, y la posibilidad de abrir el sistema operativo y adaptarlo o extenderlo para su dispositivo.
- Valor añadido para los fabricantes de dispositivos: las empresas se ahorran el coste de desarrollar un sistema operativo completo para sus dispositivos.
- Valor añadido para los desarrolladores: los desarrolladores se ahorran tener que programar APIs, entornos gráficos, aprender acceso a dispositivos hardware particulares, etc.

Android está hecho de un núcleo basado en el de Linux para el manejo de memoria, procesos y hardware. Contiene bibliotecas open source para el desarrollo de aplicaciones. Un entorno de ejecución para las aplicaciones Android. La máquina virtual Dalvik y las bibliotecas específicas dan a las aplicaciones funcionalidades específicas de Android.
Como Android tiene un sistema operativo abierto, los fabricantes pueden corregir algún desperfecto o mal funcionamiento del equipo. Sin tener que recurrir a los desarrolladores de Android. El desarrollo para dispositivos móviles y embebidos requiere que el programador tenga especial cuidado en determinados aspectos. El sistema operativo no puede encargarse de controlar todo porque limitaría demasiado al programador, así que determinados aspectos como ahorrar CPU y memoria son responsabilidad del programador en la mayoría de los casos. La limitaciones de hardware que el dispositivo impone suelen ser:

- Pequeña capacidad de procesamiento
- Memoria RAM limitada
- Memoria permanente de poca capacidad
- Pantallas pequeñas de poca resolución
- Transferencias de datos costosa (en términos de energía y económicos) y lenta
- Inestabilidad de las conexiones de datos
- Batería muy limitada
- Necesidad de terminar la aplicación en cualquier momento

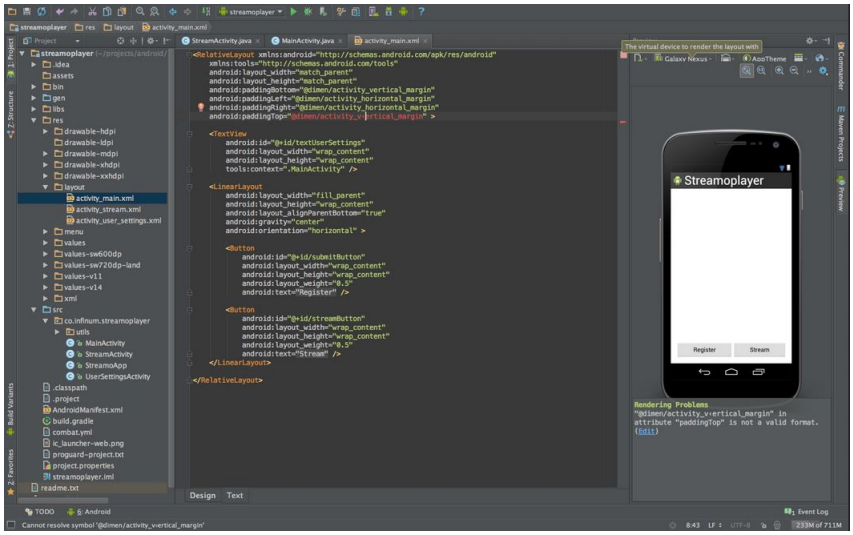


Figura 2.21: Logo Android
Fuente:(CCIA(2014))

2.13.3. Arduino

Arduino es una plataforma de hardware programable y flexible diseñada para artistas, diseñadores, manipuladores y fabricantes de cosas. La pequeña placa de circuito azul de Arduino. que lleva el nombre mítico de una publicación local en Italia, ha motivado en muy poco tiempo a una nueva generación de muchos colaboradores de todas las edades a hacer todo tipo de proyectos interesantes que se encuentran desde los laboratorios de universidades hasta los desarrolladores a nivel mundial. Por lo general, estos proyectos basados en Arduino requieren poca o ninguna habilidad de programación o conocimiento de la teoría de la electrónica, y más a menudo que no, esta facilidad de manejo es simplemente recogido en el camino. La plataforma Arduino es bastante útil

Figura 2.22: Arduino Versión
Fuente: (Evans(2011))

para proyectos de microcontroladores, pero eso no es suficiente para impulsar la popularidad y la adopción generalizada de la plataforma. En lugar de cerrar el diseño de la placa de interfaz y el entorno de desarrollo, todo el proyecto Arduino está profundamente arraigado en la práctica emergente de hardware de código abierto. A diferencia del software de código abierto, del que Linux suele ser el ejemplo a menudo citado, el hardware de código abierto busca la colaboración donde los objetos físicos son el resultado. Involucra un modelo distribuido de desarrollo de hardware con contribuyentes que generalmente residen en diferentes partes del mundo. En lugar de sistemas cerrados, los proyectos de código abierto permiten a una persona acceder libremente a los archivos fuente de un diseño, hacer mejoras y redistribuir estas mejoras a una comunidad más grande.(Evans(2011)).

El ecosistema de Arduino representa fundamentalmente esta aspiración de apertura en el diseño, la arquitectura, la colaboración y la filosofía. Puede verlo por sí mismo ya que todos los archivos de diseño, esquemas y software están disponibles para descargar, usar, modificar, rehacer o incluso revender. Lo que comenzó como una decisión aparentemente desinteresada para abrir el diseño y el software de Arduino a la comunidad, impulsado por el cierre de la escuela de diseño donde se formó el equipo Arduino, ha dado lugar a un movimiento completamente nuevo en el diseño. La práctica de los contribuyentes que tienen la libertad de utilizar estos diseños libremente (libre como en el

Figura 2.23: Arduino
Fuente:(Evans(2011))

discurso) y sin la obligación de comprar cualquier cosa (libre como en cerveza) ayuda a hacer el Arduino tan entrañable como una colección de silicio y de cobre puede ser. Sin mencionar que este circuito de retroalimentación creativa garantiza que cada innovación inspirada derivada de la plataforma Arduino se satisfaga con usos cada vez más imaginativos para cosas aún más nuevas.

2.14. MARCO CONCEPTUAL

A través del estudio e investigación definida en el capítulo 1, se puede delimitar los conceptos y términos utilizados.

- Robot. Es una máquina que contiene varios sistemas(mecánico, electrónico e informático) que interactúan entre sí para realizar una tarea específica en reemplazo del ser humano. Hoy en día se utilizan mucho los robots para distintas tareas, es posible que en el futuro los robots ocupen varias plazas de trabajo que actualmente lo realiza el ser humano.
- Microcontrolador. Es un dispositivo puramente electrónico que contiene procesador, memoria y periféricos, dentro de un encapsulado plástico,como si fuera una pequeña computadora, que puede programarse para desempeñar una tarea conjuntamente con hardware externo.
- Tarjeta Electrónica Arduino UNO. Es una tarjeta integrada con todo los elementos electrónicos, que esta lista para usarse con conexión a una computadora, para programarse mediante el software gratuito. Arduino.
- . Servomotor. Entre los tipos de servomotores están los que se controlan, mediante ancho de pulso, que tienen aplicaciones orientadas a realizar trabajo de fuerza y con movilidad limitada, como timones de autos, aviones a control remoto, retroviso-res, robots. Tienen recubrimento de plástico y son pequeños. La ventaja de utilizar estos servomotores es que funcionamiento

- MQ2. Es un sensor de gases, que funciona censando el ambiente a través de un micro tubo de cerámica y una malla que se calienta, diseñado para trabajar con la tarjeta Arduino, ya que posee las librerías para realizar la comunicación serial.

- DHT11. Este sensor igualmente esta hecho para trabajar con la Tarjeta Arduino, su función es censar la temperatura y humedad que existe en el medio ambiente, dando una precisión de +-1 grado centígrado. Es conveniente utilizarlo para aplicaciones que no necesitan mucha precisión. Los sensores que se utilizan son del tipo educativo, ya que el funcionamiento se lo realiza a temperatura ambiente. Y el consumo de corriente y voltaje es bajo.

- Fuente de alimentación. En el mercado existen varios tipos de baterías, en tipo y tamaño. Esto depende mucho de la aplicación, dentro de los aspectos importantes que se toman en cuenta es el consumo y la durabilidad. De ahí su precio. Por ejemplo las baterias de plomo tienen buena durabilidad y son utilizadas para sistemas robustos, generalmente para vehículos, ups y sistemas de iluminación de emergencias.

- MPLAB. La programación básica de los microntroladores, es el lenguaje de programación assembler, por esta razón el fabricante ofrece las hojas de especifaciones o datasheets en este código, y garantizan el funcionamiento de los microcontroladores al utilizar Assembler. Entre los software de programación en assembler esta el MPLAB, una de las herramientas más potentes para realizar diversos proyectos, por tal motivo se utiliza en el diseño del presente proyecto. Aunque la programación es un poco compleja, es la mejor opción para desarrollar procesos repetitivos y con mayor precisión, que los demás lenguajes de programación.

CAPÍTULO II PROPUESTA

DISEÑO

3.1. Diseño Mecánico

El diseño mecánico está compuesto por el esqueleto (tórax) y por las extremidades o articulaciones. Para el diseño de motricidad del robot, puede realizar a través de las 6 articulaciones y estas cuentan con 3 grados de libertad por cada una. Mediante el modelo mecánico del robot, de manera que el peso está repartido en 6 puntos de contacto con el suelo.

3.1.1. Tórax

El tórax es la parte mecánica donde se acoplan las extremidades del robot, tienen forma rectangular y esta ensanchado en la mitad, por motivos de diseño personal y mejor estabilidad. Aquí se monta todo el hardware, la tarjeta de control de los servomotores , la tarjeta Arduino Uno, el módulo de comunicación WIFI, la fuente de alimentación que es la batería de litio, los servomotores para las extremidades, el servomotor de la cámara y el cableado de las conexiones entre los dispositivos electrónicos.

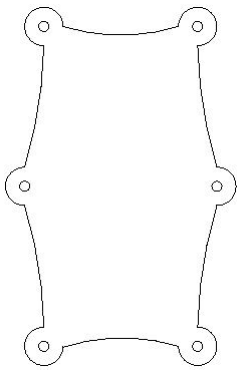


Figura 3.1: Tórax del Hexápodo.
Fuente: Elaborado por el Autor.

Articulaciones

Cada articulación cuenta con tres grados de libertad, para que el movimiento pueda tener un desplazamiento similar al de un vector en coordenadas esféricas, y simule la motricidad de un brazo mecánico. Cada articulación cuenta con hombro, codo y brazo. El hombro realiza los movimientos en dos ejes. Para poder crear este movimiento se acoplaron dos servomotores juntos a través de una cinta metálica, y para el codo uno servomotor mediante una extensión realizada en acrílico.

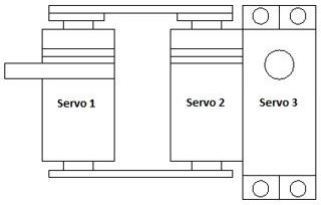


Figura 3.2: Vista en planta de una extremidad.
Fuente: Elaborado por el Autor.

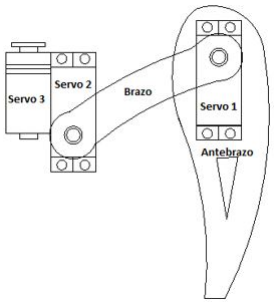


Figura 3.3: Vista frontal de una extremidad.
Fuente: Elaborado por el Autor.

Diseño de Motricidad

Existen varios tipos de desplazamiento en los que se refiere a motricidad sobre 6 extremidades. Se puede observar en la naturaleza como lo realiza una hormiga, una araña, insectos en general. Cada uno de ellos realiza de diferente manera, por ejemplo la araña; para poder desplazarse mueve una sola extremidad a la vez, mientras las otras están en contacto con el suelo, y realizan un ligero desplazamiento hacia adelante. Para este caso se realizó desplazamiento en forma de trípode, quiere decir que 3 extremidades están en el aire mientras que las otras 3 están en el suelo. De esta manera se logra 3 puntos de apoyo en el suelo que le dan una buena estabilidad al robot. Todas las extremidades no realizan el mismo movimiento, por la disposición que tienen con respecto al tórax.

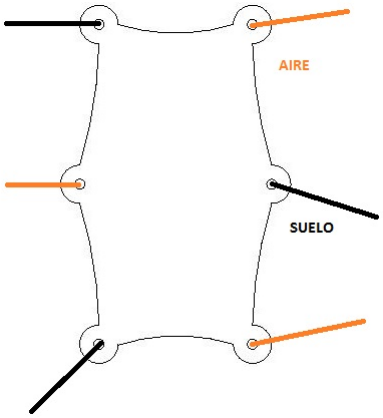


Figura 3.4: Desplazamiento del Robot Trípode-A.
Fuente: Elaborado por el Autor.

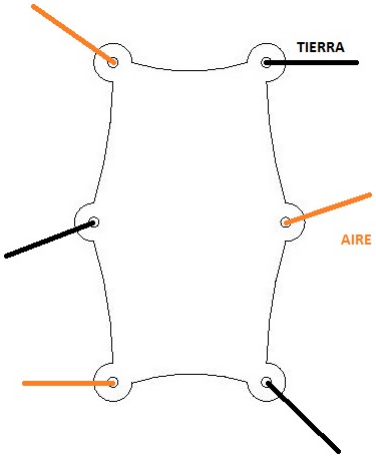


Figura 3.5: Desplazamiento del Robot Trípode-B.
Fuente: Elaborado por el Autor.

3.1.2. Acoplamiento de Motores

El acoplamiento de motores se realiza al unir dos servomotores en la unión del hombro con el tórax, mediante dos cintas metálica Se coloca de manera que los ejes del servomotor estén apuntado en dos planos opuestos perpendiculares entre sí. Para el caso en los ejes r y α.
Para el motor del antebrazo se unen mediante una extension en acrílico que unen el hombro con las terminaciones de contacto con el suelo.

3.1.3. Montaje de articulaciones al Tórax

Al armar todas las extremidades del robot, se tiene el primer diseño de robot para poder realizar pruebas de motricidad. Para poder acoplar todos los motores se utiliza láminas de metal, tornillos y tuercas. Una vez que el robot este armado se puede tomar los puntos de referencia de partida, ya que los servomotores no realizan el mismo

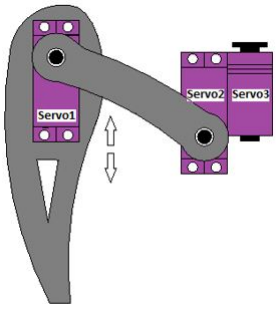


Figura 3.6: Acoplamiento de Motores.
Fuente: Elaborado por el Autor.

movimiento o tienen una misma posición inicial.

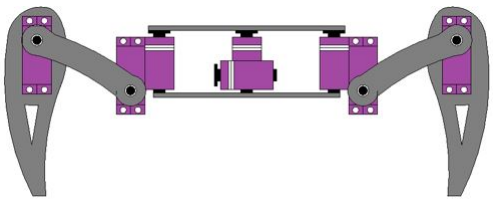


Figura 3.7: Vista Frontal Prototipo HEX.1.
Fuente: Elaborado por el Autor.

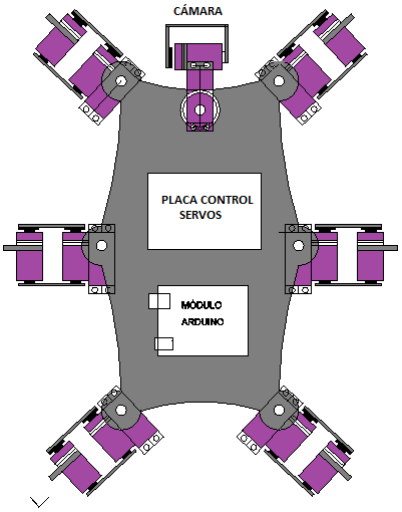


Figura 3.8: Vista Planta Prototipo HEX.1.
Fuente: Elaborado por el Autor.

3.2. Diseño Electrónico

Para el diseño electrónico se contemplan 4 etapas que conformarán todo el sistema para el funcionamiento del robot. Cada etapa tiene su desarrollo conforme las necesidades que se requieran, en lo que se refiere a hardware y el software.

1. Control de Servomotores PWM.
2. Control o mando Remoto.
3. Sensores Análogos.
4. Vídeo Transmisión.

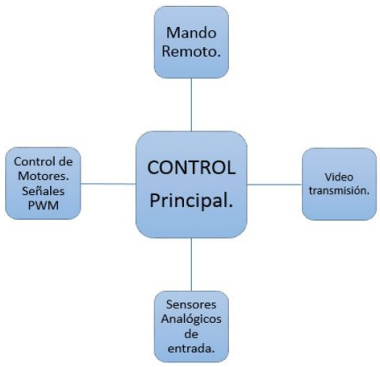


Figura 3.9: Diseño Electrónico.1.
Fuente: Elaborado por el Autor.

3.2.1. Control de Servomotores

Control PWM. La modulación por ancho de pulso se la realiza a través del PIC16F877A. Este microcontrolador se utilizó debido a la cantidad de entradas, salidas y su disposición para realizar el PCB, para realizar las conexiones hacia ambos lados, tiene una capacidad de procesamiento hasta 20 MHz, que sirve para realizar el control de las 20 señales PWM También porque permite el desempeño de multifunción. La programación en lenguaje Assembler es más fácil que otros dispositivos de la familia PIC18f. que contienen algunas características similares.

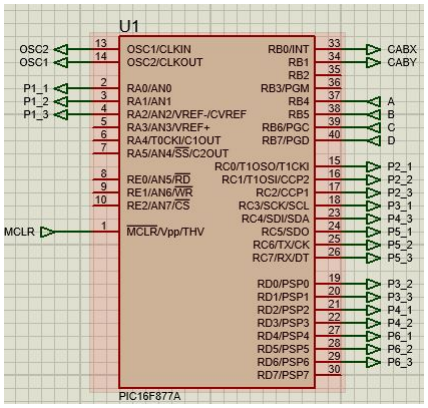


Figura 3.10: Microcontrolador PIC16f877A
Fuente: Simulacíon Proteus.

Para poder realizar el control de movimiento de los servomotores se debe variar el ancho de pulso desde 1 milisegundo hasta 2 milisegundos. Mediante pruebas se esta-

blece el punto medio de la posición del motor ya que este varía a veces por el fabricante de los motores. Como se muestra en la figura el ancho de pulso para el posicionamiento medio del motor está en 1.5 milisegundos.

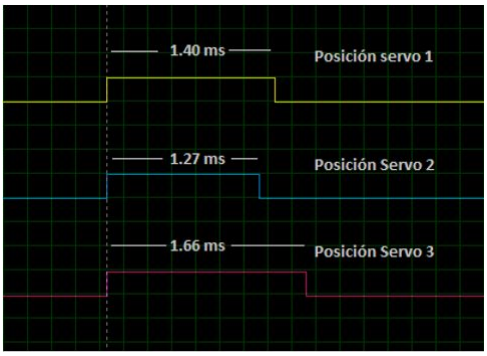


Figura 3.11: Generación PWM.
Fuente: Simulacíon Proteus.

En la generación de las 20 señales PWM a través del PIC con el uso de lenguaje de programación Assembler, se puede permitir controlar cada una de las señales, ya que se puede verificar la duración de las líneas de programación y así generar los respectivos anchos de pulso que necesita cada uno de los servomotores a través de un bucle general. Esto se realiza tomando en cuenta el nivel de procesamiento del PIC a 20 MHz.

3.2.2. Control Remoto.

Figura 3.12: Comunicación con el Robot.
Fuente: Elaborado por el Autor.

Para realizar el mando o control remoto, se implementa una comunicación bluetooth con un teléfono inteligente y el módulo bluetooth Arduino. Se implementa la interface en el software Android Studio. Desde esta aplicación se realizan los movimientos del robot al presionar los botones desarrollados en la interface. Y en los cajones de texto se mostrará las valores de las mediciones de los sensores. Inicialmente se presenta la información del proyecto. El título y la que empresa que implementa el sistema. Si el dispositivo no tiene encendido la comunicación BT, solicita que se encienda mediante una venta emergente.

En la siguiente pantalla se muestra los dispositivos bluetooth a conectarse. La lista que se despliega son los dispositivos que han sido conectados y pareados, con el propio

Figura 3.13: Presentación Aplicación Control.
Fuente: Elaborado por el Autor.

sistema software del Teléfono

Figura 3.14: Pantalla de Selección BT.
Fuente: Elaborado por el Autor.

3.2.3. Sensores

Sensor Gases MQ2. Este tipo de sensor permitirá realizar un análisis dentro de las cámaras de registro. Indicará que porcentaje de gas existe en la cámara. El porcentaje que mostrará determinará si es seguro ingresar al técnico a realizar las las tareas.

Temperatura y Humedad. Debido a que las cámaras de registro están bajo el nivel del suelo y pasan tapadas, pueden llegar a acumular una temperatura alta al igual que la humedad, estos factores pueden ser perjudiciales para el técnico, ya que al estar expuesto por mucho tiempo en esas condiciones pueden sufrir deshidratación. Con los valores medidos por el sensor se sabrá si es conveniente o no bajar a la cámara de registro. Este módulo se conecta a la tarjeta arduino a los pines de alimentación de 5V+ y GND, el pin de datos irá conectado al pin 5 en modo de funcionamiento digital. Con una librería propia del sensor llamada ”DHT.cpp”,la cual se la puede descargar desde internet, no necesita calibración lo cual facilita la comunicación con el mismo.

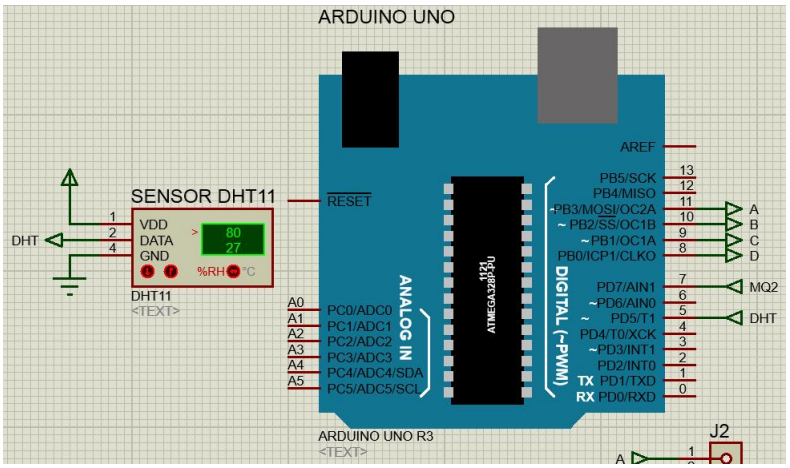


Figura 3.15: Conexión Sensor DHT11.
Fuente: Simulación Proteus.

3.2.4. Video

Figura 3.16: Operando dentro de la Cámara.
Fuente: Elaborado por el Autor.

Para poder evaluar en que condiciones se encuentra la cámara de registro el robot transmitirá el vídeo a través de la cámara tipo espía MD80S, la que cuenta con una ranura para memorias SD, lo que permitirá guardar las grabaciones para su posterior revisión.

3.2.5. Esquema de Conexiones

Para poder conectar todos los dispositivos, se realizó la conexión en paralelo de tal forma que todos los elementos reciban la misma cantidad de voltaje, para su funcionamiento. Hay que tomar en cuenta que al realizar muchas conexiones, y en el caso de realizar una revisión se utilizará colores que distingan la alimentación positiva, negativa y la señal de control. Como se muestra en la figura 3.17.

- Rojo. Voltaje Positivo

- Negro. Negativo o Tierra.
- Amarillo. Señal de Control

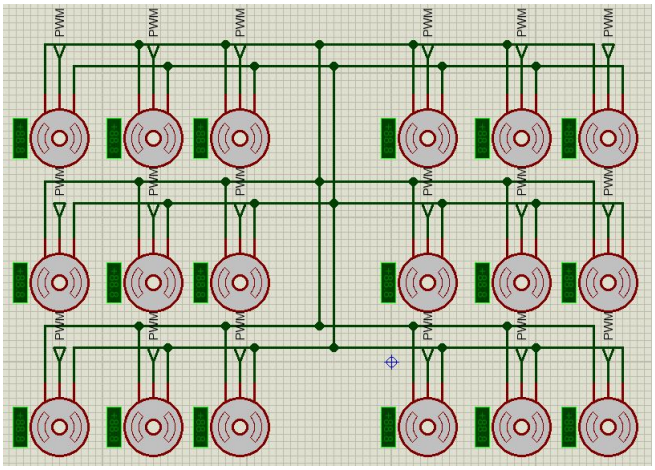


Figura 3.17: Diagrama de conexiones Servomotores.
Fuente: Elaborado por el Autor.

3.2.6. Cálculos de Consumo de Corriente

Para la etapa de potencia se utilizan reguladores de voltaje para alimentar a los servomotores y se utiliza un regulador de voltaje para poder alimentar al controlador, específicamente al PIC, de esta manera se puede separar la parte de control y la etapa de potencia, ya que el consumo de corriente de los 20 servomotores es alto y se puede mantener el voltaje para que los servos no pierdan fuerza. También porque elimina el ruido del controlador de los servos.

$$Momento = 3,02Kg * 10\frac{N}{Kg} * \frac{1m}{100cm} = 0,302N.m. \tag{3.1}$$

$$VelocidadAngular = \frac{60°}{0,19seg} * \frac{\pi}{180°} = 5,511\frac{rad}{seg} \tag{3.2}$$

$$Potencia = Momento * Velocidadangular \tag{3.3}$$

$$Potencia = 0,302N.m. * 5,511\frac{rad}{seg} = 1,664W$$

$$P = V * I \tag{3.4}$$

$$I = \frac{P}{V}$$

$$I = \frac{1,664}{5} = 0,332A$$

3.3. Diseño de Programación

Para la realización de la programación se siguen la siguiente lógica de programación.

3.3.1. Flujo de programación Controlador PWM

La mejor manera de poder controlar las señales, es realizarlas a través de un control general de condiciones, dentro de un bucle general. Dentro de este bucle se pueden tomar lectura a las entradas externas. Que son 4 pines de control como se muestra en la figura 3.10 A,B,C,D Se mantiene el PWM y se controla el movimiento que cada servomotor este realizando.

3.3.2. Programación Arduino

En la programación de la placa Arduino, se acopla el control y la comunicación de los sensores. Primeramente se estable los parámetros de comunicación en forma serial, como la comunicación que se realiza en general, todos los dispositivos utilizan 9600bd, entonces se configura a el modulo bluetooth y con la PC a este tipo de velocidad de comunicación. Se utiliza comunicación serial con la computadora para visualizar los datos que envían los sensores y el módulo bluetooth. Esto sirve para las pruebas de funcionamiento. La lógica de programación se muestra en 3.21.

3.3.3. Programación Android Studio(java)

Esta programación permite realizar una ventana de Aplicación flotante para poder compartir la pantalla del teléfono smartphone. Debido a que la aplicación de la cámara está hecha específicamente por el fabricante, no se puede compartir la pantalla ya que no cuenta con esta opción. Es por eso que la pantalla flotante es la mejor alternativa para compartir la pantalla.

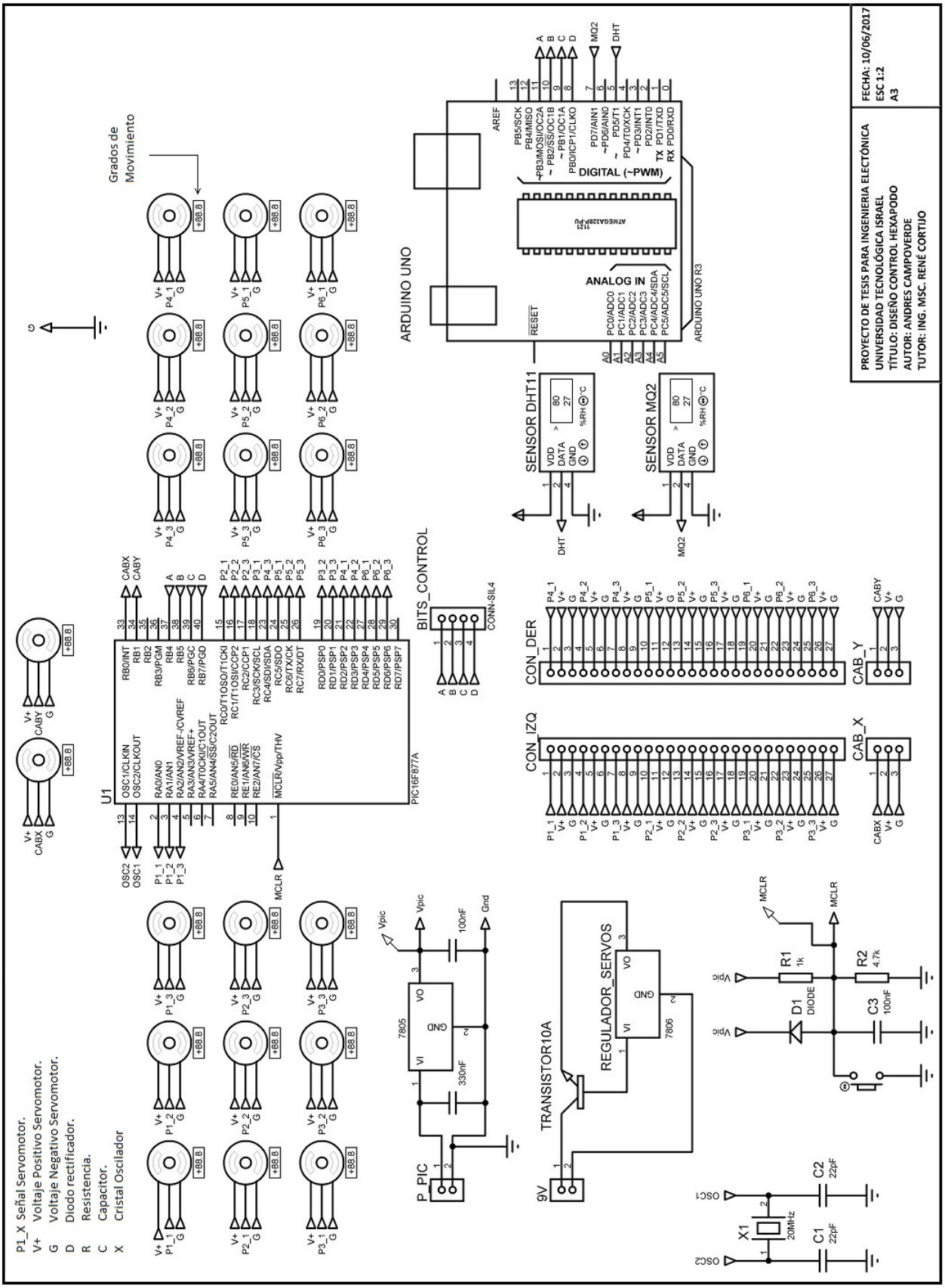


Figura 3.18: Diagrama de conexiones Controlador.
Fuente: Elaborado por el Autor.

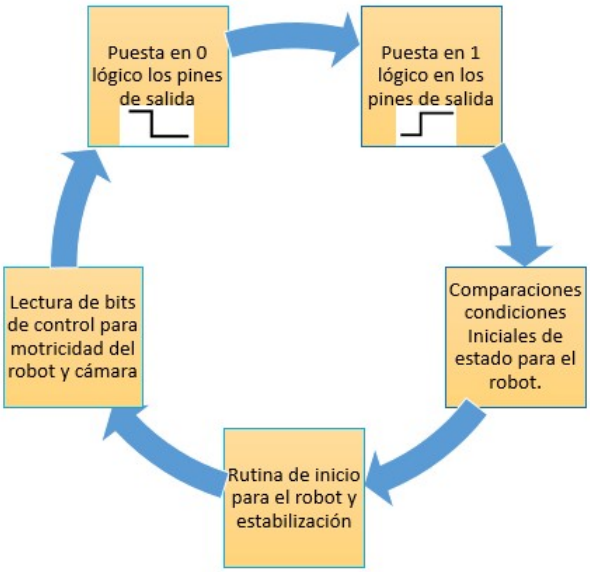


Figura 3.19: Lógica de programación PWM.
Fuente: Elaborado por el Autor.

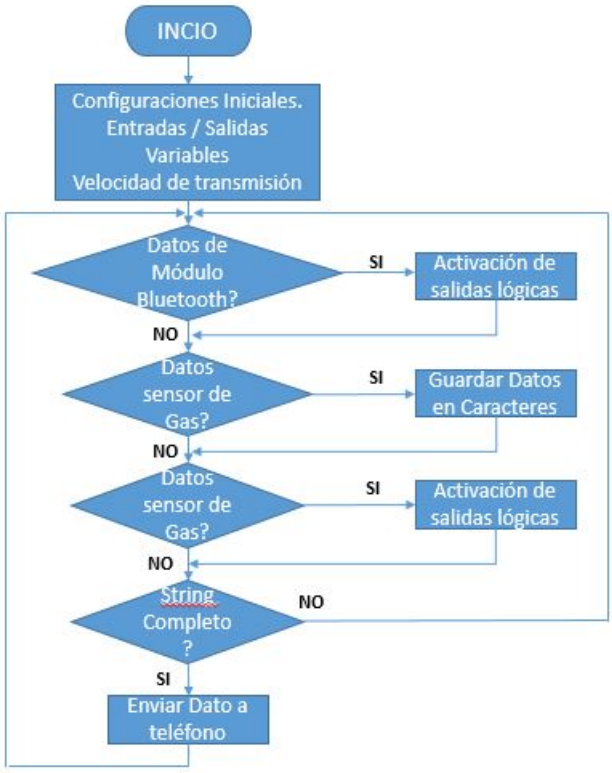


Figura 3.20: Lógica de programación Arduino.
Fuente: Elaborado por el Autor.

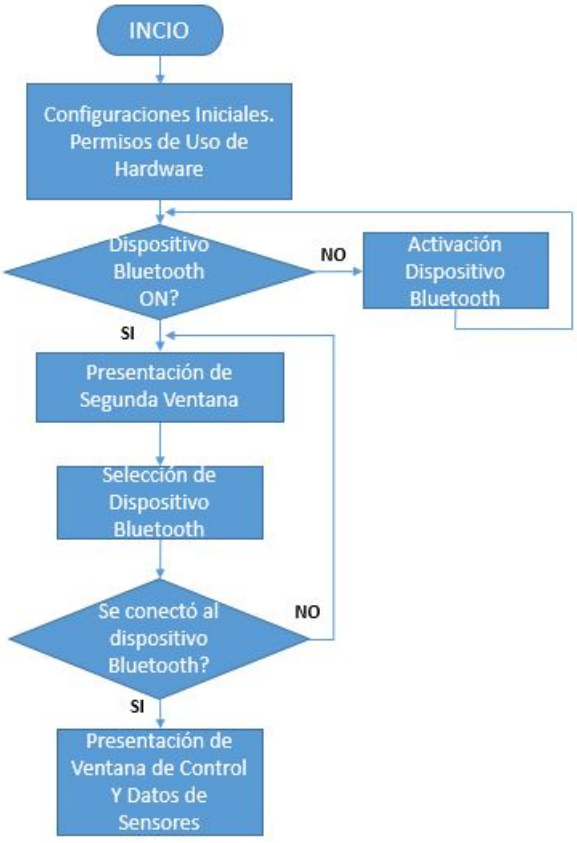


Figura 3.21: Lógica de programación Android.
Fuente: Elaborado por el Autor.

CAPÍTULO III IMPLEMENTACIÓN

5.1. Desarrollo

5.1.1. Creación PWM

En el software MPLAB se realiza mediante código la generación de las señales de control para los 20 servomotores. En este enotrno se puede compilar el programa que se esta ejecutando, como resultado se obtiene un documento de extensión .HEX el cual se graba directamente en el microcontrolador, a través del programa Pickit2.
La ventana en donde se digita el código dentro del proyecto es la ventana con extensión .asm. EL programa se puede compilar todo a la vez, o paso a paso. Esto es la venta-

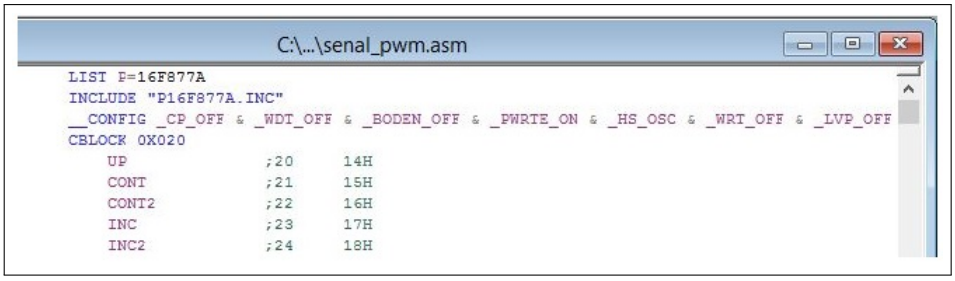


Figura 5.1: Ventana de escritura de código.
Fuente: Desarrollo Código Software MPLAB.

ja de programar en este software, que ayuda mucho a encontrar errores o para darse cuenta como sigue su curso cada línea de programación, esto brinda una seguridad al programador a la hora de realizar las comprobaciones.

Figura 5.2: Botones de ejecución del Programa.
Fuente: Desarrollo Código Software MPLAB.

Para poder observar los cambios que tienen las variables declaradas o las que tiene internamente el microcontrolador, se crea dentro de la ventana WATCH y se seleccionan las variables que se necesita. En este caso es muy importante observar los incrementos de las variables PATAx y los contadores que tiene, ya que todo se modifica en una unidad

a la vez, dentro del lazo global de programación. Uno de los más importantes es el registro WREG, es el que realiza las operaciones internas en cada línea de programación. Como cuando se realizan comparaciones u operaciones matemáticas.

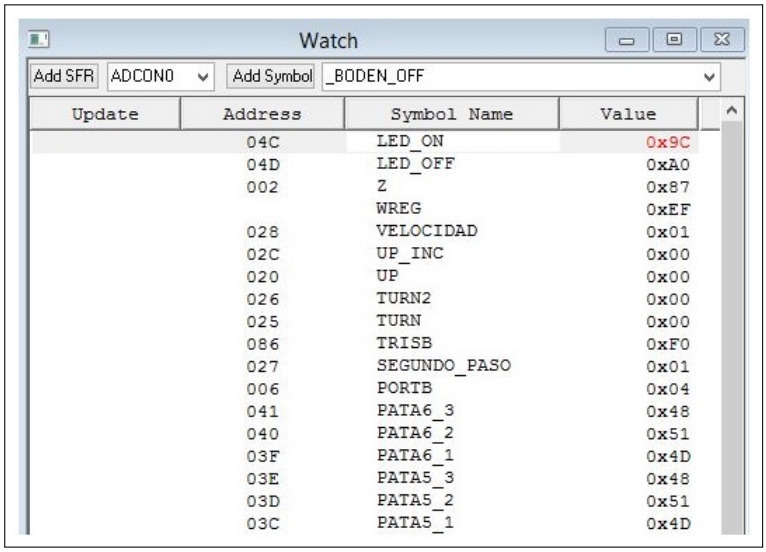


Figura 5.3: Visualización de Variables.
Fuente: Desarrollo Código Software MPLAB.

En la programación de la tarjeta controladora del robot, se debe tomar en cuenta que se tiene ingreso de datos externos. Otra de las cualidades del software MPLAB es que permite simular las entradas externas, dentro de la venta STIMULUS. Dentro de esta ventana se puede simular el ingreso de datos a nivel de BIT, en los periféricos del microcontrolador. Para este caso en el que, el control se hace a través de 4 pines RB4, RB5, RB6 y RB7.

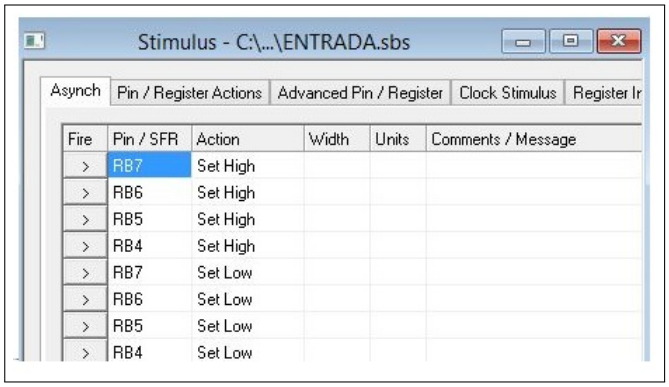


Figura 5.4: Ingreso de Datos Stimulus.
Fuente: Desarrollo Código Software MPLAB.

5.1.2. Control y Comunicación

Una vez que se tiene listo el código de programación para el control de los servomotores, la cabeza del sistema es el Arduino. En la tarjeta Arduino se controlan los bits de control para realizar los diferentes movimientos que tiene el robot.

Se acoplan los sensores, a los pines 5 para el DHT11 y 7 MQ2. Para realizar una buena comunicación se debe utilizar las librearías de los fabricantes. Esto se lo hace buscando en el internet con el Modelo y no tiene costo ya que es gratuito.
Una vez que se tiene las librerías se las coloca en la carpeta C:\Program Files (x86) \Arduino\libraries.

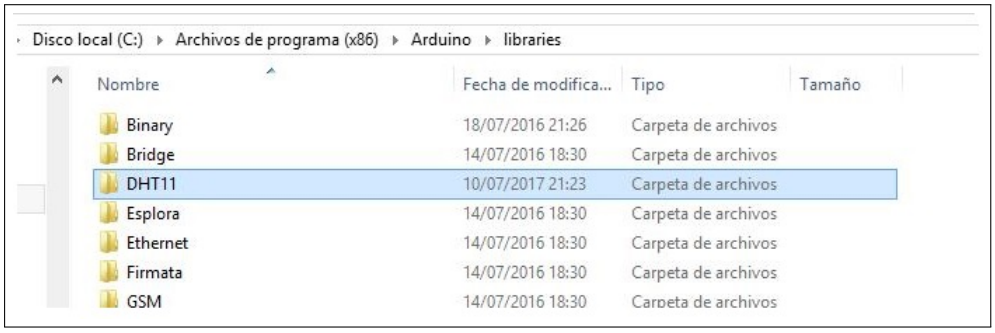


Figura 5.5: Librerías para los Sensores y Bluetooth.
Fuente: Desarrollo Código Software Arduino.

En el software Arduino se declara las variables como archivos de cabecera. Esto se hace para poder utilizar los comandos o sintaxis de las librerías incluidas, para poder simplificar código de programación. Con estas librerías se puede hacer trabajar directamente al sensor, sin necesidad de calibraciones como se lo haría manualmente. Para el proyecto se incluyen dos librerías como se muestra en la figura.5.6

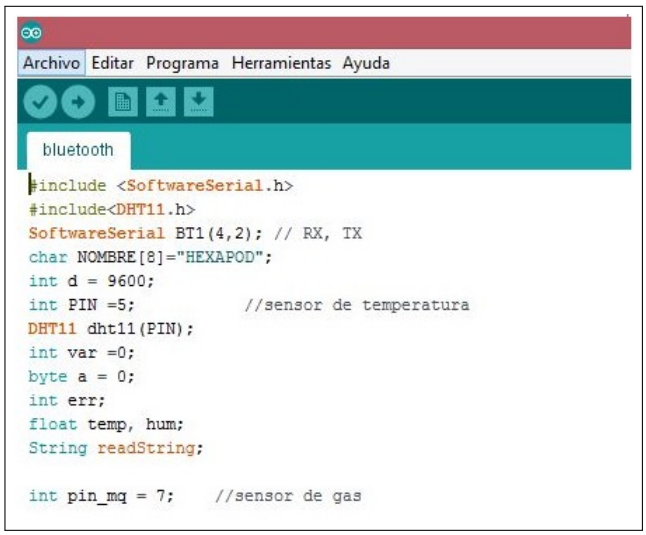


Figura 5.6: Llamado de las librerías Arduino.
Fuente: Desarrollo Código Software Arduino.

5.2. Implementación

Para la implementación total del sistema, se unen todas las secciones de diseño. Mecánica, Electrónica y Software.

5.2.1. Armado del Hexápodo

Extremidad

En la parte del brazo existe una abertura cuadrada, con dimensiones del servomotor para que poder incrustarlo dentro del brazo. Para fijar las dos partes, se realizan unas perforaciones del tamaño del diámetro del tornillo que se va a utilizar. No necesariamente tiene que ser los del kit del servomotor.

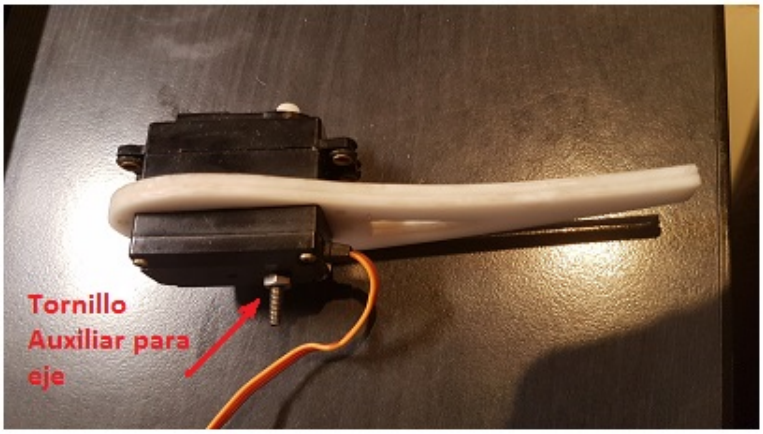


Figura 5.7: Armado del Brazo del Hexápodo.
Fuente: Elaborado por el Autor.

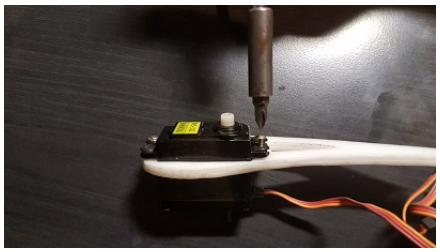


Figura 5.8: Armado del Brazo del Hexápodo.
Fuente: Elaborado por el Autor.

Armado del hombro

Para poder unir los dos servomotores que van el eje del tórax del robot, se utilizó una platina delgada de metal, y se la dobló con las dimensiones de los dos servomotores. En sentido horizontal y vertical para que con el movimiento del robot no se muevan o se aflojen.

Se realizó el montaje de cada una de las extremidades por separado. Como en el diseño que se propuso inicialmente. Colocando los acrílicos en los puntos de eje de cada servomotor.

Figura 5.9: Armado del Hombro del Hexápodo.
Fuente: Elaborado por el Autor.

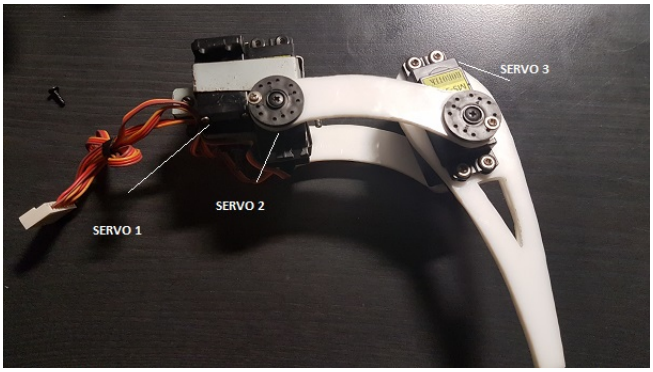


Figura 5.10: Armado de la extremidad del Hexápodo.
Fuente: Elaborado por el Autor.

Una vez realizado el armado de todas la extremidades se procedió a montar todas las partes al tórax del hexápodo. Esto se lo hizo con los accesorios que vino los servomotores, como kit de instalación. Aquí se ve puede observar como queda el montaje del hexápodo.

Figura 5.11: Tipos de tornillos utilizados.
Fuente: Elaborado por el Autor.

5.2.2. Microcontrolador

Se utilizan los pines de I/O entrada y salida, no se utilizan los pines que son activados, en bajo. Y se distribuye de manera que se encuentren de un lado los servos de las extremidades izquierdas y del otro lado las derechas. Como se muestra en la Tabla 5.1.Para mantener las 20 señales PWM y realizar al mismo tiempo las operaciones matemáticas se utiliza un oscilador de 20 MHz. Con eso se puede estimar que el tiempo de

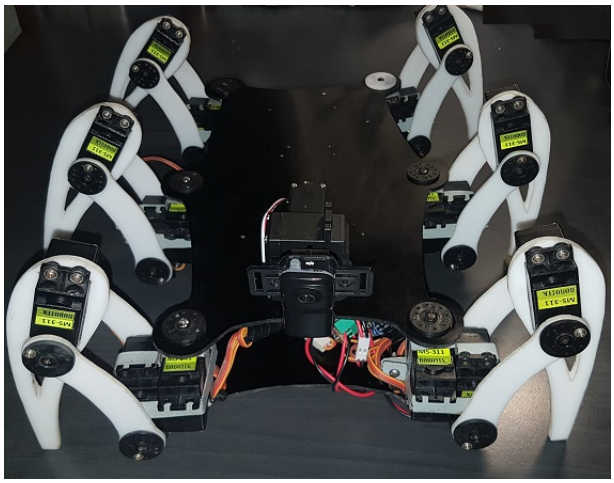


Figura 5.12: Armado del Hexápodo.
Fuente: Elaborado por el Autor.

ejecución de cada línea de comando sea 50 nanosegundos.

$$T = \frac{1,20000000}{=} 0,00000005s$$

5.2.3. PCB

El desarrollo del PCB se lo realizó para la tarjeta controladora de los servomotores. Se realizo el diseño en el software Proteus 8.0. PCB Layout. Verificando las conexiones y el diseño de las pistas de cobre.Cabe señalar que el diseño esta realizado en 2 caras, para evitar muchas vueltas y el sobremontaje de las mismas. Se modificó también los puntos de conexiones, debido a que el diseño que tiene el software es pequeño. De esta manera se mejora las soldadura y se evita que se dañen.

Las líneas verdes que se muestran en la simulación, son conexiones que no se puede realizar en el PCB. Estas líneas se pueden conectar mediante cables tipo puente.5.14 El proceso para obtener la placa es con papel auto-transferible, y se diluye con la solución química de Cloruro Férrico y agua.5.14

5.2.4. Acoplamiento Cámara

En la parte superior del hexápodo se acoplan dos servomotores, para que la cámara tenga 2 grados de libertad. La manera de acoplar los servomotores es que el eje un servo este conectado al tórax del robot y el otro servomotor debe estar pegado a este, para que pueda girar conforme el primer motor. El segundo motor debe acoplar a la base de la cámara, para realizar el otro giro en sentido perpendicular al primero.
La base de la cámara se debe acoplar al eje del segundo servomotor mediante un an-

Tabla 5.1: Pines utilizados salida PWM y Control

Pin	Puerto	Descripción
2	RA0	PATA1_1
3	RA1	PATA1_2
4	RA2	PATA1_3
15	RC0	PATA2_1
16	RC1	PATA2_2
17	RC2	PATA2_3
18	RC3	PATA3_1
19	RD0	PATA3_2
20	RD1	PATA3_3
21	RD2	PATA4_1
22	RD3	PATA4_2
23	RC4	PATA4_3
24	RC5	PATA5_1
25	RC6	PATA5_2
26	RC7	PATA5_3
27	RD4	PATA6_1
28	RD5	PATA6_2
29	RD6	PATA6_3
27	RB0	PATA6_1
33	RB1	CAB_X
34	RD6	CAB_Y
37	RB4	A_Adelante
38	RB5	B_Atras
39	RB6	C_Giro Izquierda
40	RD7	D_Giro Derecho

Fuente: Elaborado por el Autor.

gulo, creado con el material del robot. Para que pueda tener el giro libre entre el eje sel servo y la base de la cámara.

5.2.5. Conexiónes a la tarjeta Arduino

Para realizar las conexiones de los sensores y el módulo Bluetooth a la placa arduino, se instaló conectores molex para asegurar las conexiones, también por motivo de que se pueda desmontar las partes en caso de necesitarlo, o para realizar cambios a futuro.5.17

El control que se realiza en la placa Arduino Uno, sirve para la comunicación del modulo Bluetooth, conexiones de sensores y control de movimientos del robot. Los pines de distribución para el control de movimientos se lo realiza de la siguiente manera:

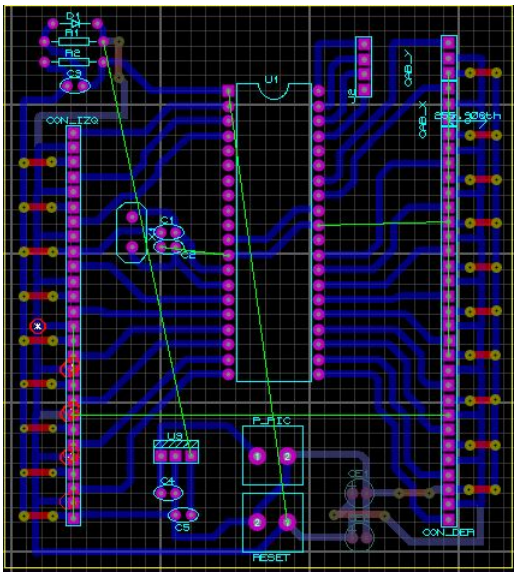


Figura 5.13: Diseño PCB .
Fuente: Elaborado por el Autor.

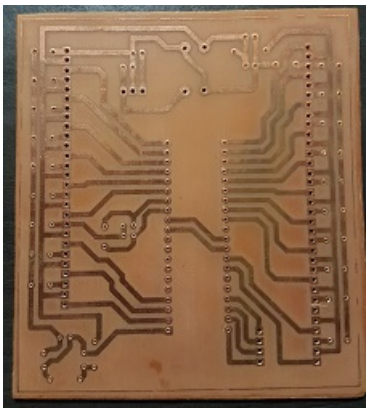


Figura 5.14: PCB tarjeta controladora.
Fuente: Elaborado por el Autor.

5.2.6. Comunicación del Teléfono con el Robot Hexápodo

Con la aplicación realizada se deben seguir estos pasos para tener el control del robot.

- Abrir la aplicación que debe estar instalada en el dispositivo Android
- Una vez presionado el botón de iniciar comunicación, aparecerá una segunda pantalla para mostrar los dispositivos con los que se ha apareado, el dispositivo móvil.
- Pantalla flotante donde se puede observar los controles de la aplicación. ”A” Control de movimiento del Robot hacia adelante.
  ”B” Control de movimiento del Robot hacia atrás.
  ”I” Control de movimiento del Robot hacia izquierda.

Tabla 5.2: Control Arduino - Robot

Pines Arduino				**Descripción**		
11	10	9	8			
0	0	0	1	Adelante	1	Control Robot
0	0	1	0	Atrás	2	Control Robot
0	0	1	1	Giro Izq	3	Control Cámara
0	1	0	0	Derecha	4	Control Robot
0	1	0	1	Giro Der	5	Control Cámara
0	1	0	1	Arriba	6	Control Cámara
0	1	1	1	Abajo	7	Control Cámara
1	0	0	0	Izquierda	4	Control Robot

Fuente: Elaborado por el Autor.

”D” Control de movimiento del Robot hacia derecha.
Las flechas realizan el giro de la cámara en los 2 ejes.

- Conexión con la Cámara. Para realizar la conexión de la cámara, primeramente se debe encenderla en modo P2P y realizar la conexión desde el dispositivos android, como si se fuera a conectar hacia un dispositivo WIFI.
- Finalmente se obtiene la interface gráfica conjuntamente con los controles del robot hexápodo.

5.3. Costos

El proyecto tiene un costo alto por la cantidad de servomotores que se necesitan, por el tipo de cámara tipo espía y también por las baterías de litio. El precio puede variar dependiendo de las tiendas electrónicas en donde se puede realizar la compra de materiales necesarios. Debido a lo complejidad del proyecto, se debe dedicar bastante tiempo de trabajo para el desarrollo y la implementación. Por eso se contempla como 300 horas, aproximadamente de trabajo para realizar el proyecto, también porque se necesita desarrollar la programación en algunos lenguajes o compiladores. LOs lenguajes con los que se trabajó son: Programación en C (para Arduino), Java (aplicación de teléfono), Assembler (control microcontrolador).

5.4. Pruebas de Funcionamiento

5.4.1. Encendido del Robot

Se enciende el robot a través del switch de la batería. Luego realiza las siguiente rutina.

- Colocación inicial de cada extremidad para que el pico de corriente sea bajo.

Tabla 5.3: Presupuesto Proyecto

Item	Cantidad	Descripción	$Costo U	$ Valor
1	20	Servomotor Hitec HS-311	15	300
2	1	Acrílico m2	28	28
3	1	Tarjeta Arduino Uno	16	16
4	1	Cámara Wifi P2P MD81	75	75
5	1	Módulo Bluetooth HC-05	10	10
6	1	Módulo Sensor HC-11	9	9
7	1	Módulo Sensor MQ2	8	8
8	1	Microcontrolador PIC16F877A	7	7
9	1	Tarjeta Controladora	35	35
10	1	Módulo Regulador voltaje 7805	8	8
11	1	Módulo Regulador voltaje 7806	12	12
12	2	Batería Litio 12V 4.5A	60	120
13	10	Conectores Molex	0.8	8
14	20	Cable AWG #24	1	20
15	1	Smartphone Android	120	120
16	300	Horas de Trabajo	5	1500
			Total$	2276

Fuente: Elaborado por el Autor.

- Elevación del robot con las extremidades, desde el suelo.
- Se acomoda cada extremidad para que se nivele la carga en cada punto de apoyo.

En este punto se encuentra listo el robot para realizar las movimientos.

5.4.2. Datos de los sensores

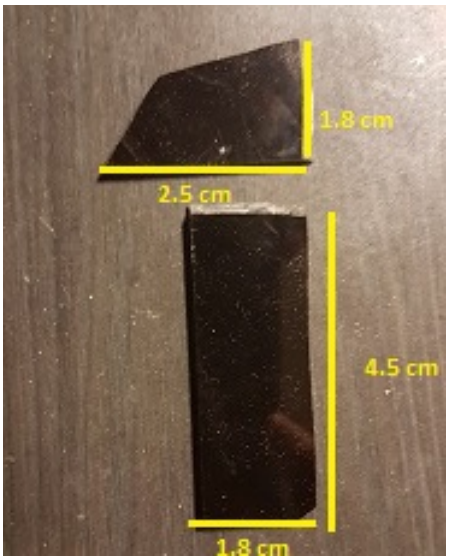
Los datos que se muestran en la interfaz gráfica del Teléfono.5.23

5.4.3. Conectividad con la Cámara

La cámara envía imágenes hasta de 640x480 dependiendo la conectividad. Con la conexión realizada con el teléfono se obtienen las siguientes imágenes, con una ligera variación en la velocidad conexión.5.24

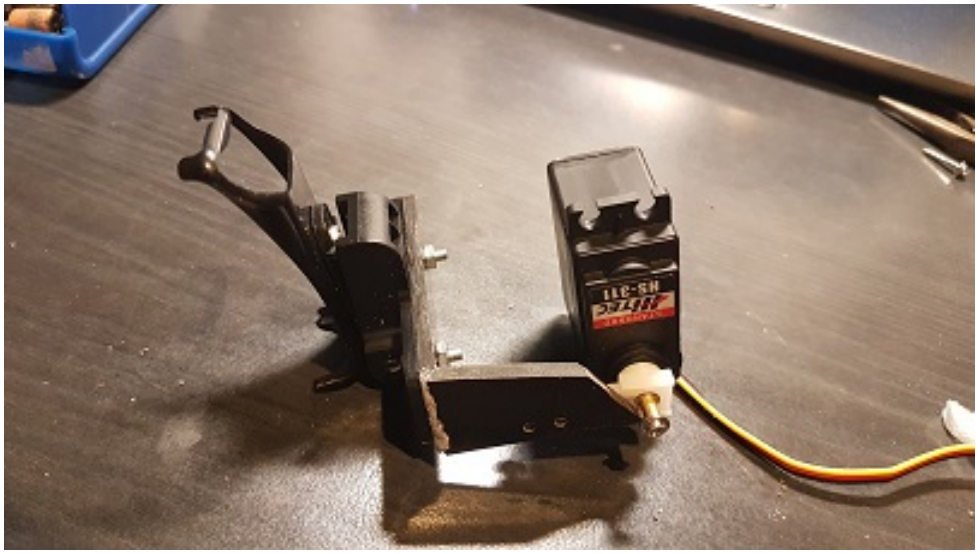
5.5. Análisis de Resultados

Al realizar las pruebas de comunicación con la aplicación Android, y el desempeño del robot se tiene las siguientes aspectos que debe cumplir el robot.



(a) Corte Ángulo

(b) Base Cámara



(c) Acoplamiento

Figura 5.15: Montaje para la Cámara
Fuente: Elaborado por el Autor.

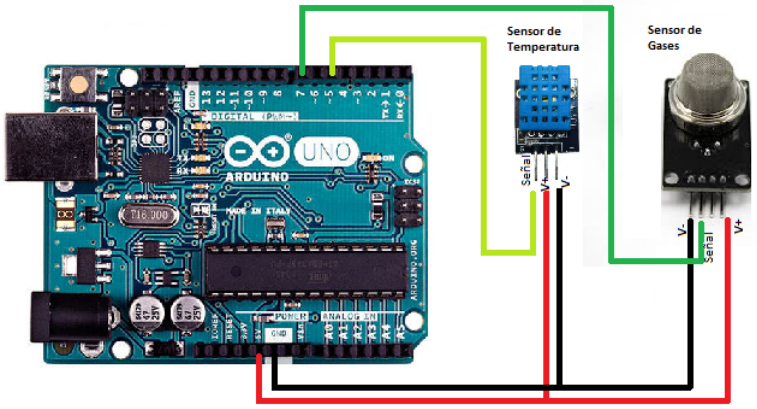


Figura 5.16: Diagrama Conexión Arduino Sensores.
Fuente: Elaborado por el Autor.

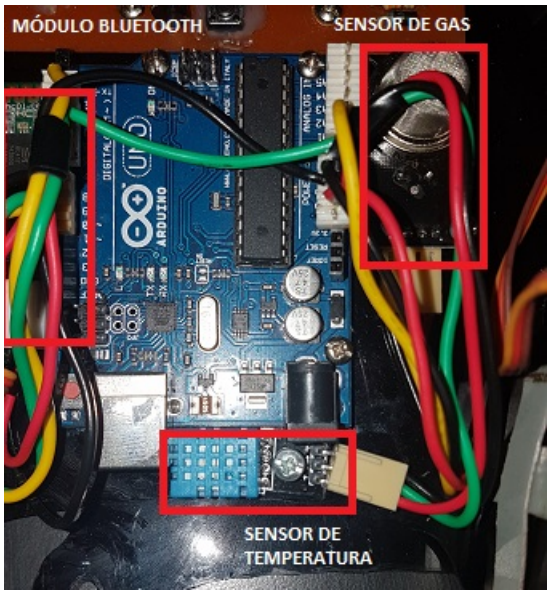


Figura 5.17: Conexión Placa Arduino Sensores.
Fuente: Elaborado por el Autor.

Figura 5.18: Presentación de la aplicación.
Fuente: Elaborado por el Autor.

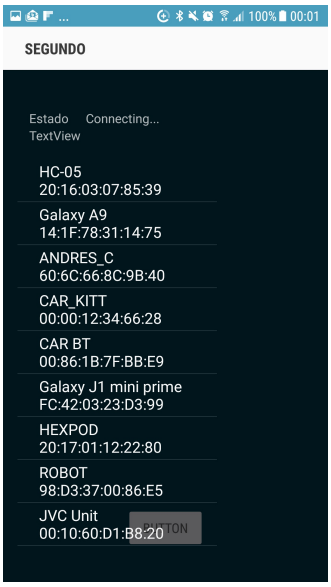


Figura 5.19: Selección del dispositivo HEXPOD.
Fuente: Elaborado por el Autor.

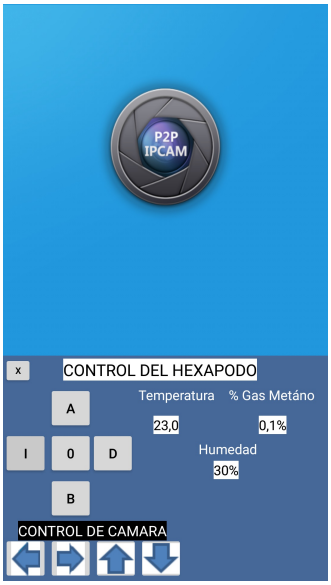


Figura 5.20: Ventana de control del dispositivo HEXPOD.
Fuente: Elaborado por el Autor.

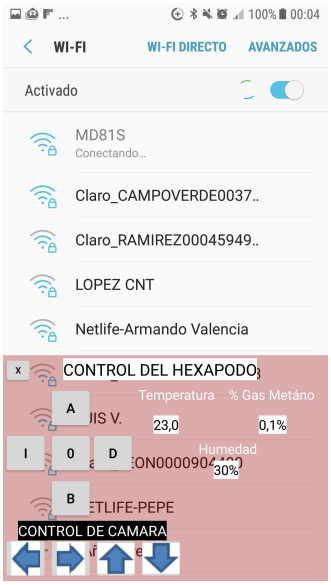


Figura 5.21: Conexión Cámara del dispositivo HEXPOD.
Fuente: Elaborado por el Autor.

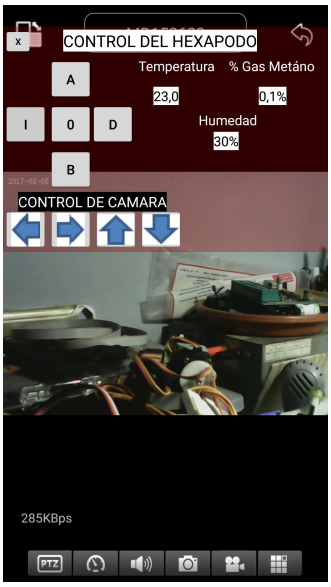
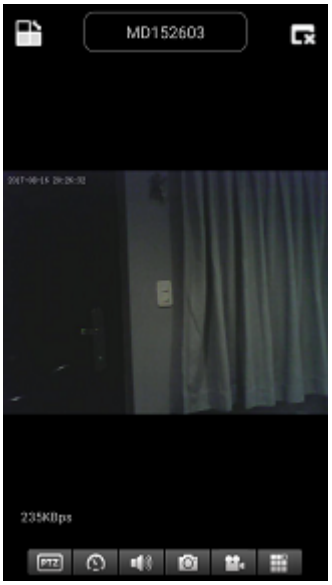


Figura 5.22: Interface Gráfica de Control HEXPOD.
Fuente: Elaborado por el Autor.

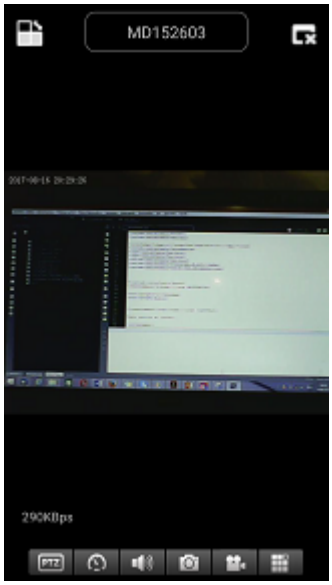
Figura 5.23: Valor de los sensores HEXPOD.
Fuente: Elaborado por el Autor.

(a) Cam1

(b) Cam2



(c) Cam3



(d) Cam4

Figura 5.24: Captura de Imágenes Cámara.
Fuente: Elaborado por el Autor.

Tabla 5.4: Análisis de Resultados

Item	Proceso	Cumple	No Cumple	Observación
1	Encendido de Robot	X		
2	Posición inicial de las Extremidades	X		
3	Elevación del Robot desde el suelo	X		
4	Nivelación de Carga	X		
5	Encendido de Tarjeta Arduino	X		
6	Encendido de Sensor de Temperatura	X		
7	Encendido de Sensor de Gases	X		
8	Lanzamiento de Aplicación Android	X		
9	Detección de Modulo Bluetooth ”HEXPOD”	X		
10	Conexión con el Módulo Bluetooth			
11	Visualización del Control Remoto	X		
12	Detección de Cámara MD81S	X		La bateria de la cámara debe estar más del 30 % para una buena señal
13	Conexión con la Cámara MD81S	X		
14	Visualización de Video	X		La velocidad de conexión a veces varía, se debe esperar unos segundos para volver a visualizar video
15	Grabación de Fotos y Video	X		Se graban en la sd card
16	Acoplamiento de interfaces Wifi y Bluetooth	X		
17	Giro de ejes de la Cámara	X		
18	Recepción de Datos de los Sensores		X	A veces la sincronización de Datos se pierde y se puede presentar caracteres extraños en pantalla

Fuente: Elaborado por el Autor.

CONCLUSIONES

- Al construir la parte mecánica, se verifica que necesariamente, se debe hacer los diseños con las medidas exactas, para poder acoplar los elementos electrónicos, en los motores y acoples. El uso del software Autocad fue eficiente para establecer las formas y tamaños del tórax y las extremidades.
- La parte mecánica es importante para crear los movimientos del hexápodo, depende mucho de la posición que tengan las extremidades con respecto al robot, porque las extremidades no realizan el mismo movimiento al momento de desplazar, un posición diferente hace cambiar radicalmente la programación.
- La parte fundamental del proyecto hexápodo es la cantidad de señales PWM a controlar, esto hace desarrollar la perspectiva de programación a través de un gran bucle. Y se puede dar cuenta de la velocidad de procesamiento que tiene el microcontrolador.
- Al desarrollar el programa del microcontrolador en el software MPLAB, se puede ahorrar tiempo para realizar las revisiones de sintaxis y procedimiento en cada línea de código. Sin tener que buscar un simulador de microcontroladores, gracias a las aplicaciones internas del software.
- Durante el desarrollo del código Assembler se debe tener puntos de referencias, ya que el código es extenso y puede llevar tiempo ubicarse en las líneas de programación que se desea trabajar o revisar.
- Al implementar la aplicación para la comunicación bluetooth, se observó que los datos a veces no llegan de manera exacta, esto sucede por que la sincronización se pierde por momentos y puede mostrar en pantalla dígitos extraños, porque durante la impresión del dato existen espacios vacíos cuando no se sincroniza.
- Cuando se realiza la comunicación serial con el módulo bluetooth y la placa Arduino, se comprobó que no existen puertos dedicados a esta transmisión, sino que el programador puede hacer uso de otros puertos diferentes, para comodidad del diagrama de conexiones.
- Durante la implementación de la interfaz gráfica en el teléfono, el programa que permitió crear la ventana flotante de control para que este sobrepuesta, a las demás aplicaciones es Android Studio, debido a su lenguaje de programación más

básico, también porque existe mucha información en Internet, en páginas de desarrolladores y foros.

- El desempeño del robot varía según el tipo de terreno que en el que esta apoyado, las extremidades pero el robot puede desplazarse con mejor facilidad que las ruedas. Como se pudo observar en las pruebas de movimientos.
- El consumo de energía es elevado frente a otro tipo de robots, que utilizan rueda. pero las prestaciones que tiene este robot es compensable con el tipo de funciones que podría realizar. Como pasar por obstáculos o terreno rocoso.
- Para el prototipo realizado el costo como tal, es un poco alto, pero al realizar la construcción en masa de este tipo de robots puede bajar el precio significativamente.
- En el momento de las pruebas que se hizo con los servomotores, se pudo constatar que es mejor la señal que produce el microcontrolador a través de la programación que utilizando un timer externo, ya que producen más ruido.

RECOMENDACIONES

- Hoy en día existen microcontroladores que podrían realizar la comunicación bluetooth y wifi respectivamente, porque tienen un nivel de procesamiento más elevado. De esta forma se podría acoplar todas las etapas de diseño electrónico a este prototipo; pero, no sería conveniente centralizar todo en un solo procesador o dispositivo electrónico. Por motivo de mantenimiento y revisión.
- Se recomienda utilizar baterías de litio ya que son las que mejor se acoplan al diseño mecánico y su duración es mayor frente a otras baterías.
- Si el robot tendría que ser de tamaño mas grande, se recomienda implementar con servomotores de más torque, la velocidad disminuye pero se compensa en fuerza, sin embargo también el consumo de voltaje se elevaría de 1 a 2 voltios pero con la misma capacidad de corriente.
- Se debe tener conocimiento de 3 lenguajes de programación que demanda el proyecto, esto implica que se utilizarán bastante tiempo de desarrollo para la culminación.
- Al realizar la programación en diferente software, es conveniente realizar una estructura del programa con comentarios, porque cuando el código es extenso se hace complicado la revisión del mismo.
- El uso de baterías de plomo es más barato, pero si se utiliza de plomo el peso es excesivo para la movilidad del robot y no se tendría espacio para montar una batería de este tamaño.
- Para la implementación del robot es conveniente realizar las pruebas de motricidad de las extremidades por separado, para determinar su movimiento requerido para el robot y para las correcciones del caso.
- Para garantizar el funcionamiento de hasta 1 hora, se debe realizar un carga de baterías por completo por el tiempo de 2 horas aproximadamente, al igual que el de la cámara WIFI.
- Se recomienda realizar las compras de los elementos electrónicos en tiendas nacionales, ya que por Internet, si bien se puede ahorrar en precios, pero no se tiene garantía de funcionamiento y tampoco se puede realizar una devolución a corto plazo.

Bibliografía

abc tecnoloía Madrid. *¿Qué es el WiFi Direct y para qué se usa?*, 2015. URL http://www.abc.es/tecnologia/consultorio/20150212/abci-wifi-direct-como-usar-201502111739.html.

Iván Almeida Hernández and Jimmy Ochoa Urgilés. Diseño y construcción de un robot explorador de terreno. B.S. thesis, 2013.

Román Gregorio Hernández Alvarez. *Microcontroladores*, 2016. URL https://www.tecnologiarobotica.com/post/microcontroladores/.

UNO Arduino, Arduino Ethernet, and Arduino Android. Shields. 2014.

ABB Inc. Robotic Industries Association. *IRB 140*, 2017. URL http://new.abb.com/products/robotics/es/robots-industriales/irb-140.

2014 Depto. CCIA. Desarrollo de aplicaciones para android. 2014. URL http://www.jtech.ua.es/cursos/apuntes/moviles/daa2013/wholesite.pdf.

Hewlett-Packard Development Company. *Bluetooth wireless technology basics*, 2004. URL http://h10032.www1.hp.com/ctg/Manual/c00186949.pdf.

Kiran. Daware. *How does a servo motor work?*, 2015. URL http://www.electricaleasy.com/2015/01/how-does-servo-motor-work.html.

Rob Ermanno. *Introducción a las redes WiFi*, 2010. URL http://www.eslared.org.ve/walcs/walc2012/material/track1/05-Introduccion_a_las_redes_WiFi-es-v2.3-notes.pdf.

Brian Evans. *Beginning Arduino Programming.* Apress, 2011.

Hitec. *HS-311 Standard Economy Servo*, 2017. URL http://hitecrcd.com/products/servos/sport-servos/analog-sport-servos/hs-311-standard-economy-servo/product.

I. Angulo Martínez J. M. Angulo Usategui. *Microcontroladores PIC. Diseño práctico de aplicaciones.* 1999.

Naylamp Mechatronics. Tutorial sensores de gas mq2, mq3, mq7 y mq135. 2016. URL http://www.naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html.

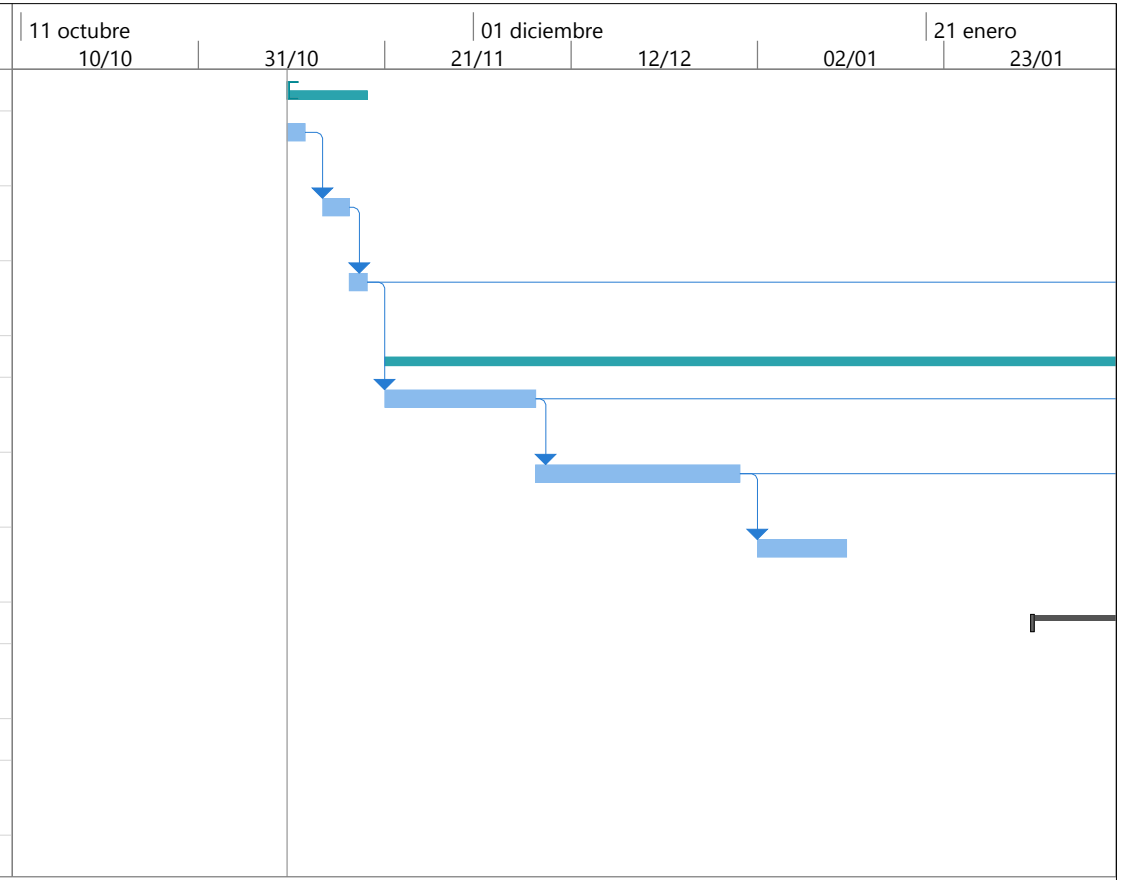
Profesor Molina. *Historia de la Robótica*, 2015. URL http://www.profesormolina.com.ar/tecnologia/robotica/historia.htm.

Ignacio Pedrosa Lojo. Proyecto mirho (mobile intelligent hexapod robot). 2008.

ANEXOS

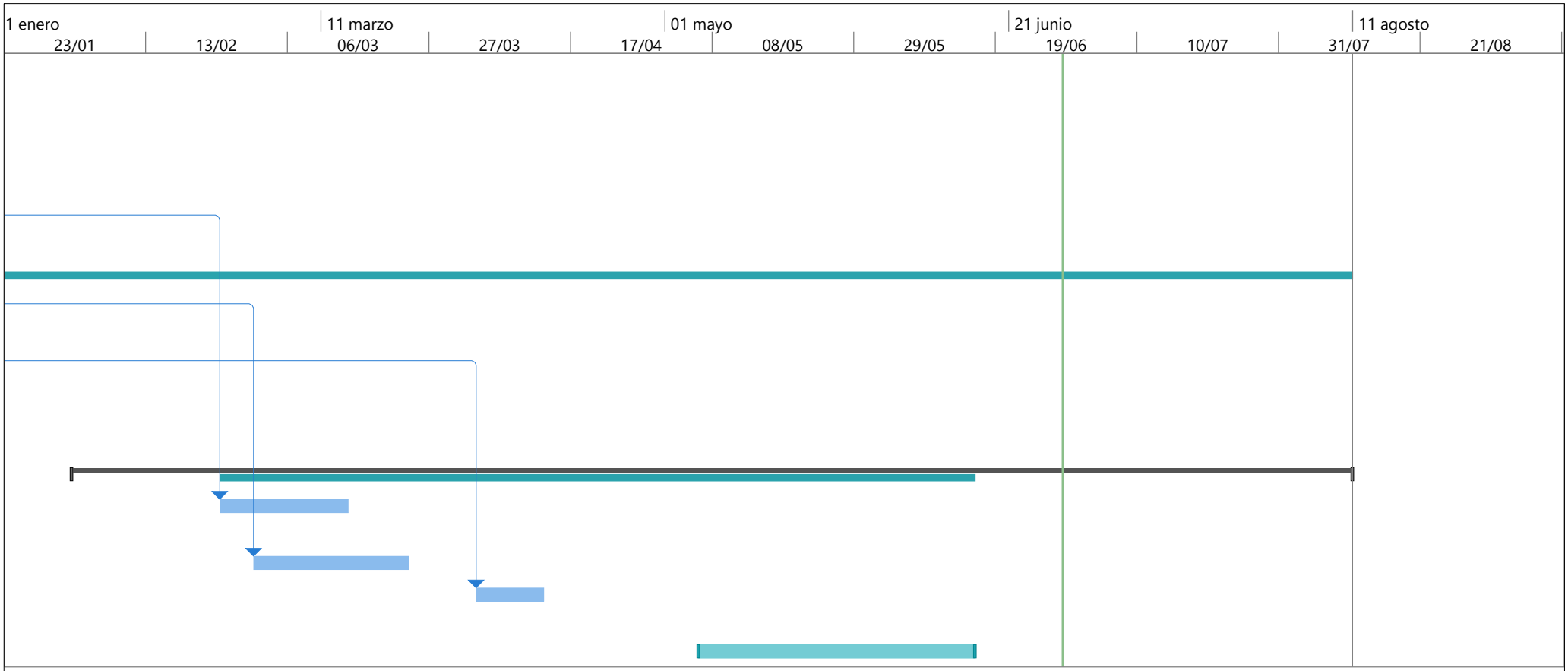
- Cronograma de Actividades
- Plano Electrónico
- Código Assembler
- Código Arduino
- Código Android Studio
- Datasheet PIC16f877A

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**Diseño del Robot**		**jue 10/11/16**
2		Diseño Prototipo	2 días	jue 10/11/16
3		Diseño Electrónico	3 días	lun 14/11/16
4		Diseño de Programación	2 días	jue 17/11/16
5		**Diseño Mecánico**		
6		Diseño Tórax	13 días	lun 21/11/16
7		Diseño de Articulaciones	17 días	jue 08/12/16
8		Construcción	8 días	lun 02/01/17
9		**Programación Software**	136 días	**jue 02/02/17**
10		Programación Microcontrolador	13 días	**vie 24/02/17**
11		Programación Arduino	17 días	**mié 01/03/17**
12		Elaboración Aaplicación Android	8 días	**lun 03/04/17**
13		**Documento Escrito**	30 días	sáb 06/05/17



Proyecto: Proyecto Robot Hexá
Fecha: vie 25/08/17

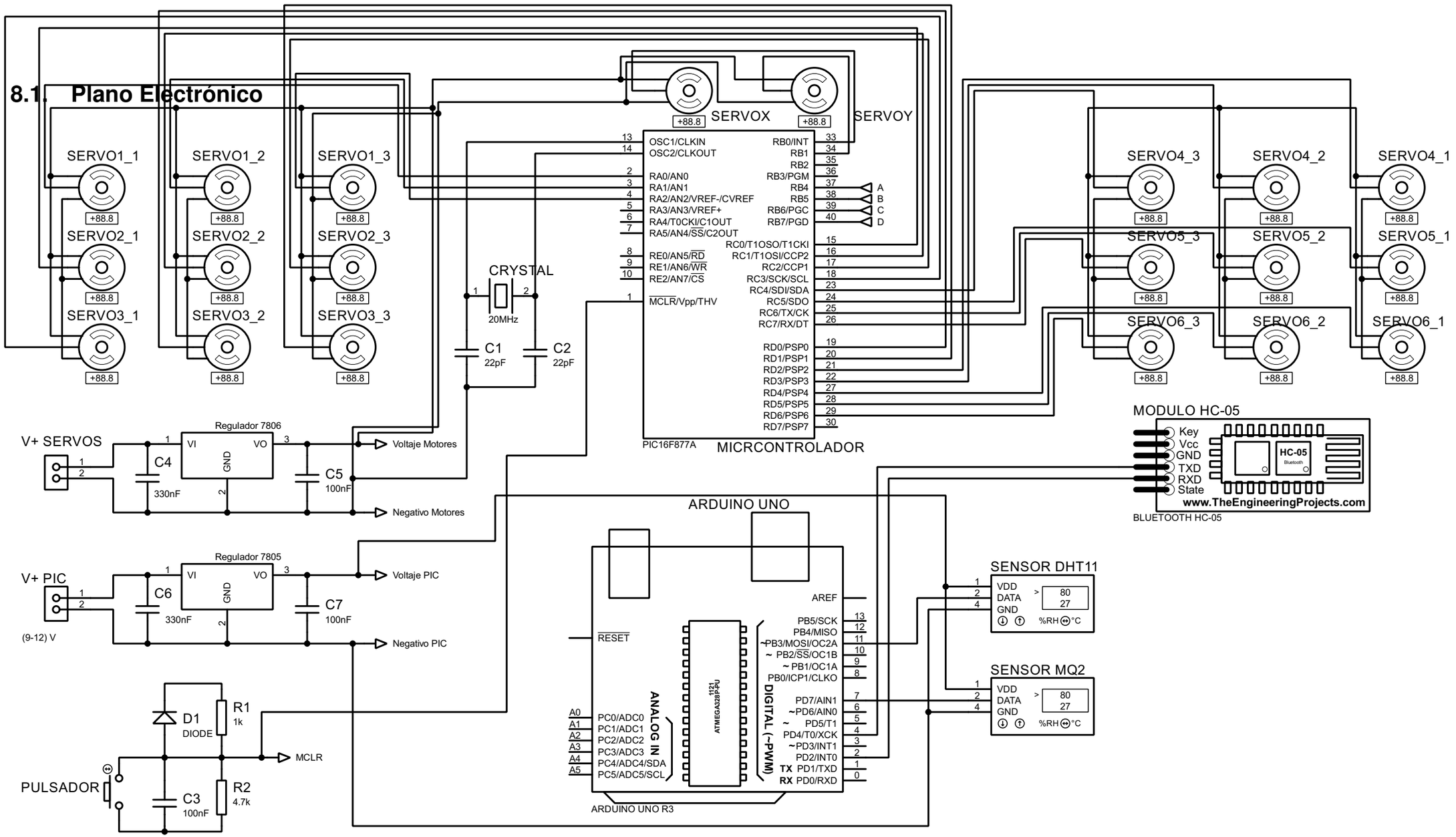
Tarea		Resumen inactivo		Tareas externas	
División		Tarea manual		Hito externo	
Hito		solo duración		Fecha límite	
Resumen		Informe de resumen manual		Progreso	
Resumen del proyecto		Resumen manual		Progreso manual	
Tarea inactiva		solo el comienzo			
Hito inactivo		solo fin			



Proyecto: Proyecto Robot Hexá
Fecha: vie 25/08/17

Tarea	Resumen inactivo	Tareas externas
División	Tarea manual	Hito externo
Hito	solo duración	Fecha límite
Resumen	Informe de resumen manual	Progreso
Resumen del proyecto	Resumen manual	Progreso manual
Tarea inactiva	solo el comienzo	
Hito inactivo	solo fin	

8.1. Plano Electrónico



V+ PIC Voltaje para alimentacion de Microcontrolador
R Resistencia
C Capacitor
D Diodo
RHT11 Sensor de Temperatura y Humedad
MQ2 Sensor de Gases
R3 Placa Arduino Uno
PIC Microcontrolador

FILE NAME: **Proyecto Hexápodo**	DATE: **24/08/2017**
DIAGRAMA DE CONEXIONES ELECTRÓNICO	PAGE: **1**
PATH: Diseño Control Hexapod	
Autor: Andres Campoverde Rev. Mg Cortijo	

8.2. Código Assembler Controlador

```
      LIST P=16F877A
INCLUDE "P16F877A.INC"
_CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC & _WRT_OFF
& _LVP_OFF & _CPD_OFF
CBLOCK 0X020
UP ;20 14H
CONT ;21 15H
CONT2 ;22 16H
INC ;23 17H
INC2 ;24 18H
TURN ;25 19H
TURN2 ;26 1AH
SEGUNDO_PASO ;27 1BH
VELOCIDAD ;28 1CH
CONT2_1 ;29 1DH
CONT2_2 ;30 1EH
MEDIO ;31 1FH
UP_INC ;32 20H
ACOM ;33 21H
LEGS ;34 22H
FIXER ;35 23H
PATA1_1 ;36 24H PATA UNO
PATA1_2 ;37 25H
PATA1_3 ;38 26H
PATA2_1 ;39 27H PATA DOS
PATA2_2 ;40 28H
PATA2_3 ;41 29H
PATA3_1 ;42 2AH PATA TRES
PATA3_2 ;43 2BH
PATA3_3 ;44 2CH
PATA4_1 ;45 2DH PATA CUATRO
PATA4_2 ;46 2EH
PATA4_3 ;47 2FH
PATA5_1 ;48 30H PATA CINCO
PATA5_2 ;49 31H
PATA5_3 ;50 32H
PATA6_1 ;51 33H PATA SEIS
PATA6_2 ;52 35H
PATA6_3 ;53 36H
CABEZAX ;55 37H CABEZAX
CABEZAY ;56 38H CABEZAY
DONE ;54 39H
INI1 ;55 3AH
INI2 ;55 3BH
INI3 ;56 3CH
```

```
INI4 ;57 3DH
INI5 ;58 3EH
INI6 ;59 3FH
INIF ;60 40H
LED_ON ;61 41H
LED_OFF ;62 42H
ENDC

    GOTO CONF_INIC

    CONF_INIC:
BSF STATUS, RP0 ;BANCO 1
CLRF TRISA
CLRF TRISC
CLRF TRISD
MOVLW b'11110000';0XF0
MOVWF TRISB
MOVLW 0X06
MOVWF ADCON1
BCF STATUS, RP0 ;BANCO 0
CLRF PORTA ;NOTA NO UTILIZAR EL PUERTO RA4 (SE ACTIVA EN BAJO)
CLRF PORTB
CLRF PORTC
CLRF PORTD
MOVLW 01H
MOVWF INC
MOVWF INC2
MOVWF ACOM
MOVWF LEGS
MOVWF SEGUNDO_PASO
MOVWF FIXER
MOVWF INIF
CLRF TURN
CLRF TURN2
CLRF UP
CLRF UP_INC
CLRF DONE
MOVLW 0A0H
MOVWF LED_ON
MOVWF LED_OFF

    ;enceramiento de las variables

    MOVLW 01H
MOVWF VELOCIDAD
```

```
MOVLW d’150’; 150
MOVWF MEDIO
MOVLW d’67’;17
MOVWF PATA1_1 ;PATA UNO PORTA,0 PIN 2
MOVWF PATA2_1 ;PATA DOS PORTC,0 PIN 15
MOVWF PATA3_1 ;PATA TRES PORTC,3 PIN 18
MOVLW d’77’;27
MOVWF PATA4_1 ;PATA CUATRO PORTD,2 PIN 21
MOVWF PATA5_1 ;PATA CINCO PORTC,5 PIN 24
MOVWF PATA6_1 ;PATA SEIS PORTD,4 PIN 27
MOVLW d’63’;13
MOVWF PATA1_2 ;PATA UNO PORTA,1 PIN 3
MOVWF PATA2_2 ;PATA DOS PORTC,1 PIN 16
MOVWF PATA3_2 ;PATA TRES PORTD,0 PIN 19
MOVLW d’81’;d’31’
MOVWF PATA4_2 ;PATA CUATRO PORTD,3 PIN 22
MOVWF PATA5_2 ;PATA CINCO PORTC,6 PIN 25
MOVWF PATA6_2 ;PATA SEIS PORTD,5 PIN 28
MOVLW d’72’;22
MOVWF PATA1_3 ;PATA UNO PORTA,2 PIN 4
MOVWF PATA2_3 ;PATA DOS PORTC,2 PIN 17
MOVWF PATA3_3 ;PATA TRES PORTD,1 PIN 20
MOVLW d’72’
MOVWF PATA4_3 ;PATA CUATRO PORTC,4 PIN 23
MOVWF PATA5_3 ;PATA CINCO PORTC,7 PIN 26
MOVWF PATA6_3 ;PATA SEIS PORTD,6 PIN 29


      MOVLW d’80’
MOVWF CABEZAX ;CABEZA EJE X
MOVWF CABEZAY ;CABEZA EJE Y


      MOVLW d’80’
MOVWF INI1;
MOVWF INI2;
MOVWF INI3;
MOVWF INI4;
MOVWF INI5;
MOVWF INI6;


      ;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
INICIO0
BSF PORTA,0 ;PORTA,0 PIN 2
BSF PORTA,1 ;PORTA,1 PIN 3
BSF PORTA,2 ;PORTA,2 PIN 4
```

```
MOVF INI1,0
XORLW 00H
BTFSC STATUS,Z
GOTO P2
DECF INI1,F
BTFSS STATUS,Z
GOTO SALIDA

    P2 BSF PORTC,0 ;PORTC,0 PIN 15
BSF PORTC,1 ;PORTC,1 PIN 16
BSF PORTC,2 ;PORTC,2 PIN 17
MOVF INI2,0
XORLW 00H
BTFSC STATUS,Z
GOTO P3
DECF INI2,F
BTFSS STATUS,Z
GOTO SALIDA

    P3 BSF PORTC,3 ;PORTC,3 PIN 18
BSF PORTD,0 ;PORTD,0 PIN 19
BSF PORTD,1 ;PORTD,1 PIN 20
MOVF INI3,0
XORLW 00H
BTFSC STATUS,Z
GOTO P4
DECF INI3,F
BTFSS STATUS,Z
GOTO SALIDA

    P4 BSF PORTD,2 ;PORTD,2 PIN 21
BSF PORTD,3 ;PORTD,3 PIN 22
BSF PORTC,4 ;PORTC,4 PIN 23
MOVF INI4,0
XORLW 00H
BTFSC STATUS,Z
GOTO P5
DECF INI4,F
BTFSS STATUS,Z
GOTO SALIDA

    P5 BSF PORTC,5 ;PORTC,5 PIN 24
BSF PORTC,6 ;PORTC,6 PIN 25
BSF PORTC,7 ;PORTC,7 PIN 26
MOVF INI5,0
```

```
XORLW 00H
BTFSC STATUS,Z
GOTO P6
DECF INI5,F
BTFSS STATUS,Z
GOTO SALIDA

    P6 BSF PORTD,4 ;PORTD,4 PIN 27
BSF PORTD,5 ;PORTD,5 PIN 28
BSF PORTD,6 ;PORTD,6 PIN 29
MOVF INI6,0
XORLW 00H
BTFSC STATUS,Z
GOTO SALIDA
DECF INI6,F
BTFSS STATUS,Z
GOTO SALIDA

    CLRF INIF

    GOTO SALIDA
;#################################################################################
;#################################################################################
;#################################################################################
INICIO

    BSF PORTB,0 ;PORTB,0 PIN 33
BSF PORTB,1 ;PORTB,1 PIN 34

    BSF PORTA,0 ;PORTA,0 PIN 2
BSF PORTA,1 ;PORTA,1 PIN 3
BSF PORTA,2 ;PORTA,2 PIN 4

    BSF PORTC,0 ;PORTC,0 PIN 15
BSF PORTC,1 ;PORTC,1 PIN 16
BSF PORTC,2 ;PORTC,2 PIN 17

    BSF PORTC,3 ;PORTC,3 PIN 18
BSF PORTD,0 ;PORTD,0 PIN 19
BSF PORTD,1 ;PORTD,1 PIN 20

    BSF PORTD,2 ;PORTD,2 PIN 21
BSF PORTD,3 ;PORTD,3 PIN 22
```

```
BSF PORTC,4 ;PORTC,4 PIN 23


	BSF PORTC,5 ;PORTC,5 PIN 24
BSF PORTC,6 ;PORTC,6 PIN 25
BSF PORTC,7 ;PORTC,7 PIN 26


	BSF PORTD,4 ;PORTD,4 PIN 27
BSF PORTD,5 ;PORTD,5 PIN 28
BSF PORTD,6 ;PORTD,6 PIN 29
;VELOCIDAD DE LOS MOVIMIENTOS
INCF VELOCIDAD,1
MOVF VELOCIDAD,0
XORLW 05H ;MAYOR NUMERO MAS DELAY
BTFSS STATUS,Z
GOTO SALIDA
MOVLW 04H
MOVWF VELOCIDAD


	;#####################################################################
;CODIGO DE PARARSE
;#####################################################################
UP_INCREMENTO
MOVF UP,0
XORLW d'1'
BTFSS STATUS,Z
GOTO SUBIR ;SALIDA;
MOVF LEGS,0
XORLW d'13'
BTFSS STATUS,Z
GOTO ACOMODARSE; CAMINAR
;#####################################################################
;MANDO
;#####################################################################
MOVF PORTB,0
ANDLW 0XF0;
XORLW 0X10;
BTFSC STATUS,Z
GOTO CAMINAR; ADELANTE
MOVF PORTB,0
ANDLW 0XF0;
XORLW 0X20;
BTFSC STATUS,Z
GOTO RETRO; ATRAS
MOVF PORTB,0
ANDLW 0XF0;
XORLW 0X40;
```

```
BTFSC STATUS,Z
GOTO GIRO_DERECHA; DERECHA
MOVF PORTB,0
ANDLW 0XF0;
XORLW 0X80;
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA; IZQUIERDA
MOVF FIXER,0
XORLW d'1'
BTFSS STATUS,Z
GOTO RECUPERACION
MOVF PORTB,0 ;##############################
ANDLW 0XF0;
XORLW 0X50;
BTFSC STATUS,Z
GOTO GIRO_DER;
MOVF PORTB,0 ;##############################
ANDLW 0XF0;
XORLW 0X30;
BTFSC STATUS,Z
GOTO GIRO_IZQ;
MOVF PORTB,0 ;##############################
ANDLW 0XF0;
XORLW 0X60;
BTFSC STATUS,Z
GOTO GIRO_ARRIBA;
MOVF PORTB,0 ;##############################
ANDLW 0XF0;
XORLW 0X70;
BTFSC STATUS,Z
GOTO GIRO_ABAJO;
GOTO SALIDA


      ;#########################################################################
SUBIR DECF PATA5_2,1
DECF PATA4_2,1
DECF PATA6_2,1
INCF PATA3_2,1
INCF PATA2_2,1
INCF PATA1_2,1
MOVF PATA1_2,0
XORLW d'90';d'100'
BTFSS STATUS,Z
GOTO MOVIMIENTO1
MOVLW d'1'
MOVWF UP
GOTO SALIDA
```

```
MOVIMIENTO1 MOVF UP_INC,0
XORLW d'1'
BTFSS STATUS,Z
GOTO SIGUE_01
GOTO SALIDA
SIGUE_01 DECF PATA5_1,1
DECF PATA4_1,1
DECF PATA6_1,1
INCF PATA3_1,1
INCF PATA2_1,1
INCF PATA1_1,1
MOVF PATA1_1,0
XORLW d'84';d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF UP_INC
GOTO SALIDA
;################################################
;FUNCION PARA ESTABILIZAR LAS CARGAS EN CADA PATA
;################################################
ACOMODARSE MOVLW d'4'
MOVWF VELOCIDAD
MOVF ACOM,0
XORLW d'0'
BTFSS STATUS,Z
GOTO PATA1_A
GOTO SALIDA;
PATA1_A MOVF LEGS,0 ;PATA1_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'1'
BTFSC STATUS,Z
GOTO PATA1_A_AUX
GOTO PATA1_B
PATA1_A_AUX DECF PATA1_2;
MOVF PATA1_2,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'2'
MOVWF LEGS
PATA1_B MOVF LEGS,0
XORLW d'2'
BTFSC STATUS,Z
GOTO PATA1_B_AUX
GOTO PATA2_A
PATA1_B_AUX INCF PATA1_2;
MOVF PATA1_2,0
XORLW d'90'
```

```
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'3'
MOVWF LEGS
PATA2_A MOVF LEGS,0 ;PATA2_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'3'
BTFSC STATUS,Z
GOTO PATA2_A_AUX
GOTO PATA2_B
PATA2_A_AUX DECF PATA2_2;
MOVF PATA2_2,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'4'
MOVWF LEGS
PATA2_B MOVF LEGS,0
XORLW d'4'
BTFSC STATUS,Z
GOTO PATA2_B_AUX
GOTO PATA3_A
PATA2_B_AUX INCF PATA2_2;
MOVF PATA2_2,0
XORLW d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'5'
MOVWF LEGS
PATA3_A MOVF LEGS,0 ;PATA3_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'5'
BTFSC STATUS,Z
GOTO PATA3_A_AUX
GOTO PATA3_B
PATA3_A_AUX DECF PATA3_2;
MOVF PATA3_2,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'6'
MOVWF LEGS
PATA3_B MOVF LEGS,0
XORLW d'6'
BTFSC STATUS,Z
GOTO PATA3_B_AUX
GOTO PATA4_A
PATA3_B_AUX INCF PATA3_2;
MOVF PATA3_2,0
XORLW d'90'
```

```
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'7'
MOVWF LEGS
PATA4_A MOVF LEGS,0 ;PATA4_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'7'
BTFSC STATUS,Z
GOTO PATA4_A_AUX
GOTO PATA4_B
PATA4_A_AUX INCF PATA4_2;
MOVF PATA4_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'8'
MOVWF LEGS
PATA4_B MOVF LEGS,0
XORLW d'8'
BTFSC STATUS,Z
GOTO PATA4_B_AUX
GOTO PATA5_A
PATA4_B_AUX DECF PATA4_2;
MOVF PATA4_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'9'
MOVWF LEGS
PATA5_A MOVF LEGS,0 ;PATA5_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'9'
BTFSC STATUS,Z
GOTO PATA5_A_AUX
GOTO PATA5_B
PATA5_A_AUX INCF PATA5_2;
MOVF PATA5_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'10'
MOVWF LEGS
PATA5_B MOVF LEGS,0
XORLW d'10'
BTFSC STATUS,Z
GOTO PATA5_B_AUX
GOTO PATA6_A
PATA5_B_AUX DECF PATA5_2;
MOVF PATA5_2,0
XORLW d'54'
```

```
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'11'
MOVWF LEGS
PATA6_A MOVF LEGS,0 ;PATA6_2@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
XORLW d'11'
BTFSC STATUS,Z
GOTO PATA6_A_AUX
GOTO PATA6_B
PATA6_A_AUX INCF PATA6_2;
MOVF PATA6_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'12'
MOVWF LEGS
PATA6_B MOVF LEGS,0
XORLW d'12'
BTFSC STATUS,Z
GOTO PATA6_B_AUX
GOTO FIN_ACOMODAR
PATA6_B_AUX DECF PATA6_2;
MOVF PATA6_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'13'
MOVWF LEGS
FIN_ACOMODAR
BSF PORTB,3
CLRF ACOM
GOTO SALIDA
;#####################################################################
;CODIGO DE CAMINAR DE FRENTE
;#####################################################################
CAMINAR BCF PORTB,3
CLRF FIXER
MOVF SEGUNDO_PASO,0
XORLW d'0'
BTFSC STATUS,Z
GOTO SIGUE
MOVF TURN,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_INC
MOVF INC,0
XORLW 01H
BTFSC STATUS,Z
```

```
GOTO INCREMENTO
GOTO DECREMENTO


      INCREMENTO MOVLW 01H ;;;;;;;;;;;;;;;;REVISAR AQUI;;;;;;;;;;;;;;;;;;;;;;;;;
MOVWF INC
INCF PATA5_1,1
DECF PATA3_1,1
DECF PATA1_1,1
INCF PATA5_2,1
DECF PATA3_2,1
DECF PATA1_2,1
MOVF PATA1_2,0
XORLW d'67';36
BTFSS STATUS,Z
GOTO SALIDA
MOVLW 01H
MOVWF TURN
CLRF INC
GIRO_INC INCF PATA2_3,1
DECF PATA6_3,1
DECF PATA4_3,1
DECF PATA3_3,1 ;@@@@@@@2
DECF PATA1_3,1
INCF PATA5_3,1
MOVF PATA5_3,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN


      DECREMENTO DECF PATA5_1,1
INCF PATA3_1,1
INCF PATA1_1,1
DECF PATA5_2,1
INCF PATA3_2,1
INCF PATA1_2,1
MOVF PATA1_2,0
XORLW d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF TURN
CLRF SEGUNDO_PASO


      SIGUE MOVF TURN2,0
XORLW 01H
```

```
BTFSC STATUS,Z
GOTO GIRO_INC2
MOVF INC2,0
XORLW 01H
BTFSC STATUS,Z
GOTO INCREMENTO2
GOTO DECREMENTO2


     INCREMENTO2 INCF PATA4_1,1
INCF PATA6_1,1
DECF PATA2_1,1
DECF PATA2_2,1 ;80
INCF PATA6_2,1 ;64
INCF PATA4_2,1
MOVF PATA4_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF INC2
MOVLW 01H
MOVWF TURN2


     GIRO_INC2 INCF PATA3_3,1 ;72
INCF PATA1_3,1
DECF PATA5_3,1
DECF PATA2_3,1 ;@@@@@@@@@@@@@@
INCF PATA6_3,1
INCF PATA4_3,1
MOVF PATA4_3,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN2


     DECREMENTO2 DECF PATA4_1,1
DECF PATA6_1,1
INCF PATA2_1,1
INCF PATA2_2,1
DECF PATA6_2,1
DECF PATA4_2,1
MOVF PATA4_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF INC
```

```
MOVWF INC2
MOVWF SEGUNDO_PASO
CLRF TURN
CLRF TURN2
GOTO SALIDA
;#########################################################
;CODIGO GIRO A LA DERECHA
;#########################################################
GIRO_DERECHA
BCF PORTB,3
CLRF FIXER
MOVF SEGUNDO_PASO,0
XORLW d'0'
BTFSC STATUS,Z
GOTO GIRO_DERECHA_SIGUE
MOVF TURN,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_DERECHA_GIRO_INC
MOVF INC,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_DERECHA_INCREMENTO
GOTO GIRO_DERECHA_DECREMENTO


      GIRO_DERECHA_INCREMENTO MOVLW 01H
MOVWF INC
INCF PATA4_1,1
INCF PATA6_1,1
DECF PATA2_1,1
DECF PATA2_2,1 ;80
INCF PATA6_2,1 ;64
INCF PATA4_2,1
MOVF PATA4_2,0
XORLW d'77';77
BTFSS STATUS,Z
GOTO SALIDA
MOVLW 01H
MOVWF TURN
CLRF INC
GIRO_DERECHA_GIRO_INC
INCF PATA1_3,1
INCF PATA3_3,1
INCF PATA5_3,1 ;@@@@@@2
DECF PATA2_3,1
DECF PATA4_3,1
DECF PATA6_3,1
```

```
MOVF PATA4_3,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN

    GIRO_DERECHA_DECREMENTO
DECF PATA4_1,1
DECF PATA6_1,1
INCF PATA2_1,1
INCF PATA2_2,1
DECF PATA6_2,1
DECF PATA4_2,1
MOVF PATA4_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF TURN
CLRF SEGUNDO_PASO

    GIRO_DERECHA_SIGUE MOVF TURN2,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_DERECHA_GIRO_INC2
MOVF INC2,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_DERECHA_INCREMENTO2
GOTO GIRO_DERECHA_DECREMENTO2

    GIRO_DERECHA_INCREMENTO2
INCF PATA5_1,1
DECF PATA3_1,1
DECF PATA1_1,1
INCF PATA5_2,1
DECF PATA3_2,1
DECF PATA1_2,1
MOVF PATA1_2,0
XORLW d'67';36
BTFSS STATUS,Z
GOTO SALIDA
CLRF INC2
MOVLW 01H
MOVWF TURN2
```

```
        GIRO_DERECHA_GIRO_INC2
DECF PATA1_3,1
DECF PATA3_3,1 ;72
DECF PATA5_3,1
INCF PATA2_3,1 ;@@@@@@@@@@@@@@
INCF PATA4_3,1
INCF PATA6_3,1
MOVF PATA4_3,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN2


        GIRO_DERECHA_DECREMENTO2
DECF PATA5_1,1
INCF PATA3_1,1
INCF PATA1_1,1
DECF PATA5_2,1
INCF PATA3_2,1
INCF PATA1_2,1
MOVF PATA1_2,0
XORLW d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF INC
MOVWF INC2
MOVWF SEGUNDO_PASO
CLRF TURN
CLRF TURN2
GOTO SALIDA
;#####################################################################
;CODIGO GIRO A LA IZQUIERDA
;#####################################################################
GIRO_IZQUIERDA
BCF PORTB,3
CLRF FIXER
MOVF SEGUNDO_PASO,0
XORLW d'0'
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA_SIGUE
MOVF TURN,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA_GIRO_INC
MOVF INC,0
XORLW 01H
```

```
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA_INCREMENTO
GOTO GIRO_IZQUIERDA_DECREMENTO


      GIRO_IZQUIERDA_INCREMENTO
MOVLW 01H
MOVWF INC
INCF PATA5_1,1
DECF PATA3_1,1
DECF PATA1_1,1
INCF PATA5_2,1
DECF PATA3_2,1
DECF PATA1_2,1
MOVF PATA1_2,0
XORLW d'67';36
BTFSS STATUS,Z
GOTO SALIDA
MOVLW 01H
MOVWF TURN
CLRF INC
GIRO_IZQUIERDA_GIRO_INC
INCF PATA1_3,1
INCF PATA3_3,1
INCF PATA5_3,1 ;@@@@@@@2
DECF PATA2_3,1
DECF PATA4_3,1
DECF PATA6_3,1
MOVF PATA6_3,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN


      GIRO_IZQUIERDA_DECREMENTO
DECF PATA5_1,1
INCF PATA3_1,1
INCF PATA1_1,1
DECF PATA5_2,1
INCF PATA3_2,1
INCF PATA1_2,1
MOVF PATA1_2,0
XORLW d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF TURN
CLRF SEGUNDO_PASO
```

```
        GIRO_IZQUIERDA_SIGUE
MOVF TURN2,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA_GIRO_INC2
MOVF INC2,0
XORLW 01H
BTFSC STATUS,Z
GOTO GIRO_IZQUIERDA_INCREMENTO2
GOTO GIRO_IZQUIERDA_DECREMENTO2


        GIRO_IZQUIERDA_INCREMENTO2
INCF PATA4_1,1
INCF PATA6_1,1
DECF PATA2_1,1
DECF PATA2_2,1 ;80
INCF PATA6_2,1 ;64
INCF PATA4_2,1
MOVF PATA4_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF INC2
MOVLW 01H
MOVWF TURN2


        GIRO_IZQUIERDA_GIRO_INC2
DECF PATA1_3,1
DECF PATA3_3,1
DECF PATA5_3,1 ;@@@@@@@2
INCF PATA2_3,1
INCF PATA4_3,1
INCF PATA6_3,1
MOVF PATA4_3,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN2


        GIRO_IZQUIERDA_DECREMENTO2
DECF PATA4_1,1
DECF PATA6_1,1
INCF PATA2_1,1
INCF PATA2_2,1
```

```
DECF PATA6_2,1
DECF PATA4_2,1
MOVF PATA4_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF INC
MOVWF INC2
MOVWF SEGUNDO_PASO
CLRF TURN
CLRF TURN2
GOTO SALIDA
;*********************************************************************
;CODIGO DE CAMINAR DE RETRO
;*********************************************************************
RETRO BCF PORTB,3
CLRF FIXER
MOVF SEGUNDO_PASO,0
XORLW d'0'
BTFSC STATUS,Z
GOTO RETRO_SIGUE
MOVF TURN,0
XORLW 01H
BTFSC STATUS,Z
GOTO RETRO_GIRO_INC
MOVF INC,0
XORLW 01H
BTFSC STATUS,Z
GOTO RETRO_INCREMENTO
GOTO RETRO_DECREMENTO


    RETRO_INCREMENTO MOVLW 01H
MOVWF INC
INCF PATA5_1,1
DECF PATA3_1,1
DECF PATA1_1,1
INCF PATA5_2,1
DECF PATA3_2,1
DECF PATA1_2,1
MOVF PATA1_2,0
XORLW d'67';36
BTFSS STATUS,Z
GOTO SALIDA
MOVLW 01H
MOVWF TURN
CLRF INC
```

```
RETRO_GIRO_INC INCF PATA1_3,1
DECF PATA2_3,1
INCF PATA3_3,1
INCF PATA4_3,1
DECF PATA5_3,1
INCF PATA6_3,1
MOVF PATA6_3,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN

    RETRO_DECREMENTO DECF PATA5_1,1
INCF PATA3_1,1
INCF PATA1_1,1
DECF PATA5_2,1
INCF PATA3_2,1
INCF PATA1_2,1
MOVF PATA1_2,0
XORLW d'90'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF TURN
CLRF SEGUNDO_PASO

    RETRO_SIGUE MOVF TURN2,0
XORLW 01H
BTFSC STATUS,Z
GOTO RETRO_GIRO_INC2
MOVF INC2,0
XORLW 01H
BTFSC STATUS,Z
GOTO RETRO_INCREMENTO2
GOTO RETRO_DECREMENTO2

    RETRO_INCREMENTO2 INCF PATA4_1,1
INCF PATA6_1,1
DECF PATA2_1,1
DECF PATA2_2,1 ;80
INCF PATA6_2,1 ;64
INCF PATA4_2,1
MOVF PATA4_2,0
XORLW d'77'
BTFSS STATUS,Z
GOTO SALIDA
```

```
CLRF INC2
MOVLW 01H
MOVWF TURN2


      RETRO_GIRO_INC2
DECF PATA1_3,1
INCF PATA2_3,1 ;@@@@@@@@@@@@@@
DECF PATA3_3,1 ;72
DECF PATA4_3,1
INCF PATA5_3,1
DECF PATA6_3,1
MOVF PATA6_3,0
XORLW d'67'
BTFSS STATUS,Z
GOTO SALIDA
CLRF TURN2


      RETRO_DECREMENTO2 DECF PATA4_1,1
DECF PATA6_1,1
INCF PATA2_1,1
INCF PATA2_2,1
DECF PATA6_2,1
DECF PATA4_2,1
MOVF PATA4_2,0
XORLW d'54'
BTFSS STATUS,Z
GOTO SALIDA
MOVLW d'1'
MOVWF INC
MOVWF INC2
MOVWF SEGUNDO_PASO
CLRF TURN
CLRF TURN2
GOTO SALIDA
;————————————GIRO DERECHA X————————————
;MOVLW d'67'
;MOVWF CABEZAX ;CABEZA EJE X
;MOVWF CABEZAY ;CABEZA EJE Y


      GIRO_DER MOVLW d'90'
SUBWF CABEZAX,0
BTFSC STATUS,Z;
GOTO SALIDA
INCF CABEZAX,1
GOTO SALIDA
GIRO_IZQ MOVLW d'60'
```

```
SUBWF CABEZAX,0
BTFSC STATUS,Z;
GOTO SALIDA
DECF CABEZAX,1
GOTO SALIDA


      GIRO_ARRIBA MOVLW d'90'
SUBWF CABEZAY,0
BTFSC STATUS,Z;
GOTO SALIDA
INCF CABEZAY,1
GOTO SALIDA
GIRO_ABAJO MOVLW d'60'
SUBWF CABEZAY,0
BTFSC STATUS,Z;
GOTO SALIDA
DECF CABEZAY,1
GOTO SALIDA


      ;#############################################################
;RECUPERACION POSICION
;#############################################################
RECUPERACION
MOVF PATA1_1,0
SUBLW d'84'
BTFSC STATUS,Z
GOTO FIX2_AUX ;F
INCF PATA1_1,1
GOTO FIX2
FIX2_AUX INCF DONE,1
FIX2 MOVF PATA2_1,0
SUBLW d'84'
BTFSC STATUS,Z
GOTO FIX3_AUX ;F
INCF PATA2_1,1
GOTO FIX3
FIX3_AUX INCF DONE,1
FIX3 MOVF PATA3_1,0
SUBLW d'84'
BTFSC STATUS,Z
GOTO FIX4_AUX ;F
INCF PATA3_1,1
GOTO FIX4
FIX4_AUX INCF DONE,1
FIX4 MOVF PATA4_1,0
SUBLW d'60'
BTFSC STATUS,Z
```

```
GOTO FIX5_AUX ;F
DECF PATA4_1,1
GOTO FIX5
FIX5_AUX INCF DONE,1
FIX5 MOVF PATA5_1,0
SUBLW d'60'
BTFSC STATUS,Z
GOTO FIX6_AUX ;F
DECF PATA5_1,1
GOTO FIX6
FIX6_AUX INCF DONE,1
FIX6 MOVF PATA6_1,0
SUBLW d'60'
BTFSC STATUS,Z
GOTO FIX7_AUX ;F
DECF PATA6_1,1
GOTO FIX7
FIX7_AUX INCF DONE,1
FIX7 MOVF PATA1_2,0
SUBLW d'90'
BTFSC STATUS,Z
GOTO FIX8_AUX ;F
INCF PATA1_2,1
GOTO FIX8
FIX8_AUX INCF DONE,1
FIX8 MOVF PATA2_2,0
SUBLW d'90'
BTFSC STATUS,Z
GOTO FIX9_AUX ;F
INCF PATA2_2,1
GOTO FIX9
FIX9_AUX INCF DONE,1
FIX9 MOVF PATA3_2,0
SUBLW d'90'
BTFSC STATUS,Z
GOTO FIX10_AUX ;F
INCF PATA3_2,1
GOTO FIX10
FIX10_AUX INCF DONE,1
FIX10 MOVF PATA4_2,0
SUBLW d'54'
BTFSC STATUS,Z
GOTO FIX11_AUX ;F
DECF PATA4_2,1
GOTO FIX11
FIX11_AUX INCF DONE,1
FIX11 MOVF PATA5_2,0
SUBLW d'54'
```

```
BTFSC STATUS,Z
GOTO FIX12_AUX ;F
DECF PATA5_2,1
GOTO FIX12
FIX12_AUX INCF DONE,1
FIX12 MOVF PATA6_2,0
SUBLW d'54'
BTFSC STATUS,Z
GOTO FIX13_AUX ;F
DECF PATA6_2,1
GOTO FIX13
FIX13_AUX INCF DONE,1
FIX13 MOVF PATA1_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO FIX14_AUX ;F
BTFSS STATUS,C
DECF PATA1_3,1
BTFSC STATUS,C
INCF PATA1_3,1
GOTO FIX14
FIX14_AUX INCF DONE,1
FIX14 MOVF PATA2_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO FIX15_AUX ;F
BTFSS STATUS,C
DECF PATA2_3,1
BTFSC STATUS,C
INCF PATA2_3,1
GOTO FIX15
FIX15_AUX INCF DONE,1
FIX15 MOVF PATA3_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO FIX16_AUX ;F
BTFSS STATUS,C
DECF PATA3_3,1
BTFSC STATUS,C
INCF PATA3_3,1
GOTO FIX16
FIX16_AUX INCF DONE,1
FIX16 MOVF PATA4_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO FIX17_AUX ;F
BTFSS STATUS,C
DECF PATA4_3,1
```

```
BTFSC STATUS,C
INCF PATA4_3,1
GOTO FIX17
FIX17_AUX INCF DONE,1
FIX17 MOVF PATA5_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO FIX18_AUX ;F
BTFSS STATUS,C
DECF PATA5_3,1
BTFSC STATUS,C
INCF PATA5_3,1
GOTO FIX18
FIX18_AUX INCF DONE,1
FIX18 MOVF PATA6_3,0
SUBLW d'72'
BTFSC STATUS,Z
GOTO HECHO ;F
BTFSS STATUS,C
DECF PATA6_3,1
BTFSC STATUS,C
INCF PATA6_3,1
HECHO INCF DONE,1
MOVF DONE,0
XORLW d'18'
BTFSS STATUS,Z
GOTO NO
BSF PORTB,3
MOVLW 01H
MOVWF FIXER
MOVWF SEGUNDO_PASO
MOVWF INC
MOVWF INC2
CLRF TURN
CLRF TURN2
CLRF DONE
NO CLRF DONE
SALIDA
CALL DELAY1 ;DDD EEEE L A Y Y
CALL DELAY2 ;D D EE L AAA Y
MOVF INIF,0 ;DDD EEEE LLLL A A Y
XORLW 00H
BTFSS STATUS,Z
GOTO INICIO0
GOTO INICIO


    DELAY1: MOVF MEDIO,0 ;1 CICLO 40=2.00 ms//20=1.00 ms//30=1.5 ms
```

```
MOVWF CONT ;1 CICLO
CICLO1_1 NOP ;1 CICLO
DECF CONT,F
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;PATA UNO
MOVF CONT,0
XORWF PATA1_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT1
GOTO PATA1_UNO
PATA1_UNO BCF PORTA,0;;;;;;
GOTO NEXT1
NEXT1 MOVF CONT,0
XORWF PATA1_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT2
GOTO PATA1_DOS
PATA1_DOS BCF PORTA,1;;;;;;
GOTO NEXT2
NEXT2 MOVF CONT,0
XORWF PATA1_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT2_UNO
GOTO PATA1_TRES
PATA1_TRES BCF PORTA,2;;;;;;
GOTO NEXT2_UNO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PATA DOS
NEXT2_UNO MOVF CONT,0
XORWF PATA2_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT2_DOS
GOTO PATA2_UNO
PATA2_UNO BCF PORTC,0 ;PUERTO_C0
GOTO NEXT2_DOS
NEXT2_DOS MOVF CONT,0
XORWF PATA2_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT2_TRES
GOTO PATA2_DOS
PATA2_DOS BCF PORTC,1 ;PUERTO_C1
GOTO NEXT2_TRES
NEXT2_TRES MOVF CONT,0
XORWF PATA2_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT3_UNO
GOTO PATA2_TRES
PATA2_TRES BCF PORTC,2 ;PUERTO_C2
GOTO NEXT3_UNO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PATA TRES
```

```
NEXT3_UNO MOVF CONT,0
XORWF PATA3_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT3_DOS
GOTO PATA3_UNO
PATA3_UNO BCF PORTC,3 ;PUERTO_C0
GOTO NEXT3_DOS
NEXT3_DOS MOVF CONT,0
XORWF PATA3_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT3_TRES
GOTO PATA3_DOS
PATA3_DOS BCF PORTD,0 ;PUERTO_C1
GOTO NEXT3_TRES
NEXT3_TRES MOVF CONT,0
XORWF PATA3_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT4_UNO
GOTO PATA3_TRES
PATA3_TRES BCF PORTD,1 ;PUERTO_C2
GOTO NEXT4_UNO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PATA CUATRO
NEXT4_UNO MOVF CONT,0
XORWF PATA4_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT4_DOS
GOTO PATA4_UNO
PATA4_UNO BCF PORTD,2 ;PUERTO_C0
GOTO NEXT4_DOS
NEXT4_DOS MOVF CONT,0
XORWF PATA4_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT4_TRES
GOTO PATA4_DOS
PATA4_DOS BCF PORTD,3 ;PUERTO_C1
GOTO NEXT4_TRES
NEXT4_TRES MOVF CONT,0
XORWF PATA4_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT5_UNO
GOTO PATA4_TRES
PATA4_TRES BCF PORTC,4 ;PUERTO_C2
GOTO NEXT5_UNO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PATA CINCO
NEXT5_UNO MOVF CONT,0
XORWF PATA5_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT5_DOS
```

```
GOTO PATA5_UNO
PATA5_UNO BCF PORTC,5 ;PUERTO_C0
GOTO NEXT5_DOS
NEXT5_DOS MOVF CONT,0
XORWF PATA5_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT5_TRES
GOTO PATA5_DOS
PATA5_DOS BCF PORTC,6 ;PUERTO_C1
GOTO NEXT5_TRES
NEXT5_TRES MOVF CONT,0
XORWF PATA5_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT6_UNO
GOTO PATA5_TRES
PATA5_TRES BCF PORTC,7 ;PUERTO_C2
GOTO NEXT6_UNO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PATA SEIS
NEXT6_UNO MOVF CONT,0
XORWF PATA6_1,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT6_DOS
GOTO PATA6_UNO
PATA6_UNO BCF PORTD,4 ;PUERTO_C0
GOTO NEXT6_DOS
NEXT6_DOS MOVF CONT,0
XORWF PATA6_2,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXT6_TRES
GOTO PATA6_DOS
PATA6_DOS BCF PORTD,5 ;PUERTO_C1
GOTO NEXT6_TRES
NEXT6_TRES MOVF CONT,0
XORWF PATA6_3,0 ;VARIABLE
BTFSS STATUS,Z
GOTO NEXTCABX
GOTO PATA6_TRES
PATA6_TRES BCF PORTD,6 ;PUERTO_C2
GOTO NEXTCABX
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
NEXTCABX MOVF CONT,0
XORWF CABEZAX,0
BTFSS STATUS,Z
GOTO NEXTCABY
GOTO CABEZA_X
CABEZA_X BCF PORTB,0 ;PUERTO B0
GOTO NEXTCABY
NEXTCABY MOVF CONT,0
```

```
XORWF CABEZAY,0
BTFSS STATUS,Z
GOTO DENUEVO
GOTO CABEZA_Y
CABEZA_Y BCF PORTB,1 ;PUERTO B1
GOTO DENUEVO
DENUEVO
MOVF CONT,0
XORLW 00H
BTFSS STATUS,Z
GOTO CICLO1_1
RETURN


    DELAY2: MOVLW d'70';70 ;1 CICLO 72,255
MOVWF CONT2_1 ;1 CICLO
CICLO2_2 MOVLW d'239'; 240
MOVWF CONT2_2
CICLO2_1 NOP ;1 CICLO
DECF CONT2_2,F ;1 CICLO
BTFSS STATUS,Z ;1 CICLO PARA FINALIZAR 2 CICLOS
GOTO CICLO2_1 ;2 CICLOS
DECF CONT2_1,F
BTFSS STATUS,Z
GOTO CICLO2_2
MOVF LED_ON,0
XORLW 00H
BTFSS STATUS,Z
GOTO LED_PRENDER;
MOVF LED_OFF,0
XORLW 00H
BTFSS STATUS,Z
GOTO LED_APAGAR
MOVLW 0A0H
MOVWF LED_ON
MOVWF LED_OFF
FIN RETURN
LED_APAGAR BCF PORTB,2
DECF LED_OFF
GOTO FIN
LED_PRENDER BSF PORTB,2
DECF LED_ON
GOTO FIN
END
```

8.3. Código Arduino

```
#include <SoftwareSerial.h>
#include<DHT11.h>
SoftwareSerial BT1(4,2); // RX, TX
char NOMBRE[8]="HEXAPOD";
int d = 9600;
int PIN =5;          //sensor de temperatura
DHT11 dht11(PIN);
int var =0;
byte a = 0;
int err;
float temp, hum;
String readString;

int pin_mq = 7;    //sensor de gas


void setup() {
  // put your setup code here, to run once:

      pinMode(7,INPUT); //Pin MQ2
      pinMode(6,OUTPUT); //BLuetooth
      //Pines de control
      pinMode(8,OUTPUT);
      pinMode(9,OUTPUT);
      pinMode(10,OUTPUT);
      pinMode(11,OUTPUT);
      digitalWrite(8,LOW);
      digitalWrite(9,LOW);
      digitalWrite(10,LOW);
      digitalWrite(11,LOW);

      digitalWrite(6,HIGH);
```

```
        delay(1000);
        Serial.begin(d);
        //Serial.println("Enter AT commands:");
        //Modo AT modulo Bluetooth.
        //Serial.println("Esperando comandos AT:");
        BT1.begin(9600);
        //BT1.print("AT+NAMEHEXPOD");
       /*
        while(millis() < 30000) // wait for a reply for up to 30 seconds
        {
         while(BT1.available() > 0)
         {
         char aChar = BT1.read();
         Serial.print(aChar);
         }
        }
        */
}

void loop() {
  // put your main code here, to run repeatedly:
         var=0;

         if (BT1.available())
           {
           //Serial.write(BT1.read());
           var = BT1.read()-48;
           Serial.println(var);
           }
           if(var==9)
           {
```

```
  Serial.println("PARO");   //01
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
if(var==1)
{
  Serial.println("Adelante");   //01
  digitalWrite(8,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
if(var==2)
{
  Serial.println("Atras");      //02
  digitalWrite(8,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
if(var==3)
{
  Serial.println("Izquierda");   // 08
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
}
if(var==4)
{
```

```
  Serial.println("Derecha");      //04
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}
if(var==5)
{
  Serial.println("Giro_Izq");     //3
  digitalWrite(8,HIGH);
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
if(var==6)
{
  Serial.println("Giro_Der");     //5
  digitalWrite(8,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}
 if(var==7)
{
  Serial.println("Giro_Up");     //6
  digitalWrite(8,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}
if(var==8)
{
```

```
            Serial.println("Giro_Down");   //7
            digitalWrite(8,HIGH);
            digitalWrite(9,HIGH);
            digitalWrite(10,HIGH);
            digitalWrite(11,LOW);
          }
      if (Serial.available())
      {
        Serial.read();
      }
       /*
           for(int i=0;i<100;i++)
           {
           BT1.println(i);
           Serial.println(i);
           delay(500);
           }

        */
  //SENSOR DE TEMPERATURA Y HUMEDAD;
        if((err = dht11.read(hum, temp)) == 0)    // Si devuelve 0 es que ha leido bien
          {
            Serial.print("Temperatura: ");
            Serial.print(temp);
            Serial.print(" Humedad: ");
            Serial.print(hum);
            Serial.println();
          }
        else
          {
            Serial.println();
            Serial.print("Error Num :");
```

```
            Serial.print(err);
            Serial.println();
          }
        delay(1000);

  //SENSOR MQ2 GASES.
        boolean mq_estado = digitalRead(pin_mq);//Leemos el sensor
        if(mq_estado) //si la salida del sensor es 1
         {
           Serial.println("Sin presencia de Metano");
         }
          else //si la salida del sensor es 0
         {
           Serial.println("Metano detectado");
         }
        delay(100);

}
```

8.4. Código Android Studio

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.fastc240.segundo">

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="24" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
/>

    <permission android:name="android.permission.BLUETOOTH"
android:label="BLUETOOTH" />
    <permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <uses-permission
android:name="android.permission.SYSTEM_ALERT_WINDOW" />

    <android:uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <android:uses-permission
android:name="android.permission.READ_PHONE_STATE" />
    <android:uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Bluetooth"
        android:supportsRtl="true"
        android:theme="@android:style/Theme.DeviceDefault">
        <activity android:name=".MainActivity">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Main2Activity"
            android:multiprocess="true"
            android:label="@string/app_name">
        </activity>
        <service android:name=".FloatingWindow" />
    </application>

</manifest>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.fastc240.segundo.MainActivity">

    <TextView
        android:text="ROBOT PARA REVISION DE CÁMARAS DE REGISTRO
TELCONET"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView2"
        android:textSize="24sp"
        android:textAlignment="center"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:text="INICIAR COMUNICACIÓN"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/start"
        android:layout_marginBottom="20dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@mipmap/logo_telconet"
        android:id="@+id/imageView2"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:src="@mipmap/logo_telconet" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@mipmap/symbol_wifi"
        android:id="@+id/imageView8"
        android:layout_above="@+id/start"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="27dp"
        android:src="@mipmap/symbol_wifi" />

    <TextView
        android:text="DEPARTAMENTO DE OPERACIONES URBANAS      FIBRA
ÓPTICA"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:textSize="24sp"
        android:textAlignment="center"
        android:layout_above="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="14dp" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.fastc240.segundo;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;



public class MainActivity extends Activity{
    private Button b;

    int ACTIVA_BLUETOOTH = 1;

    BluetoothAdapter Adaptador_bluetooth = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



       b = (Button)findViewById(R.id.start);
       b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Adaptador_bluetooth =
BluetoothAdapter.getDefaultAdapter();
                Set<BluetoothDevice> pairedDevices =
Adaptador_bluetooth.getBondedDevices();

               if(Adaptador_bluetooth == null){
                   Toast.makeText(getApplicationContext(), "El
dispositivo no esta conectado", Toast.LENGTH_LONG).show();

               }
               else if (!Adaptador_bluetooth.isEnabled()){
                   Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                   startActivityForResult(enableBtIntent,
ACTIVA_BLUETOOTH);

               }
                if(Adaptador_bluetooth.isEnabled()){
                    Intent pantalla_listas = new
Intent(MainActivity.this, Main2Activity.class);
                    startActivity(pantalla_listas);


                    }
```

```
                }


//
        });
    }
}
```

Main2Activity.java

```
package com.example.fastc240.segundo;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Handler;
import android.os.SystemClock;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;

public class Main2Activity extends Activity{
    private ListView listado;
    private ArrayList<String> mDispositivo = new ArrayList<String>();
    private ArrayAdapter<String> mBTArrayAdapter;
    private TextView mBluetoothStatus;
    private TextView mReadBuffer;
    private Handler mHandler;
    private Set<BluetoothDevice> mPairedDevices;

    //public final static String dato = "";

    private BluetoothSocket mBTSocket = null; // bi-directional
client-to-client data path
    public  static ConnectedThread mConnectedThread = null; //
bluetooth background worker thread to send and receive data
```

```
    public  static String readMessage = null;

    private static final UUID BTMODULEUUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); // "random"
unique identifier

    private final static int MESSAGE_READ = 2; // used in bluetooth
handler to identify message update
    private final static int REQUEST_ENABLE_BT = 1; // used to
identify adding bluetooth names
    private final static int CONNECTING_STATUS = 3; // used in
bluetooth handler to identify message status

    private BluetoothAdapter AdaptadorBluetooth;
    private Button boton;

    public Main2Activity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        boton = (Button) findViewById(R.id.button);
        AdaptadorBluetooth = BluetoothAdapter.getDefaultAdapter();
        AdaptadorBluetooth.startDiscovery();
        mBluetoothStatus =
(TextView)findViewById(R.id.bluetoothStatus);
        mReadBuffer = (TextView)findViewById(R.id.ReadBuffer);

        mBTArrayAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);

        listado = (ListView) findViewById(R.id.lista);
        listado.setAdapter(mBTArrayAdapter);
        listado.setOnItemClickListener(mDeviceClickListener);


            mPairedDevices = AdaptadorBluetooth.getBondedDevices();
            if(AdaptadorBluetooth.isEnabled()) {
                // put it's one to the adapter
                for (BluetoothDevice device : mPairedDevices)
                    mBTArrayAdapter.add(device.getName() + "\n" +
device.getAddress());

                Toast.makeText(getApplicationContext(), "Show Paired
Devices", Toast.LENGTH_SHORT).show();
            }
            else
                Toast.makeText(getApplicationContext(), "Bluetooth not
on", Toast.LENGTH_SHORT).show();


        ////////////////////////////codigo insertado
        mHandler = new Handler(){
            public void handleMessage(android.os.Message msg){
                if(msg.what == MESSAGE_READ){
                    //String readMessage = null;
                    try {
                        readMessage = new String((byte[]) msg.obj,
"UTF-8");
```

```
                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }
                   mReadBuffer.setText(readMessage);
                }

                if(msg.what == CONNECTING_STATUS){
                    if(msg.arg1 == 1) {
                        mBluetoothStatus.setText("Connected to Device:
" + (String) (msg.obj));
                        startService(new Intent(Main2Activity.this,
FloatingWindow.class));
                    }
                    else
                        mBluetoothStatus.setText("Connection Failed");
                }
            }
        };



        ////////////////////////////código insertado

//        ArrayAdapter<String> adaptador = new
ArrayAdapter<String>(this, android.R.layout.simple list item 1,
personas);
//        listado.setAdapter(adaptador);
//        listado.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
//            @Override
//            public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
//                Toast.makeText(getApplicationContext(), "posición "
+ position, Toast.LENGTH SHORT).show();
        boton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
//
//            }
//        });


        ////////////////////////////////////////////////////
//        Intent intent = getIntent();
//        String  dato = intent.getStringExtra(FloatingWindow.dato);
//
//            if (mBTArrayAdapter == null) {
//                // Device does not support Bluetooth
//                mBluetoothStatus.setText("Status: Bluetooth not
found");
//                Toast.makeText(getApplicationContext(), "Bluetooth
device not found!", Toast.LENGTH_SHORT).show();
//            }
//
//            else {
//
//
```

```
//                          if (mConnectedThread != null){
//                              //First check to make sure thread
created
//
//
//                              mConnectedThread.write(dato);
//                          }
//
//
//         }
        ////////////////////////////////////////////////////




    }


    final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            String action= intent.getAction();
            Toast.makeText(getApplicationContext(), "bluetooth ",
Toast.LENGTH_SHORT).show();
            if(BluetoothDevice.ACTION_FOUND.equals(action)){

                BluetoothDevice dispositivo =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                mDispositivo.add(dispositivo.getName()+ "\n" +
dispositivo.getAddress());

                //Log.i("BT", dispositivo.getName() + "\n" +
dispositivo.getAddress());
                //ArrayAdapter<String> adaptador = new
ArrayAdapter<String>(android.R.layout.simple_list_item_1, context,
mDispositivo);
                listado.setAdapter(new
ArrayAdapter<String>(context,android.R.layout.simple_list_item_1,
mDispositivo));
            }

        }
    };
    @Override
    protected void onDestroy() {
        unregisterReceiver(mReceiver);
        super.onDestroy();
    }
    public AdapterView.OnItemClickListener mDeviceClickListener = new
AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2,
long arg3) {

            if(!AdaptadorBluetooth.isEnabled()) {
                Toast.makeText(getBaseContext(), "Bluetooth not on",
Toast.LENGTH_SHORT).show();
                return;
            }

            mBluetoothStatus.setText("Connecting...");
```

```
            // Get the device MAC address, which is the last 17 chars
in the View
            String info = ((TextView) v).getText().toString();
            final String address = info.substring(info.length() - 17);
            final String name = info.substring(0,info.length() - 17);

            // Spawn a new thread to avoid blocking the GUI one
            new Thread()
            {
                public void run() {
                    boolean fail = false;

                    BluetoothDevice device =
AdaptadorBluetooth.getRemoteDevice(address);

                    try {
                        mBTSocket = createBluetoothSocket(device);
                    } catch (IOException e) {
                        fail = true;
                        Toast.makeText(getBaseContext(), "Socket
creation failed", Toast.LENGTH_SHORT).show();
                    }
                    // Establish the Bluetooth socket connection.
                    try {
                        mBTSocket.connect();
                    } catch (IOException e) {
                        try {
                            fail = true;
                            mBTSocket.close();
                            mHandler.obtainMessage(CONNECTING_STATUS,
-1, -1)
                                    .sendToTarget();
                        } catch (IOException e2) {
                            //insert code to deal with this
                            Toast.makeText(getBaseContext(), "Socket
creation failed", Toast.LENGTH_SHORT).show();
                        }
                    }
                    if(fail == false) {
                        mConnectedThread = new
ConnectedThread(mBTSocket);
                        mConnectedThread.start();

                        mHandler.obtainMessage(CONNECTING_STATUS, 1, -
1, name)
                                .sendToTarget();
                    }
                }
            }.start();
        }
    };

    private BluetoothSocket createBluetoothSocket(BluetoothDevice
device) throws IOException {
        return
device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        //creates secure outgoing connection with BT device using UUID
    }
    public class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
```

```
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket) {
            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            // Get the input and output streams, using temp objects
because
            // member streams are final
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[1024];  // buffer store for the
stream
            int bytes; // bytes returned from read()

            // Keep listening to the InputStream until an exception
occurs
            while (true) {
                try {
                    // Read from the InputStream
                    bytes = mmInStream.read(buffer);
                    if(bytes != 0) {
                        SystemClock.sleep(100);
                        mmInStream.read(buffer);
                    }
                    // Send the obtained bytes to the UI activity

                    mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
buffer)
                            .sendToTarget();
                } catch (IOException e) {
                    break;
                }
            }
        }

        /* Call this from the main activity to send data to the remote
device */
        public void write(String input) {
            byte[] bytes = input.getBytes();           //converts
entered String into bytes
            try {
                mmOutStream.write(bytes);
            } catch (IOException e) { }
        }

        /* Call this from the main activity to shutdown the connection
*/
        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
```

```
        }
    }
}
```

FloatingWindows.java

```
package com.example.fastc240.segundo;

/**
 * Created by FAST C240 on 07/03/2017.
 */
import android.app.Service;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.PixelFormat;
import android.graphics.drawable.Drawable;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.support.annotation.IntegerRes;
import android.support.annotation.Nullable;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;

import static
com.example.fastc240.segundo.Main2Activity.mConnectedThread;
import static com.example.fastc240.segundo.Main2Activity.readMessage;


public class FloatingWindow extends Service{
    private WindowManager wm;
    private FrameLayout ll;
    private Button stop;
    private Button adelante;
    private Button atras;
    private Button izquierda;
    private Button derecha;
    private Button paro;

    private TextView camara;
```

```
    private ImageButton giro_izquierda;
    private ImageButton giro_derecha;
    private ImageButton arriba;
    private ImageButton abajo;

    private TextView temperatura;
    private TextView temp;
    private TextView gas;
    private TextView valor_gas;
    private TextView humedad;
    private TextView valor_humedad;
    private TextView titulo;

    private ArrayAdapter<String> mBTArrayAdapter;
    private TextView mBluetoothStatus;
    private TextView mReadBuffer;
    private Handler mHandler;

    public final static String dato ="";

    private final static int MESSAGE_READ = 2; // used in bluetooth
handler to identify message update
    private final static int CONNECTING_STATUS = 3; // used in
bluetooth handler to identify message status


    //private ConnectedThread mConnectedThread; // bluetooth
background worker thread to send and receive data
    private BluetoothSocket mBTSocket = null; // bi-directional
client-to-client data path
    private BluetoothAdapter AdaptadorBluetooth;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        wm = (WindowManager) getSystemService(WINDOW_SERVICE);
        ll = new FrameLayout(this);



        stop = new Button(this);
        adelante = new Button(this);
        atras = new Button(this);
        izquierda = new Button(this);
        derecha = new Button(this);
        paro = new Button(this);

        temperatura = new TextView(this);
        temp = new TextView(this);
        gas = new TextView(this);
        valor_gas = new TextView(this);
        humedad=new TextView(this);
        valor_humedad = new TextView(this);
        titulo = new TextView(this);
        camara = new TextView(this);
```

```
        giro_izquierda = new ImageButton(this);
        giro_derecha = new ImageButton(this);
        arriba = new ImageButton(this);;
        abajo = new ImageButton(this);;

        int h = 600;
        int w = 150;

        ViewGroup.LayoutParams btnParameters = new
ViewGroup.LayoutParams(100, 100);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        stop.setX(0);
        stop.setY(0);
        stop.setPadding(0, 0, 0, 0);
        stop.setText("X");
        stop.setTextColor(Color.BLACK);
        stop.setTextSize(10);
        stop.setLayoutParams(btnParameters);

        //ADELANTE
        ViewGroup.LayoutParams btn1Parameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        adelante.setX(150);
        adelante.setY(100);
        adelante.setPadding(0, 0, 0, 0);
        adelante.setText("A");
        adelante.setTextColor(Color.BLACK);
        adelante.setTextSize(15);
        adelante.setLayoutParams(btn1Parameters);
        //ATRAS
        ViewGroup.LayoutParams btn2Parameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        atras.setX(150);
        atras.setY(400);
        atras.setPadding(0, 0, 0, 0);
        atras.setText("B");
        atras.setTextColor(Color.BLACK);
        atras.setTextSize(15);
        atras.setLayoutParams(btn2Parameters);
        //IZQUIERDA
        ViewGroup.LayoutParams btn3Parameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP CONTENT,ViewGroup.LayoutParams.WRAP CONT
ENT);
        izquierda.setX(0);
        izquierda.setY(250);
        izquierda.setPadding(0, 0, 0, 0);
        izquierda.setText("I");
        izquierda.setTextColor(Color.BLACK);
        izquierda.setTextSize(15);
        izquierda.setLayoutParams(btn3Parameters);
        //DERECHA
        ViewGroup.LayoutParams btn4Parameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
```

```
        derecha.setX(290);
        derecha.setY(250);
        derecha.setPadding(0, 0, 0, 0);
        derecha.setText("D");
        derecha.setTextColor(Color.BLACK);
        derecha.setTextSize(15);
        derecha.setLayoutParams(btn4Parameters);

        //PARO
        ViewGroup.LayoutParams btnParoParameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP CONTENT,ViewGroup.LayoutParams.WRAP CONT
ENT);
        paro.setX(150);
        paro.setY(250);
        paro.setPadding(0, 0, 0, 0);
        paro.setText("0");
        paro.setTextColor(Color.BLACK);
        paro.setTextSize(15);
        paro.setLayoutParams(btnParoParameters);

        //CONTROLES DE CAMARA
        ViewGroup.LayoutParams titulo_cam = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
//ViewGroup.LayoutParams.WRAP CONTENT,ViewGroup.LayoutParams.WRAP CONT
ENT);
        camara.setX(50);
        camara.setY(h-50);
        camara.setText("CONTROL DE CAMARA");
        camara.setTextColor(Color.WHITE);
        camara.setBackgroundColor(Color.BLACK);
        camara.setTextSize(16);
        camara.setLayoutParams(titulo_cam);

        //GIRO IZQUIERDA    ************************

        ViewGroup.LayoutParams btn_giParameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        giro_izquierda.setImageResource(R.mipmap.izquierda_cam);
        giro_izquierda.setX(0);
        giro_izquierda.setY(h);
        giro_izquierda.setLayoutParams(btn_giParameters);
        //GIRO DERECHA
        ViewGroup.LayoutParams btn_gdParameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        giro_derecha.setImageResource(R.mipmap.derecha_cam);
        giro_derecha.setX(150);
        giro_derecha.setY(h);
        giro_derecha.setLayoutParams(btn_gdParameters);

        //GIRO ARRIBA
        ViewGroup.LayoutParams btn_arParameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        arriba.setImageResource(R.mipmap.arriba_cam);
```

```
        arriba.setX(300);
        arriba.setY(h);
        arriba.setLayoutParams(btn_arParameters);

        //GIRO ABAJO
        ViewGroup.LayoutParams btn_abParameters = new
ViewGroup.LayoutParams(150, 150);
//ViewGroup.LayoutParams.WRAP CONTENT,ViewGroup.LayoutParams.WRAP CONT
ENT);
        abajo.setImageResource(R.mipmap.abajo_cam);
        abajo.setX(450);
        abajo.setY(h);
        abajo.setLayoutParams(btn_abParameters);

        //TEMPERATURA
        ViewGroup.LayoutParams btn5Parameters = new
ViewGroup.LayoutParams(350, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        temperatura.setX(450);
        temperatura.setY(100);
        temperatura.setPadding(0, 0, 0, 0);
        temperatura.setText("Temperatura");
        temperatura.setTextColor(Color.WHITE);
        temperatura.setTextSize(15);
        temperatura.setLayoutParams(btn5Parameters);
        //VALOR TEMPERATURA
        ViewGroup.LayoutParams btn6Parameters = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        temp.setX(500);
        temp.setY(200);
        //temp.setPadding(40, 30, 15, 25);
        temp.setText("23,0");
        temp.setTextColor(Color.BLACK);
        temp.setBackgroundColor(Color.WHITE);
        temp.setTextSize(15);
        temp.setLayoutParams(btn6Parameters);
        //GAS METANO
        ViewGroup.LayoutParams btn7Parameters = new
ViewGroup.LayoutParams(400, 150);
//ViewGroup.LayoutParams.WRAP CONTENT,ViewGroup.LayoutParams.WRAP CONT
ENT);
        gas.setX(750);
        gas.setY(100);
        gas.setPadding(0, 0, 0, 0);
        gas.setText("% Gas Metáno");
        gas.setTextColor(Color.WHITE);
        gas.setTextSize(15);
        gas.setLayoutParams(btn7Parameters);
        //VALOR GAS METANO
        ViewGroup.LayoutParams btn8Parameters = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        valor_gas.setX(850);
        valor_gas.setY(200);
        //valor_gas.setPadding(40, 30, 15, 25);
```

```
        valor_gas.setText("0,1%");
        valor_gas.setTextColor(Color.BLACK);
        valor_gas.setBackgroundColor(Color.WHITE);
        valor_gas.setTextSize(15);
        valor_gas.setLayoutParams(btn8Parameters);
        //HUMEDAD
        ViewGroup.LayoutParams btn9Parameters = new
ViewGroup.LayoutParams(350, 150);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        humedad.setX(650);
        humedad.setY(250);
        humedad.setPadding(0, 30, 0, 0);
        humedad.setText("Humedad");
        humedad.setTextColor(Color.WHITE);
        humedad.setTextSize(15);
        humedad.setLayoutParams(btn9Parameters);
        //VALOR LUMINOSIDAD
        ViewGroup.LayoutParams btn10Parameters = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        valor_humedad.setX(700);
        valor_humedad.setY(350);
        //valor lum.setPadding(40, 30, 15, 25);
        valor_humedad.setText("30%");
        valor_humedad.setTextColor(Color.BLACK);
        valor_humedad.setBackgroundColor(Color.WHITE);
        valor_humedad.setTextSize(15);
        valor_humedad.setLayoutParams(btn10Parameters);
        //TITULO
        ViewGroup.LayoutParams btn11Parameters = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
//ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONT
ENT);
        titulo.setX(200);
        titulo.setY(10);
        //titulo.setPadding(0, 30, 0, 0);
        titulo.setText("CONTROL DEL HEXAPODO");
        titulo.setTextColor(Color.BLACK);
        titulo.setBackgroundColor(Color.WHITE);
        titulo.setTextSize(18);
        titulo.setLayoutParams(btn11Parameters);

        FrameLayout.LayoutParams llParameters = new
FrameLayout.LayoutParams(500,
500);//(LinearLayout.LayoutParams.MATCH_PARENT,LinearLayout.LayoutPara
ms.MATCH_PARENT);//(LinearLayout.LayoutParams.MATCH_PARENT,LinearLayou
t.LayoutParams.MATCH_PARENT);//(LinearLayout.LayoutParams.MATCH_PARENT
,LinearLayout.LayoutParams.MATCH_PARENT);
        ll.setBackgroundColor(Color.argb(80, 150, 0, 0));
        ll.setLayoutParams(llParameters);


        final WindowManager.LayoutParams parameters = new
WindowManager.LayoutParams(WindowManager.LayoutParams.MATCH_PARENT,750
, WindowManager.LayoutParams.TYPE_TOAST,
WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE,
PixelFormat.TRANSLUCENT);
```

```
        parameters.x = 0;
        parameters.y = 0;
        parameters.gravity = Gravity.CENTER_HORIZONTAL |
Gravity.BOTTOM;

        ll.addView(izquierda, btn3Parameters);
        ll.addView(adelante, btn1Parameters);
        ll.addView(atras, btn2Parameters);
        ll.addView(derecha, btn4Parameters);
        ll.addView(paro, btnParoParameters);
        ll.addView(temp, btn6Parameters);
        ll.addView(temperatura, btn5Parameters);
        ll.addView(gas, btn7Parameters);
        ll.addView(valor_gas, btn8Parameters);
        ll.addView(humedad, btn9Parameters);
        ll.addView(valor_humedad, btn10Parameters);
        ll.addView(titulo, btn11Parameters);
        ll.addView(stop, btnParameters);
        ll.addView(camara, titulo_cam);
        ll.addView(giro_izquierda, btn_giParameters);
        ll.addView(giro_derecha, btn_gdParameters);
        ll.addView(arriba, btn_arParameters);
        ll.addView(abajo, btn_abParameters);

        wm.addView(ll, parameters);

        ll.setOnTouchListener(new View.OnTouchListener() {
            private WindowManager.LayoutParams updateParameters =
parameters;
            int x, y;
            float touchedX, touchedY;

            @Override
            public boolean onTouch(View arg0, MotionEvent event) {
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        x = updateParameters.x;
                        y = updateParameters.y;
                        touchedX = event.getRawX();
                        touchedY = event.getRawY();
                    case MotionEvent.ACTION_MOVE:
                        updateParameters.x = (int) (x +
(event.getRawX() - touchedX));
                        updateParameters.y = (int) (y +
(event.getRawY() - touchedY));

                        wm.updateViewLayout(ll, updateParameters);

                        break;
                    default:
                        break;

                }
                return false;

            }
        });
        stop.setOnClickListener(new View.OnClickListener()

                                {
                                    @Override
```

```
                                    public void onClick (View v){
                                        wm.removeView(ll);
                                        stopSelf();
                                    }
                                }
        );


////////////////////////////////////////////////////////////////////////
/
        mHandler = new Handler(){
            public void handleMessage(android.os.Message msg){
                if(msg.what == MESSAGE_READ){
                    //String readMessage = null;
                    try {
                        readMessage = new String((byte[]) msg.obj,
"UTF-8");
                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }

temp.setText(readMessage);//mReadBuffer.setText(readMessage);
                }

            }
        };


        adelante.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                    mConnectedThread.write("1");

            }
        });
        atras.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("2");
            }
        });
        izquierda.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("3");
            }
        });
        derecha.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("4");

            }
        });
        paro.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                mConnectedThread.write("9");
            }
        });
```

```
        giro_izquierda.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("5");

            }
        });
        giro_derecha.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("6");

            }
        });
        arriba.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("7");

            }
        });
        abajo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mConnectedThread.write("8");

            }
        });



    }


}
```

8.5. Datasheet PIC16F877A

PICmicro MID-RANGE MCU FAMILY

4.1 Introduction

The high performance of the PICmicro™ devices can be attributed to a number of architectural features commonly found in RISC microprocessors. These include:

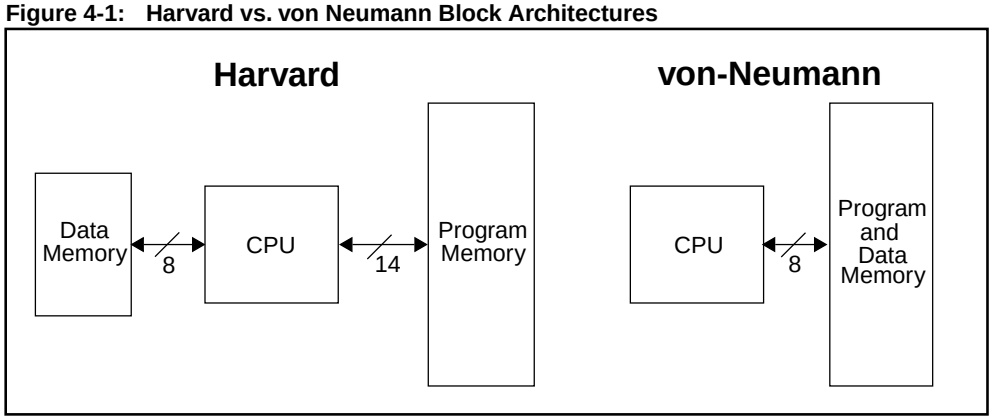
- Harvard architecture
- Long Word Instructions
- Single Word Instructions
- Single Cycle Instructions
- Instruction Pipelining
- Reduced Instruction Set
- Register File Architecture
- Orthogonal (Symmetric) Instructions

Figure 4-2 shows a simple core memory bus arrangement for Mid-Range MCU devices.

Harvard Architecture:

Harvard architecture has the program memory and data memory as separate memories and are accessed from separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. To execute an instruction, a von Neumann machine must make one or more (generally more) accesses across the 8-bit bus to fetch the instruction. Then data may need to be fetched, operated on, and possibly written. As can be seen from this description, that bus can be extremely conjested. While with a Harvard architecture, the instruction is fetched in a single instruction cycle (all 14-bits). While the program memory is being accessed, the data memory is on an independent bus and can be read and written. These separated buses allow one instruction to execute while the next instruction is fetched. A comparison of Harvard vs. von-Neumann architectures is shown in Figure 4-1.

Figure 4-1: Harvard vs. von Neumann Block Architectures



Long Word Instructions:

Long word instructions have a wider (more bits) instruction bus than the 8-bit Data Memory Bus. This is possible because the two buses are separate. This further allows instructions to be sized differently than the 8-bit wide data word which allows a more efficient use of the program memory, since the program memory width is optimized to the architectural requirements.

Single Word Instructions:

Single Word instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. With single word instructions, the number of words of program memory locations equals the number of instructions for the device. This means that all locations are valid instructions.

Typically in the von Neumann architecture, most instructions are multi-byte. In general, a device with 4-KBytes of program memory would allow approximately 2K of instructions. This 2:1 ratio is generalized and dependent on the application code. Since each instruction may take multiple bytes, there is no assurance that each location is a valid instruction.

Instruction Pipeline:

The instruction pipeline is a two-stage pipeline which overlaps the fetch and execution of instructions. The fetch of the instruction takes one TCY, while the execution takes another TCY. However, due to the overlap of the fetch of current instruction and execution of previous instruction, an instruction is fetched and another instruction is executed every single TCY.

Single Cycle Instructions:

With the Program Memory bus being 14-bits wide, the entire instruction is fetched in a single machine cycle (TCY). The instruction contains all the information required and is executed in a single cycle. There may be a one cycle delay in execution if the result of the instruction modified the contents of the Program Counter. This requires the pipeline to be flushed and a new instruction to be fetched.

Reduced Instruction Set:

When an instruction set is well designed and highly orthogonal (symmetric), fewer instructions are required to perform all needed tasks. With fewer instructions, the whole set can be more rapidly learned.

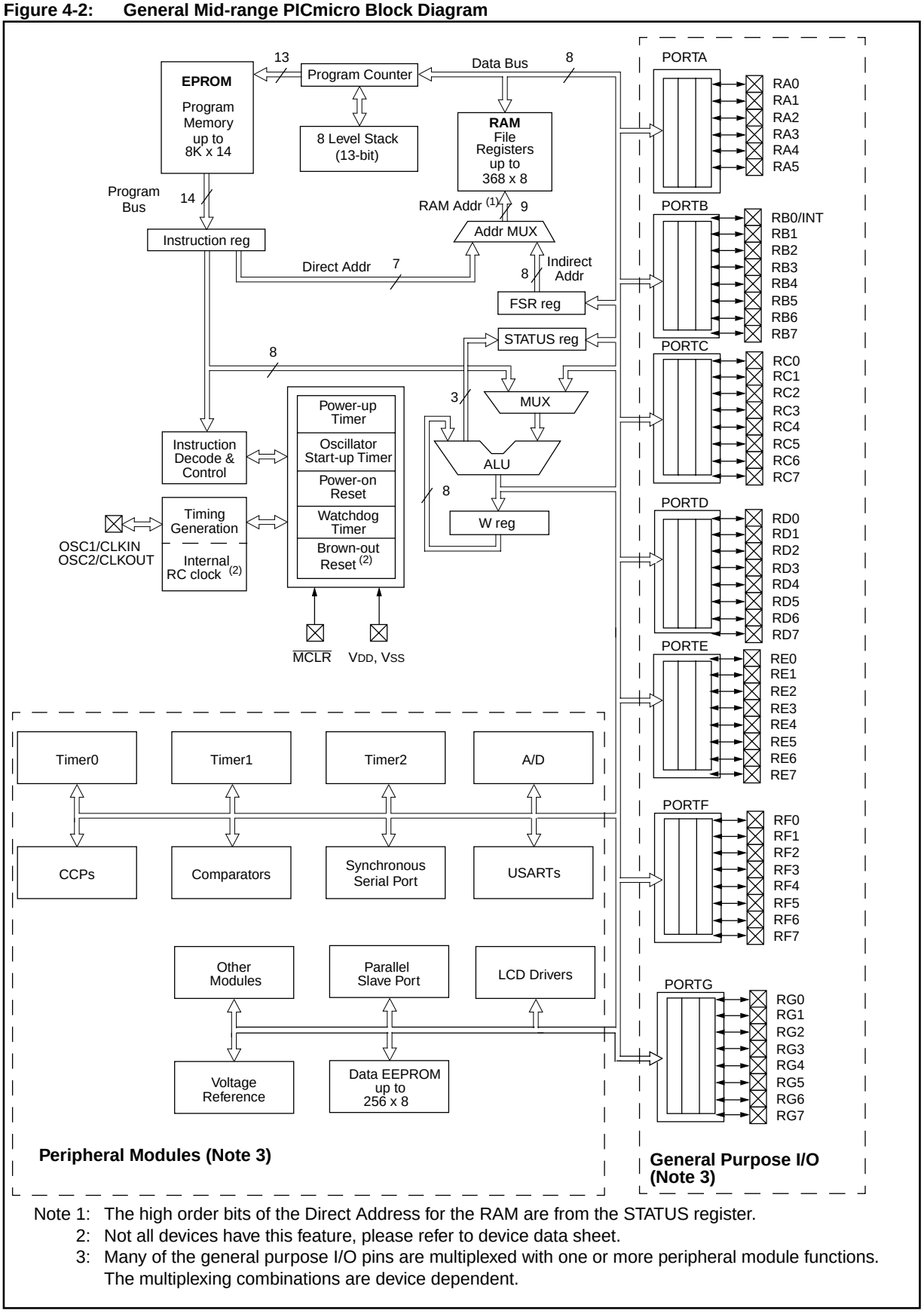
Register File Architecture:

The register files/data memory can be directly or indirectly addressed. All special function registers, including the program counter, are mapped in the data memory.

Orthogonal (Symmetric) Instructions:

Orthogonal instructions make it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of “special instructions” make programming simple yet efficient. In addition, the learning curve is reduced significantly. The mid-range instruction set uses only two non-register oriented instructions, which are used for two of the cores features. One is the SLEEP instruction which places the device into the lowest power use mode. The other is the CLRWDT instruction which verifies the chip is operating properly by preventing the on-chip Watchdog Timer (WDT) from overflowing and resetting the device.

Figure 4-2: General Mid-range PICmicro Block Diagram

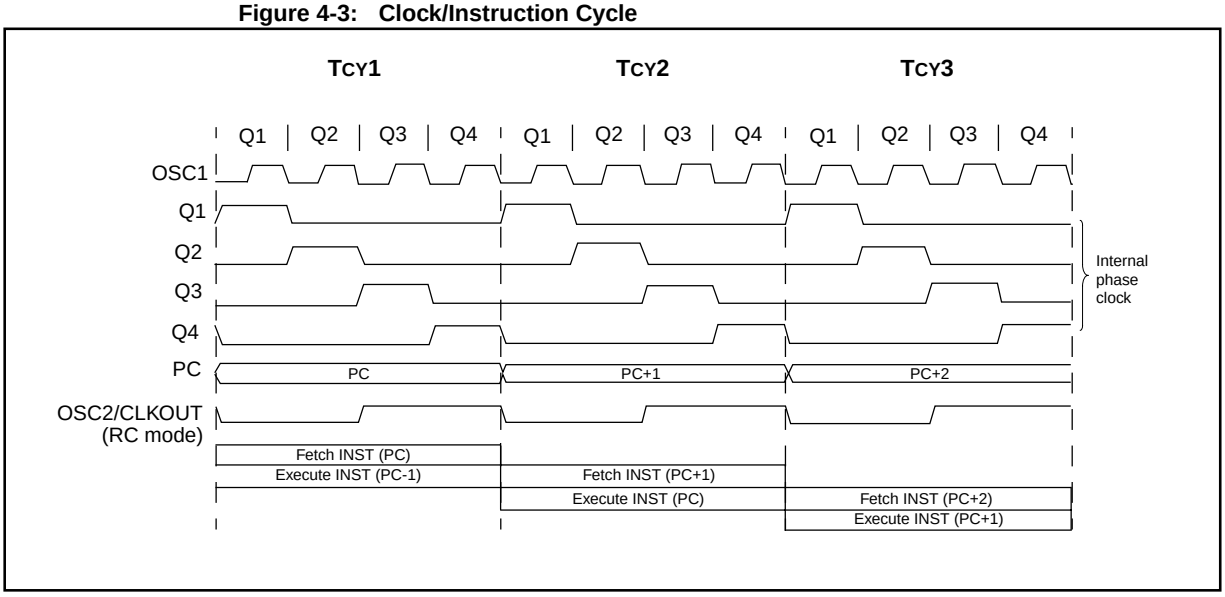


Note 1: The high order bits of the Direct Address for the RAM are from the STATUS register.
2: Not all devices have this feature, please refer to device data sheet.
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

4.2 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3, and Q4. Internally, the program counter (PC) is incremented every Q1, and the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are illustrated in Figure 4-3, and Example 4-1.

Figure 4-3: Clock/Instruction Cycle



4.3 Instruction Flow/Pipelining

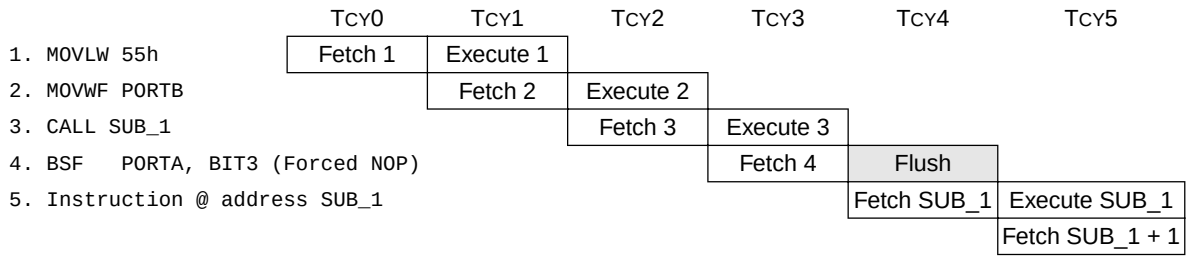
An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3, and Q4). Fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to Pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g. GOTO) then an extra cycle is required to complete the instruction (Example 4-1).

The instruction **fetch** begins with the program counter incrementing in Q1.

In the **execution** cycle, the fetched instruction is latched into the “Instruction Register (IR)” in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

Example 4-1 shows the operation of the two stage pipeline for the instruction sequence shown. At time TCY0, the first instruction is fetched from program memory. During TCY1, the first instruction executes while the second instruction is fetched. During TCY2, the second instruction executes while the third instruction is fetched. During TCY3, the fourth instruction is fetched while the third instruction (CALL SUB_1) is executed. When the third instruction completes execution, the CPU forces the address of instruction four onto the Stack and then changes the Program Counter (PC) to the address of SUB_1. This means that the instruction that was fetched during TCY3 needs to be “flushed” from the pipeline. During TCY4, instruction four is flushed (executed as a NOP) and the instruction at address SUB_1 is fetched. Finally during TCY5, instruction five is executed and the instruction at address SUB_1 + 1 is fetched.

Example 4-1: Instruction Pipeline Flow



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

4.4 I/O Descriptions

Table 4-1 gives a brief description of the functions that may be multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction (TRIS bit) of the port pin (such as in the A/D and LCD modules).

Table 4-1: I/O Descriptions

Pin Name	Pin Type	Buffer Type	Description
			Analog Input Channels
AN0	I	Analog	
AN1	I	Analog	
AN2	I	Analog	
AN3	I	Analog	
AN4	I	Analog	
AN5	I	Analog	
AN6	I	Analog	
AN7	I	Analog	
AN8	I	Analog	
AN9	I	Analog	
AN10	I	Analog	
AN11	I	Analog	
AN12	I	Analog	
AN13	I	Analog	
AN14	I	Analog	
AN15	I	Analog	
AVDD	P	P	Analog Power
AVSS	P	P	Analog Ground
C1	I	Analog	LCD Voltage Generation
C2	I	Analog	LCD Voltage Generation
CCP1	I/O	ST	Capture1 input/Compare1 output/PWM1 output
CCP2	I/O	ST	Capture2 input/Compare2 output/PWM2 output.
CDAC	O	Analog	A/D ramp current source output. Normally connected to external capacitor to generate a linear voltage ramp.
CK	I/O	ST	USART Synchronous Clock, always associated with TX pin function (See related TX, RX, DT)
CLKIN	I	ST/CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKIN, OSC2/CLKOUT pins)
CLKOUT	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. Always associated with OSC2 pin function. (See related OSC2, OSC1)
CMPA	O	—	Comparator A output
CMPB	O	—	Comparator B output

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
COM0	L	—	LCD Common Driver0
COM1	L	—	LCD Common Driver1
COM2	L	—	LCD Common Driver2
COM3	L	—	LCD Common Driver3
$\overline{CS}$	I	TTL	chip select control for parallel slave port (See related $\overline{RD}$ and $\overline{WR}$)
DT	I/O	ST	USART Synchronous Data. Always associated RX pin function. (See related RX, TX, CK)
			GP is a bi-directional I/O port. Some pins of port GP can be software programmed for internal weak pull-ups on the inputs.
GP0	I/O	TTL/ST	TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming mode.
GP1	I/O	TTL/ST	TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming mode.
GP2	I/O	ST	
GP3	I	TTL	
GP4	I/O	TTL	
GP5	I/O	TTL	
INT	I	ST	External Interrupt
$\overline{MCLR}$/VPP	I/P	ST	Master clear (reset) input or programming voltage input. This pin is an active low reset to the device.
NC	—	—	These pins should be left unconnected.
OSC1	I	ST/CMOS	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. CMOS otherwise.
OSC2	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
PBTN	I	ST	Input with weak pull-up resistor, can be used to generate an interrupt.
PSP0	I/O	TTL	Parallel Slave Port for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
PSP1	I/O	TTL	
PSP2	I/O	TTL	
PSP3	I/O	TTL	
PSP4	I/O	TTL	
PSP5	I/O	TTL	
PSP6	I/O	TTL	
PSP7	I/O	TTL	
			PORTA is a bi-directional I/O port.
RA0	I/O	TTL	
RA1	I/O	TTL	
RA2	I/O	TTL	
RA3	I/O	TTL	
RA4	I/O	ST	RA4 is an open drain when configured as output.
RA5	I/O	TTL	

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
			PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0	I/O	TTL	
RB1	I/O	TTL	
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	Interrupt on change pin.
RB5	I/O	TTL	Interrupt on change pin.
RB6	I/O	TTL/ST	Interrupt on change pin. Serial programming clock. TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming clock.
RB7	I/O	TTL/ST	Interrupt on change pin. Serial programming data. TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming data.
			PORTC is a bi-directional I/O port.
RC0	I/O	ST	
RC1	I/O	ST	
RC2	I/O	ST	
RC3	I/O	ST	
RC4	I/O	ST	
RC5	I/O	ST	
RC6	I/O	ST	
RC7	I/O	ST	
$\overline{RD}$	I	TTL	Read control for parallel slave port (See also $\overline{WR}$ and $\overline{CS}$ pins)
			PORTD is a bi-directional I/O port.
RD0	I/O	ST	
RD1	I/O	ST	
RD2	I/O	ST	
RD3	I/O	ST	
RD4	I/O	ST	
RD5	I/O	ST	
RD6	I/O	ST	
RD7	I/O	ST	
			PORTE is a bi-directional I/O port.
RE0	I/O	ST	
RE1	I/O	ST	
RE2	I/O	ST	
RE3	I/O	ST	
RE4	I/O	ST	
RE5	I/O	ST	
RE6	I/O	ST	
RE7	I/O	ST	

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to V_{DD}
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
REFA	O	CMOS	Programmable reference A output.
REFB	O	CMOS	Programmable reference B output.
			PORTF is a digital input or LCD Segment Driver Port
RF0	I/O	ST	
RF1	I/O	ST	
RF2	I/O	ST	
RF3	I/O	ST	
RF4	I/O	ST	
RF5	I/O	ST	
RF6	I/O	ST	
RF7	I/O	ST	
			PORTG is a digital input or LCD Segment Driver Port
RG0	I/O	ST	
RG1	I/O	ST	
RG2	I/O	ST	
RG3	I/O	ST	
RG4	I/O	ST	
RG5	I/O	ST	
RG6	I/O	ST	
RG7	I/O	ST	
RX	I	ST	USART Asynchronous Receive
SCL	I/O	ST	Synchronous serial clock input/output for I^2C mode.
SCLA	I/O	ST	Synchronous serial clock for I^2C interface.
SCLB	I/O	ST	Synchronous serial clock for I^2C interface.
SDA	I/O	ST	I^2C™ Data I/O
SDAA	I/O	ST	Synchronous serial data I/O for I^2C interface
SDAB	I/O	ST	Synchronous serial data I/O for I^2C interface
SCK	I/O	ST	Synchronous serial clock input/output for SPI mode.
SDI	I	ST	SPI Data In
SDO	O	—	SPI Data Out (SPI mode)
$\overline{SS}$	I	ST	SPI Slave Select input
SEG00 to SEG31	I/L	ST	LCD Segment Driver00 through Driver31.
SUM	O	AN	AN1 summing junction output. This pin can be connected to an external capacitor for averaging small duration pulses.
T0CKI	I	ST	Timer0 external clock input
T1CKI	I	ST	Timer1 external clock input
T1OSO	O	CMOS	Timer1 oscillator output
T1OSI	I	CMOS	Timer1 oscillator input
TX	O	—	USART Asynchronous Transmit (See related RX)

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to V_{DD}
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

I^2C is a trademark of Philips Corporation.

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
VLCD1	P	—	LCD Voltage
VLCD2	P	—	LCD Voltage
VLCD3	P	—	LCD Voltage
VLCDADJ	I	Analog	LCD Voltage Generation
VREF	I	Analog	Analog High Voltage Reference input. DR reference voltage output on devices with comparators.
VREF+	I	Analog	Analog High Voltage Reference input. Usually multiplexed onto an analog pin.
VREF-	I	Analog	Analog Low Voltage Reference input. Usually multiplexed onto an analog pin.
VREG	O	—	This pin is an output to control the gate of an external N-FET for voltage regulation.
VSS	P	—	Ground reference for logic and I/O pins.
VDD	P	—	Positive supply for logic and I/O pins.
$\overline{WR}$	I	TTL	Write control for parallel slave port (See $\overline{CS}$ and $\overline{RD}$ pins also).

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver