UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT HEXÁPODO TIPO RHEX PARA LA EXPLORACIÓN DE AMBIENTES INFLAMABLES CONTROLADO VÍA INALÁMBRICA

AUTORES:
ROMERO MACAS HEINERTH GUILLERMO
CUEVA CASTILLO JAHIRO PAÚL

DIRECTOR:
PÉREZ CHECA ANÍBAL ROBERTO

Quito, marzo de 2015

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además declaramos que los conceptos, análisis desarrollados y las conclusiones de este trabajo son de exclusiva responsabilidad de los autores.

_____

Heinerth Guillermo Romero Macas
C.I. 2100235288

_____

Jahiro Paúl Cueva Castillo
C.I. 2100273057

DEDICATORIA

Dedico este trabajo de titulación a toda mi familia, profesores, amigos y compañero de tesis por el apoyo brindado durante mi etapa de estudiante universitario y por contribuir de alguna manera en la culminación de este trabajo de titulación.

Heinerth Guillermo Romero Macas

Dedico este trabajo a mis padres, Hermógenes Cueva y Gladys Castillo, quienes supieron darme un ejemplo de esfuerzo y dedicación para superar con éxito los obstáculos que conlleva la culminación de una carrera de ingeniería y por su eterna paciencia durante el desarrollo del presente trabajo de titulación.

Jahiro Paúl Cueva Castillo

AGRADECIMIENTO

A mi profesor de sistemas microprocesados y tutor del presente trabajo de titulación el Ingeniero Roberto Pérez por su colaboración entusiasta en el diseño y desarrollo del robot Rhex el mismo que fue objeto de investigación de este proyecto.

Jahiro Paúl Cueva Castillo

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

En este proyecto se recurre a la investigación con fin de construir un robot como herramienta para la exploración de ambientes hostiles y proteger la integridad del ser humano. El robot será controlado mediante wireless de forma remota con la ayuda de un operador y este a su vez recibirá apoyo de algoritmos computacionales para el procesamiento de las imágenes adquiridas en búsqueda de seres humanos dentro del ambiente. Se buscará además, que su diseño mecánico sea capaz de sortear todo tipo de obstáculos para que su navegación sea fluida dentro del entorno. Llevará incorporada una cámara de video y un sensor de temperatura los mismos que recolectarán datos que serán enviados a un computador vía inalámbrica para su respectivo análisis e interpretación por parte del operador y el software especializado.

ABSTRACT

In this project it turned to research to use a robot as a tool for the exploration of hostile environments and protect the integrity of the human being. The robot will be controlled by wireless remote with the help of an operator, and this in turn will be supported by computer algorithms processing for acquired image in search of human beings within the environment. Also it seek that its mechanical design is capable of overcome all kinds of obstacles to its navigation is fluid within the environment. It shall be equipped with a video camera and a temperature sensor to collect the same data to be sent to a computer by wireless for examination and interpretation by the operator and specialized software.

INTRODUCCIÓN

El ser humano a lo largo de la historia se ha visto en la necesidad de optimizar recursos para así evitar el trabajo pesado y/o peligroso, desde un inicio buscó la manera de diseñar y construir pequeñas herramientas que faciliten las tareas complejas y que no pongan en peligro su integridad física.

Actualmente existen máquinas especializadas que realizan tareas específicas, dentro de estas máquinas están los robots. Existen varios tipos de robots que se clasifican en función a las tareas que estos vayan a realizar, pero de forma general se dividen en fijos y móviles.

Para el ser humano la percepción de temperatura es algo trivial ya que gracias a la experiencia o a conocimientos adquiridos está en la capacidad de discernir hasta qué punto puede permanecer en un determinado lugar sin sufrir daños, mientras que para un robot es solo uno de los problemas a resolver.

Gracias a los sentidos el ser humano puede distinguir colores, intensidades de luz, olores, texturas, sabores, y procesar gran cantidad de información que recibe del ambiente que lo rodea, dándole a estos estímulos un significado y un contexto. Por otro lado los robots desarrollan tareas específicas en un entorno previamente definido y si este entorno sufre alguna alteración, no prevista, el robot por sí solo no es capaz de sobrellevar su tarea. Para esto se puede proveer al autómata de ‘‘sentidos artificiales’’ como visión, detección o tacto, pero esto no es una solución inmediata ya que es necesario que el mismo lleve a cabo un procesamiento de todas estas variables las interprete y les dé un significado.

La información esperada de la operación de los robots móviles exploradores es el conocimiento del ambiente en donde estos se desplazan. A diferencia de los robots manipuladores o fijos, que no pueden desplazarse y llevan a cabo tareas repetitivas los robots móviles exploradores deben ser capaces de sortear cualquier eventualidad que se le presente durante su funcionamiento.

Para esto su sistema de locomoción es muy importante, existen mecanismos muy sencillos que pueden ser las convencionales ruedas hasta robots ápodos (sin patas), capaces de desplazarse por ductos de difícil acceso.

El sistema de locomoción escogido para este proyecto está formado por seis patas en semicírculo, es decir es un robot hexápodo similar al robot Rhex de Boston Dynamics. A continuación se enlista los principales problemas que tiene que resolver el prototipo motivo de este trabajo de grado:

- Libre desplazamiento por cualquier ambiente.
- Comunicación entre el robot móvil y la estación de mando.
- Envío de información de la cámara web hacia la estación de mando.
- Procesamiento de imágenes para la búsqueda de las personas en peligro.

Razón por la cual este proyecto de grado se enfocará en el diseño de un robot móvil explorador que sea capaz de cumplir todos los puntos antes mencionados. Sin embargo la mayoría de robots exploradores cuentan con sistemas avanzados como cámaras infrarrojas, sonares o incluso sistemas GPS que le permiten realizar su trabajo. La idea fundamental en este trabajo es implementar este sistema de monitoreo de temperatura y localización de personas de manera simple con base en sensores sencillos y una cámara comercial**.**

CAPÍTULO 1

1.1 Hipótesis

El diseño y construcción de un robot hexápodo tipo Rhex para la exploración de ambientes inflamables controlado vía inalámbrica va a permitir localizar personas que se encuentran atrapadas en ambientes hostiles gracias a un software especializado para la detección de personas.

1.2 Justificación

Al momento que una unidad de bomberos llega a un lugar siniestrado debe seguir una serie de pasos previo al rescate de una persona: buscar, localizar, ingresar, estabilizar y rescatar. Pero estos pasos deben llevarse a cabo aplicando todas las medidas de seguridad necesarias tanto para el personal de rescate como para la víctima, por medio de una estructura organizada, planes, protocolos y procedimientos de operación, sin embargo todos estos protocolos retrasan el acceso al área del desastre reduciendo el tiempo y la probabilidad de supervivencia de las víctimas que se encuentren atrapadas.

Durante el incendio se presentan varios escenarios, unos donde la accesibilidad es relativamente fácil y las victimas pueden ser rescatadas con mayor facilidad, pero los lugares de difícil acceso y potencial peligro van a ser los espacios donde nuestro prototipo se desenvolverá. Para esto nuestro proyecto propone la construcción de un robot móvil controlado via inalámbrica que nos permita facilitar la labor de búsqueda y recolección de datos de temperatura, todo esto para disminuir el tiempo de rescate y aumentar las probabilidades de supervivencia de una persona.

1.3 Objetivo general

Diseñar y construir un robot hexápodo para la exploración de ambientes inflamables controlado vía wireless enlazando un microcontrolador AVR con un computador,

para la transmisión y recepción de datos de: control, temperatura e imágenes en tiempo real, para la generación de un HMI mediante el uso de Matlab.

1.4 Objetivos específicos

- Diseñar y construir el hardware mecánico para un robot Rhex.
- Diseñar y construir las tarjetas electrónicas para el control del robot Rhex.
- Implementar un sistema de comunicación inalámbrico para comandar el robot de forma remota y recibir información de imagen y temperatura.
- Desarrollar el programa en lenguaje C para el computador de abordo para comandar al robot.
- Diseñar un interfaz gráfico con el software Matlab para el manejo y procesamiento de imágenes en búsqueda de personas en el ambiente hostil.
- Desarrollar un sistema de alarmas con la información de temperatura capturada por los sensores del robot.
- Ejecutar las pruebas de campo para la validación del buen funcionamiento del robot Rhex.

CAPÍTULO 2
ESTADO DEL ARTE

2.1 Descripción de un robot móvil

Un robot móvil es una máquina capaz de desplazarse sobre cualquier terreno al que sea asignado.

Existen varios tipos de sistemas de locomoción que se pueden usar a la hora de construir un robot móvil, esto dependerá de la superficie en donde el robot deba desenvolverse.

2.1.1 Robots con ruedas

Las ruedas son usadas para superficies lisas, donde no existan obstáculos considerables que entorpezca el desplazamiento del robot.

Robot con ruedas

Figura 1.
Fuente: Irene Robótica.

Este tipo de robots son los más comunes y fáciles de construir, una de las ventajas de este sistema con ruedas, es la carga que pueden transportar, ya que supera la capacidad a los otros sistemas de locomoción.

Se pueden adaptar sistemas de carros a control a remoto para la construcción de robots con ruedas, haciendo más fácil el armado del sistema mecánico.

La desventaja es que si encuentra algún tipo de irregularidad en el terreno que supere el diámetro de sus ruedas no podrá seguir su recorrido.

2.1.2 Robots con orugas

Si el terreno tiene algún tipo de irregularidad, es recomendable usar un sistema con orugas.

Robot con orugas

Figura 2.

Fuente: Irene Robótica.

Los robots con orugas tienen mayor superficie de contacto sobre el piso, lo que hace que el robot traccione con mayor facilidad sobre la misma permitiendo sortear terrenos con irregularidades considerables.

2.1.3 Robots con patas

Las patas son usadas para los robots que tienen que desplazarse sobre superficie muy irregulares.

Robot con patas

Figura 3.

Fuente: Irene Robótica.

Este tipo de robots están diseñados para sortear superficies irregulares con mayor facilidad, esto también implica que son más complejos de fabricar.

Según el número de patas se pueden denominar cuadrúpedos, hexápodos, octópodos, etc. Para este tipo de robots existe también una gran variedad de diseños de patas.

2.1.4 Robots ápodos

Este sistema de locomoción es usado para robots donde su campo de acción está limitado por el espacio físico.

Robot sin patas

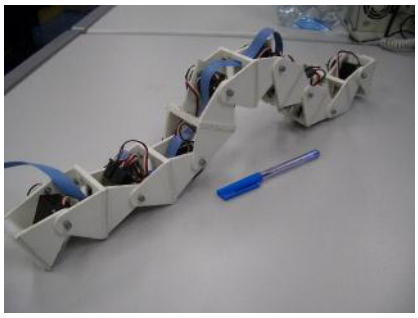


Figura 4.

Fuente: IEARobotics**.**

A diferencia de los otros robots, los robots sin patas no son tan ágiles ni veloces, pero su principal ventaja es que puede desplazarse por lugares muy pequeños ya que poseen una forma alargada. Gracias a su utilidad es que se investiga este tipo de locomoción.

2.2 Robots exploradores

Los robots exploradores son una herramienta importante a la hora de salvaguardar la integridad física de las personas en lugares peligrosos. Este tipo de robots permite disminuir el riesgo a los que se expone el ser humano en lugares como temblores, en zonas de altas temperaturas, humos tóxicos, y ambientes radiactivos.

2.2.1 Aplicaciones de los robots exploradores

La aplicación de este tipo de robots es muy variada, la inaccesibilidad del terreno y circunstancias que pongan en riesgo la vida de las personas son factores preponderantes que hacen que los robots exploradores sean de gran ayuda a la hora de recolectar información. A continuación se enlista algunos ejemplos de aplicaciones de los robots exploradores.

2.2.1.1 Robots exploradores espaciales.

Este tipo de robots son usados para expediciones espaciales, que facilitarán la recolección de datos de suelos, rocas, factores atmosféricos, etc. que se encuentran en el espacio.

Robot "Curiosity"

Figura 5.

Fuente: MuyInteresante.

2.2.1.2 Robots exploradores para rescate de personas.

Los robots exploradores de rescate son usados en lugares siniestrados, donde los escombros causados por el terremoto, explosión o incendio, no permite el fácil acceso del equipo de rescate.

Robot "Quince"

Figura 6.

Fuente: FayerWayer.

2.2.1.3 Robots exploradores de ductos.

Otro ejemplo de aplicación de los robots exploradores son los robots de inspección de ductos, este tipo de robots son ideales cuando se necesita explorar lugares de difícil acceso, son muy versátiles y pueden servir para inspeccionar ductos de aire acondicionado, tuberías de aguas, alcantarillas, etc. Son resistentes al agua y pueden fácilmente desplazarse sin temor a sufrir desperfectos por condiciones del ambiente.

Robot "iVision"

Figura 7.

Fuente: IBERobotics.

Este tipo de robots permiten la exploración de lugares con alto índice de humedad, desiertos, terrenos fangosos, laderas rocosas, etc. Por tal motivo los robots militares deben ser muy robustos y estar dispuestos a afrontar todo tipo de terreno.

Robot "Rhex1"

Figura 8.

Fuente: Boston Dynamics.

Estos son algunos ejemplos de las aplicaciones que se les puede dar a los robots exploradores, el presente trabajo de investigación se enfocará en el desarrollo de un robot tipo Rhex militar pero que será adaptado para que sirva como herramienta de rescate para personas que se encuentren atrapadas en lugares siniestrados por el fuego, dándole un plus adicional a este trabajo de grado, ya que contará con un sistema de análisis digital de imágenes para facilitar el trabajo de los rescatistas.

2.3 Qué es un robot explorador tipo Rhex?

Rhex es un prototipo creado por la empresa Boston Dynamics, el cual posee 6 actuadores, uno para cada pata que son controladas individualmente. Alcanza una velocidad promedio de 9,72 km/h, puede subir pendientes mayores a 45º e incluso subir gradas.

“Rhex2” - Boston Dynamics

Figura 9.

Fuente: Boston Dynamics.

El proyecto Rhex fue financiado por la Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA) de Estados Unidos, que se encarga del desarrollo de nuevas tecnologías para uso militar. Y desembolsó 5000.000,00 de dólares para terminar el proyecto.

Robot “Rhex2”

Figura 10.

Fuente: Robotpark.

Este robot hexápodo y su peculiar diseño de locomoción hace que pueda atravesar superficies irregulares con mayor facilidad haciéndolo un robot todo terreno. Posee dos baterías de alta duración que le permite estar en funcionamiento por un tiempo

estimado de 6 horas, viene integrado con cámaras de 320x240 píxeles de resolución, y puede soportar temperaturas que oscilen entre -15 y 45 ºC.

2.4 Partes que conforman el prototipo

Luego de explicar que son los robots móviles y de conocer las aplicaciones de los mismos, se empezará a analizar todos los componentes necesarios para la construcción del prototipo que es tema de estudio en este proyecto de grado.

2.4.1 Estructura

El chasis es el esqueleto del robot, tiene como fin soportar el peso de todos los demás componentes que conforman el mismo, los materiales usados para la construcción son: hierro, aluminio y fibra de vidrio.

El diseño de Rhex está hecho de tal manera que sea capaz de soportar el peso de las patas, baterías, tarjetas electrónicas de control, y su propio peso.

La parte interior del chasis está protegida con aislante térmico para que pueda desplazarse por ambientes inflamables.

2.4.2 Aislante térmico

Todos los elementos electrónicos que estén dentro del prototipo deben estar bien protegidos del calor por lo que se ha usado “termo-espuma” como aislante térmico.

La termo-espuma es una mezcla de polietileno y aluminio, posee una delgada lámina de aluminio adherida a la espuma de polietileno. Estos dos componentes son muy buenos alisadores térmicos, la función del aluminio es disipar y rechazar el calor que venga del exterior y la espuma se encarga de mantener estable la temperatura interna del robot.

Termo-Espuma

Figura 11.

Fuente: Tecnolibertador.

2.4.3 Tarjeta de control

La tarjeta de control es el conjunto de dispositivos electrónicos que se encarga de la toma de decisiones, o en otras palabras, es el cerebro del robot. Su tarea es controlar todos los periféricos, manejan entradas, salidas, comunicaciones con otros dispositivos, etc.

Esta tarjeta es su parte más esencial, está formada por microcontroladores, que son dispositivos programables, que junto con las herramientas de programación permiten cumplir tareas predeterminadas dependiendo de la aplicación para la que esté diseñada.

2.4.3.1 Conversor DC-DC.

Los conversores DC-DC son dispositivos que transforman corriente continua de una tensión a otra. Son reguladores de conmutación, teniendo a su salida una tensión regulada y, casi siempre con limitación de corriente. Se utiliza frecuencias de conmutación cada vez más altas esto permite reducir la capacidad de los capacitores, para disminuir el volumen, peso y precio de los módulos. (Wikipedia, 2014)

Hay de varios tipos. Pero se los puede clasificar en tres grupos: los que aumentan la tensión a su salida (conversor elevador) los que disminuyen la tensión a su salida (conversor reductor), y los que son capaces de realizar ambas funciones.

El conversor usado en la tarjeta de control de este proyecto es un módulo comercial, como el que se muestra en la Figura 12.

Convertidor DC-DC, step down

Figura 12.

Fuente: Aliexpress.

A diferencia de los reguladores convencionales LM7805, estos conversores pueden proveer de hasta 3 amperios a la tarjeta de control de manera muy eficiente y emitiendo una menor cantidad de calor, y consecuente disipando menos potencia.

2.4.3.2 Microntroladores AVR de Atmel.

Los AVR son MCU's con arquitectura RISC del fabricante norteamericano Atmel. Actualmente cuenta con muchos seguidores alrededor del mundo debido a su simple diseño y la facilidad de programación.

Los AVR's usados en este proyecto son de la familia ATmega, estos son microcontroladores con memoria flash programable de hasta 256kB, su encapsulado puede ser de 28 hasta 100 pines, tienen un conjunto amplio de instrucciones y un conjunto amplio de periféricos.

El AVR es una unidad de procesamiento lógico (CPU) tipo Harvard, donde los conjuntos de datos y los conjuntos de instrucciones están conceptualmente separados.

2.4.3.3 Transceptor.

Para lograr la comunicación entre el microcontrolador maestro y los microcontroladores esclavos, se ha armado una red RS485, para esto se tuvo que utilizar transceptores MAX 485, que son una manera fácil y eficiente de hacer una red de comunicación sin tener pérdida de información.

Transceptor MAX485

Figura 13.
Fuente: ElectronicaMagnabit.

Estos transceptores permiten conectar hasta 32 microcontroladores en una red con una comunicación semidúplex, es decir que pueden enviarse señales en ambos sentidos, pero no simultáneamente.

2.4.3.4 Oscilador externo.

Los microcontroladores Atmel tienen incorporado un oscilador interno que puede ser configurado por software, pero para este proyecto se necesita trabajar a una frecuencia que realice la comunicación serial entre maestro y esclavos sin tener margen de error, para esto se escogió un cristal externo de 11.059200Mhz que es un valor comercial hecho específicamente para trabajar con comunicación serial.

Si se divide por ejemplo un cristal de 8Mhz para 9600 (valor estándar de transmisión) esto dará 833,333 un valor decimal, el 0,333 será el porcentaje de error que se tendrá si se usa un cristal de 8Mhz, por otro lado si se divide 11059200 para 9600 se tiene 1152 que es un valor entero, es por esto que se ha escogido este valor para el oscilador externo, ya que es divisible para todas las velocidades estándar de comunicación.

Con este oscilador se puede cambiar la velocidad de transmisión de datos sin tener que preocuparse por cambiar el oscilador del microcontrolador maestro y de los esclavos.

2.4.3.5 Smith Trigger.

El Smith Trigger es un circuito integrado que usa la histéresis para eliminar el ruido de una señal, en este proyecto se ha utilizado para estabilizar la señal de los encoders ópticos de los motores, tiene el mismo funcionamiento que una compuerta lógica NOT (SN7404), pero con la característica de que mantiene un valor en alto ó 1 lógico (5V) si el nivel de voltaje de la entrada no cambia lo suficiente como para ser considerada como el estado lógico opuesto es decir 0 lógico (0V).

Smith Trigger SN7414

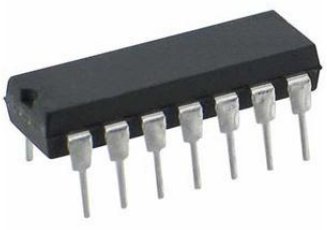


Figura 14.

Fuente: Tu-proyecto.

2.4.3.6 X-bee pro.

Los X-bee son módulos comerciales de comunicación inalámbrica, que trabajan a una frecuencia de 2,4Ghz, hay varios modelos de acuerdo a la necesidad del cliente, para este proyecto en particular se usa el modelo Xbee Pro Serie1 de compañía Digi International, que tiene un alcance máximo de 1Milla (1.6Km).

Módulos X-bee Pro, Digi International

Figura 15.

Fuente: Digi International.

Los módulos Xbee Pro se comunican entre sí mediante el protocolo de comunicación inalámbrico Zigbee, que está basado en el estándar IEEE_802.15.4, que fue creado por Zigbee Alliance.

Principales características de los módulos Xbee:

- Bajo Costo
- Ultra bajo consumo de potencia (<10uA)
- Uso de bandas de radio libres y sin necesidad de licencias
- Instalación barata y simple
- Redes flexibles y extensibles
- Interfaz serial.
- 65,000 direcciones para cada uno de los 16 canales disponibles. Se pueden tener muchos de estos dispositivos en una misma red. (Oyarce, 2010)

2.4.3.7 Adaptador Xbee-USB.

Para conectar el módulo Xbee al computador se adquirió un adaptador serial Xbee-TTL de la compañía Ideas y Tecnología.

Módulo adaptador Xbee-USB

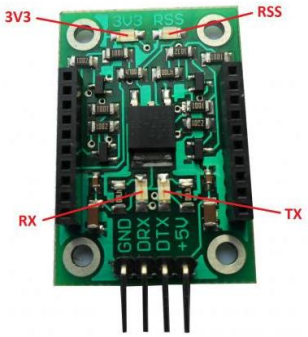


Figura 16.

Fuente: IdeasTechnology.

El adaptador XBEE - TTL I&T es un módulo de comunicación serial que sirve para ajustar los niveles de voltajes de cualquier módulo XBEE a niveles TTL.

Especificaciones:

- Led indicador de encendido (3V3), RSS, TX y RX
- Bus de 4 pines para alimentación y comunicaciones serial.
- Transferencia de datos 300 Baud hasta 3MBaud
- Soporta 7 a 8 bits datos, 1 o 2 bits stop, y odd/even/mark/space/no parity

Aplicaciones:

- Interface de comunicación serial con el computador y módulo XBEE.
- Adquisición y envío de datos hacia el computador. (IdeasyTecnología, 2014)

2.4.4 Tarjeta de potencia

La tarjeta de potencia es la encargada de manejar altas corrientes, a diferencia de la tarjeta del control o cerebro del robot que solo maneja valores en mili amperios.

Estas dos tarjetas deben estar interactuando durante el proceso de control, por lo que son diseñadas de tal manera que no generen ruido ni perturbaciones al sistema de control.

La tarjeta de potencia en un sistema robótico es la encargada del control de los actuadores, ya sean estos servos, motores a pasos, motores de corriente continua entre otros, que tengan un alto consumo de corriente.

Para este caso se encargan del control de velocidad y giro de los motores de corriente continua del robot, todos los elementos de la tarjeta de control deben tener características especiales para soportar las tensiones requeridas por los motores. Las tarjetas de control del robot "RHEX", soportan un máximo de voltaje de 30 voltios y 50 amperios de corriente pico.

2.4.4.1 Driver de motores 50A de dos canales.

Las tarjetas de potencia para el control de motores usada en este proyecto son: "50A Dual-channel Motor Drive module-Arduino Compatible" de la compañía Elechouse.

Driver de Motores ElecHouse

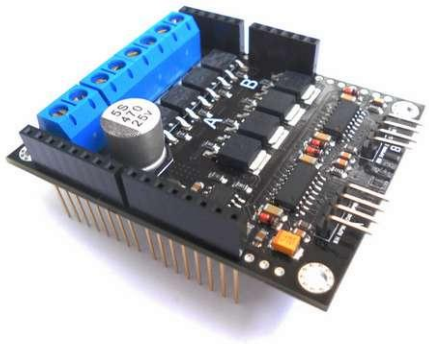


Figura 17.
Fuente: ElecHouse.

Este controlador de motores trabaja con una carga máxima de hasta 50 amperios por canal. Tiene mejor rendimiento que algunos controladores de motores comerciales

como por ejemplo MC33886 ó L298, especialmente en términos de control de velocidad del motor y eficiencia energética.

Posee una función de freno, la cual permite detener de una manera rápida el motor, es muy fácil de manejar, y contiene un chip controlador de puente completo y MOSFET's de baja resistencia interna.

Características:

- Pico de corriente (carga): 50 A
- Corriente máxima de trabajo recomendada: 20A
- Voltaje de alimentación (carga): 0V-30V
- Voltaje de alimentación recomendado: 12V-26V
- Voltaje de control: 4V-12V
- Volatje de control TTL: 2.5V-12V

(Elechouse, 2014)

2.4.4.2 Modulación por ancho de pulso (PWM).

Para variar la velocidad de los motores y tener control sobre las patas se usará la modulación por ancho de pulso. El principio es sencillo lo que se hace es variar la relación de tiempo en el que un pin permanece encendido versus el tiempo en el que se apaga. Si el período de la señal es constante, el hecho de que la señal esté mucho o poco tiempo encendida hace que el voltaje promedio que se refleje sea variable. (Burgath-Guatemala-S.A., 2008)

PWM, Modulación por ancho de Pulso

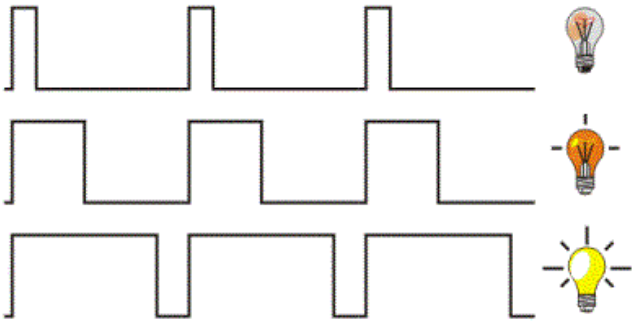


Figura 18.

Fuente: Burgath.

2.4.5 Sensores

Los sensores son los sentidos del robot, así como las personas tienen 5 sentidos para interactuar con los estímulos del ambiente, el robot también tiene incorporado sensores capaces de transformar estos estímulos en pulsos eléctricos o cambios de voltaje para que la tarjeta de control pueda interpretarlos y tome las acciones necesarias para desenvolverse con su entorno. El robot usa sensores de temperatura para determinar hasta qué punto puede permanecer en un determinado lugar, y un sensor de imagen que serán los ojos del robot.

2.4.5.1 Sensores de temperatura.

Los sensores de temperatura son dispositivos que transforman la variación de temperatura en cambios se señales eléctricas, para luego ser procesadas por las tarjetas de control.

En este proyecto se usa el sensor de temperatura LM35. Este sensor es de alta precisión, abarca un rango de -55ºC a 150ºC, su salida es lineal y tiene una equivalencia de 10mV por cada grado centígrado.

Encapsulado sensor lm35

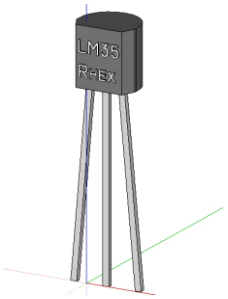


Figura 19.

Elaborado por: Heinerth Romero Macas

Principales características:

- Voltaje de alimentación: 4v a 30v
- Consumo: <60uA
- Precisión: 0,5ºC a +25ºC
- Se calibra en grados centígrados
- Factor de escala lineal +10mV/ºC
- Baja impedancia de salida
- Apropiado para aplicaciones remotas
- Bajo costo

2.4.5.2 Sensor de imagen.

A medida que la luz atraviesa un objetivo, ésta se enfoca en el sensor de imagen de la cámara. Un sensor de imagen está compuesto de muchos fotositos y cada fotosito corresponde a un elemento de la imagen, comúnmente conocido como "píxel". Cada píxel de un sensor de imagen registra la cantidad de luz a la que se expone y la convierte en un número de electrones correspondiente. Cuanto más brillante es la luz, más electrones se generan.

Cuando se fabrica una cámara, existen dos tecnologías principales que pueden utilizarse para el sensor de imagen:

- CCD (dispositivo de almacenamiento de carga)
- CMOS (semiconductor de óxido metálico complementario)
- Sensores megapíxel

Sensor de imagen CMOS

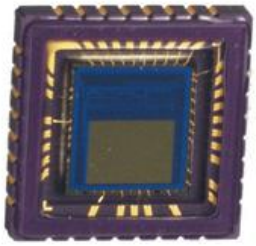


Figura 20.

Fuente: Axis.

Sólo se hablará sobre el sensor CMOS, ya que la cámara que se ha escogido para este proyecto viene incorporado un sensor CMOS.

Sensores CMOS: Los recientes avances en los sensores CMOS los están acercando a sus homólogos CCD en términos de calidad de la imagen. Los sensores CMOS reducen el costo total de las cámaras ya que contienen todas las funciones lógicas necesarias para fabricar cámaras para ellos. En comparación con los sensores CCD, los sensores CMOS permiten mayores posibilidades de integración y más funciones. Los sensores CMOS también tienen un tiempo menor de lectura (lo que resulta una ventaja cuando se requieren imágenes de alta resolución), una disipación de energía menor a nivel del chip, así como un tamaño del sistema menor. Los sensores CMOS megapíxel están mucho más extendidos y son menos caros que los sensores CCD. (AxisCommunicationsAB, 2013)

Cámara web inalámbrica

Figura 21.

Fuente: Sukasa.

La cámara web que se ha usado para este proyecto es de la marca Apex, tiene un alcance máximo de 100 metros, y funciona a 2,4Ghz vienen con su receptor USB para conectar directamente hacia un computador y poder adquirir la imagen.

2.4.5.3 Sensor infrarrojo.

Los sensores infrarrojos poseen un diodo emisor y un forotransistor, el emisor se encarga de emitir luz infrarroja y el receptor es un fototransistor que entra en saturación cuando influye sobre él una cantidad de luz infrarroja considerable.

Sensor QRD1114

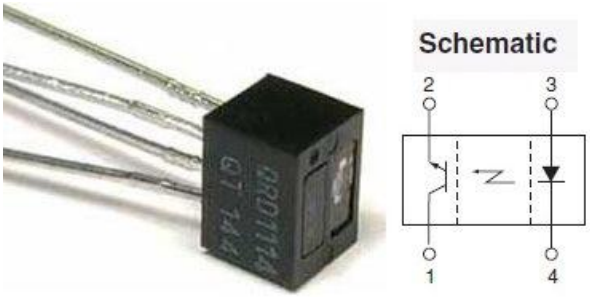


Figura 22.

Fuente: ElectronicayRobotica.wordpress.

Se han usado sensores infrarrojos para construir los encoders de los motores, ya que es de suma importancia saber la posición de cada pata, a continuación se explica que es un encoder óptico.

2.4.5.4 Encoder.

Los encoders son una herramienta muy usada en proyectos de robótica y sirven para transformar el movimiento angular en pulsos electrónicos, dependiendo del movimiento que se vaya a medir pueden ser lineales y rotatorios. Para este proyecto se ha usado sensores fotoreflectivos para construir los encoders, y fueron adaptados a los motores para medir y transformar el ángulo de desplazamiento de cada pata.

Se construyó un encoder óptico incremental, para lo cual se necesitó un disco dividido en zonas opacas (negro) y zonas reflectivas (aluminio).

Disco para encoder de 100 divisiones, hecho en Autocad 2011.

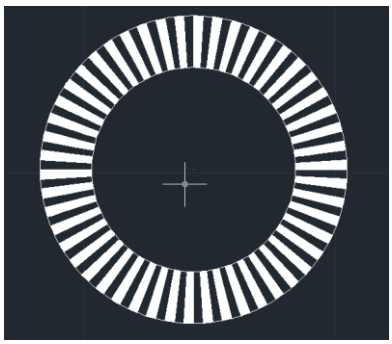


Figura 23.

Elaborado por: Heinerth Romero Macas

El sensor lo que hace es leer los cambios de color de blanco al negro o viceversa cada cambio de color genera un pulso eléctrico lo que permite contar los pulsos que la pata se ha desplazado.

2.4.6 Baterías

Las baterías son una parte muy importante dentro de los componentes que permiten el buen funcionamiento del robot, en base a la experiencia adquirida en proyectos anteriores se eligió usar Baterías de Polímero de Litio o más conocidas como Baterías de LIPO a continuación se da una breve explicación de sus características.

Batería usada para alimentación de motores

Figura 24.

Fuente: Hobbyking.

Las Baterías Lipo vienen en celdas que proporcionan un voltaje de 3,7 voltios, y por lo general son conectadas en serie para elevar su voltaje y/o paralelo para incrementar su corriente. Este tipo de baterías vienen en su empaquetado con ciertos parámetros que se deben tener en cuenta como por ejemplo:

- S: Número de celdas
- C: Capacidad

Por ejemplo si una batería específica es de 500mA, 3S y 30C quiere decir que tiene 3 celdas en serie que proporcionará 11.1 voltios (celdas=3 x 3,7v), y la C o capacidad es la corriente máxima que puede soportar sin sufrir daños, para este ejemplo la capacidad máxima que puede soportar es 15A (corriente=0.5 x C=30).

En comparación con otro tipo de baterías, son muy ligeras pero con las mismas características tanto en voltaje como en corriente, la única desventaja es que son delicadas y tienen un costo muy alto, pero vale la pena adquirirlas. (Soto, 2013)

2.5 Comunicación serial

El concepto de la comunicación serial es muy sencillo, no es más que el envío de datos uno detrás de otro, “en serie”, es decir es el envío de un bit a la vez, Si se hace una comparación con la comunicación paralela que permite enviar un byte completo por vez, la comunicación serial es más lenta, pero es mucho más sencilla y puede alcanzar mayores distancias.

2.5.1 Tipos de comunicación serial

Existen dos tipos de comunicaciones seriales la Síncrona y Asíncrona

Síncrona, necesita dos líneas, una por la cual se transmitirá los datos y la otra para transmitir los pulsos de reloj que indicará cuando un dato es válido.

Ejemplos de comunicación síncrona son los protocolos I2C y SPI.
Asíncrona, no necesita pulsos de reloj, la duración de la transmisión de datos se determina por la velocidad a la cual se va a transmitir los datos.

2.5.1.1 Comunicación RS-232.

El puerto serial del computador es conocido como RS-232, su principal ventaja es que todas las computadoras poseen al menos un puerto serial, lo que permite que se pueda comunicar con otra computadora, la impresora, el mouse y para el caso del desarrollo de este trabajo de grado con los microcontroladores.

2.5.1.2 Comunicación RS-422.

RS-422 es el conector serial utilizado en las computadoras Apple de Macintosh. RS-422 usa señales eléctricas diferenciales, en comparación con señales referenciadas a tierra como en RS-232. La transmisión diferencial, que utiliza dos líneas para transmitir y recibir, tiene la ventaja que es más inmune al ruido y puede lograr

mayores distancias que RS-232. La inmunidad al ruido y la distancia son dos puntos clave para ambientes y aplicaciones industriales.

2.5.1.3 Comunicación RS-485.

RS-485 es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a un computador u otro controlador para la recolección de datos, HMI, u otras operaciones. El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 1219 metros de cable.

Existen dos opciones para montar una red RS-485:

- Full-duplex (4 cables)
- Half-duplex (2 cables)

2.5.1.4 Comunicación RS-485 a cuatro hilos (full-duplex).

Para hacer una comunicación a 4 hilos es necesario un transceptor para protocolo RS-485, que no es más que un chip capaz de manejar el envío y recepción de datos con los niveles de voltaje requeridos para el protocolo RS-485.

Transceptor RS-485

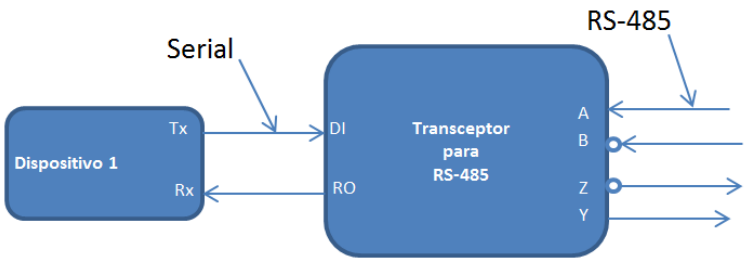


Figura 25.

Elaborado por: Heinerth Romero Macas

Es muy sencillo, solo se debe conectar el transceptor RS-485 a los terminales Tx y Rx de un dispositivo previsto de comunicación serial.

Hardware para comunicación RS-485 a 4 hilos

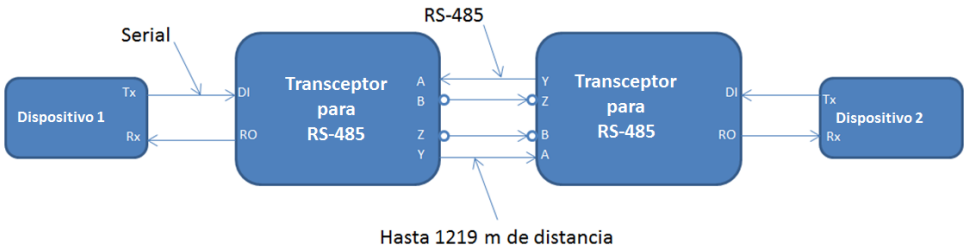


Figura 26.

Elaborado por: Heinerth Romero Macas

El emisor y receptor son independientes, esto permite que un instante de tiempo el emisor y el receptor puedan transmitir y recibir información simultáneamente, haciendo que la comunicación sea mucho más rápida.

2.5.1.5 Comunicación RS-485 a dos hilos (half-duplex).

Al igual que la comunicación de 4 hilos, también se necesita de transceptores RS-485 pero para dos hilos, y la diferencia es que el receptor y el emisor no pueden transmitir datos al mismo tiempo, o sólo recibe o solo transmite pero nunca al mismo tiempo.

Hardware para comunicación RS-485 a 2 hilos

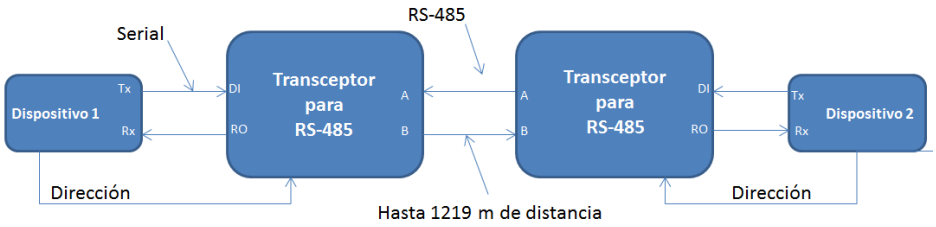


Figura 27.

Fuente: Elaborado por Heinerth Romero Macas

Una de las ventajas de la comunicación half-duplex es que se tiene únicamente dos cables y si se desea armar una red donde los dispositivos se encuentren a largas distancias esta es la solución.

2.5.1.6 Ventajas de usar comunicación RS-485.

- Soporta grandes distancias entre dispositivos, hasta más un kilómetro.
- Se necesita únicamente dos cables para su conexión, usando dos cables se puede conectar varios dispositivos, lo que permite ahorrar dinero en cableado.
- Los transceptores usados en el proyecto, son fáciles de encontrar en el mercado, y no cuestan más de 5,00 usd. cada uno.
- Una red de comunicación RS-485, es escalable lo que permite conectar más dispositivos a nuestra red si en el futuro necesitamos hacerlo, actualmente ya existen chips que permiten conectar más de 32 dispositivos.
- Es inmune al ruido, esto permite que sea un sistema confiable, ya que no existirá perdida de datos, durante la comunicación.

2.5.1.7 Arreglo maestro-esclavo.

Generalmente cuando se conectan varios dispositivos en una red, hay uno al que se le denomina Maestro, y otros a los que se les llama Esclavos. El maestro será el encargado de enviar comandos a cada uno de los esclavos, para que estos realicen un trabajo específico.

Arreglo maestro-esclavo

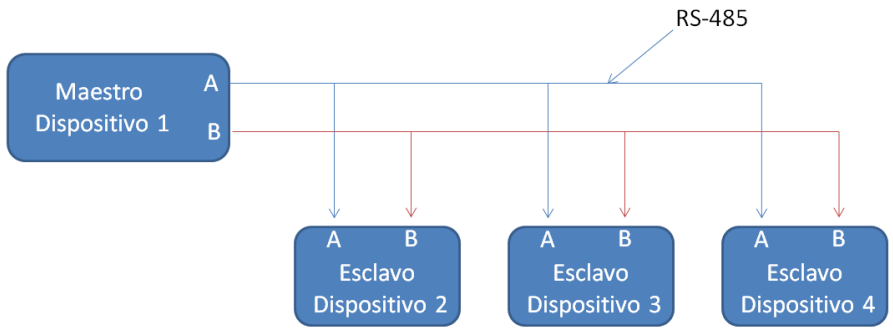


Figura 28.

Elaborado por: Heinerth Romero Macas

La imagen muestra la conexión RS-485 a dos hilos, con tres dispositivos esclavos y un maestro, se omite los transceptores para simplificar el gráfico.

Se ha escogido un arreglo maestro esclavo porque el sistema tiene siete tarjetas de control, una tarjeta principal (master) y seis tarjetas pequeñas (esclavos), donde cada una realiza una tarea en específico para la operación del robot, razón por la que se necesita dar jerarquía, al sistema de control.

La tarjeta principal al tener un microcontrolador de mayor memoria será el maestro y el encargado de dar las ordenes a cada una de las placas pequeñas que serán los esclavos. Todo esto para que el sistema de control tenga un orden y facilite el trabajo del maestro para que pueda ocuparse de otras funciones.

Cada esclavo realizarán su trabajo cada vez que el maestro se lo solicite, lo que facilita en cierta forma el trabajo del maestro, al mismo tiempo permite un control total de todo el sistema, ya que el maestro estará en constate comunicación con los esclavos para verificar que cada uno de ellos cumplan con la labor designada.

2.6 Comunicación inalámbrica

En la comunicación inalámbrica la conexión entre emisor y receptor no se encuentra unida mediante un cable, sino que se usa la modulación de ondas electromagnéticas

para la transmisión y recepción de datos, esto permite intercambiar información sin tener que preocuparse de los cables.

2.6.1 Protocolo de comunicación Zigbee

El protocolo Zigbee fue diseñado por la Zigbee Alliance, el cual es un conjunto de protocolos de alto nivel para comunicaciones inalámbricas, basada en el estándar IEEE 82.15.4. Para usos industriales, médicos y científicos utiliza bandas reservadas internacionalmente, en concreto: 868 Mhz en Europa, 915 Mhz en Estados Unidos y 2.4Ghz en todo el mundo, en este proyecto se usará la banda libre de 2.4Ghz. A continuación se enlista sus principales ventajas:

- Reduce tiempos de espera en el envío y recepción de datos
- Opera en la banda libre de ISM de 2.4Ghz para comunicación inalámbrica
- Bajo ciclo de trabajo, por lo que garantiza larga duración a las baterías
- Alojamiento de 16 bits a 64 bits de dirección extendida
- Ideal para conexiones punto a punto y punto a multipunto
- Diseñado para el direccionamiento de información y la actualización de la red

Otro aspecto muy importante es la seguridad de las transmisiones y de los datos, los cuales son puntos clave en la tecnología ZigBee que utiliza el modelo de seguridad de la subcapa MAC IEEE 802.15.4, la cual especifica 4 servicios de seguridad:

- Control de accesos, el dispositivo mantiene una lista de dispositivos "comprobados" en la red.
- Datos encriptados, las cuales utilizan una encriptación con un código de 128 bits.
- Integración de tramas, para proteger los datos de ser modificados por otros.
- Secuencias de refresco, para comprobar que las tramas no han sido reemplazadas por otras. El controlador de red comprueba estas tramas de refresco y su valor, para ver si son las esperadas. (Wikispaces, 2015)

CAPÍTULO 3

DESARROLLO E IMPLEMENTACIÓN DEL SOFTWARE Y HARDWARE

3.1 Hardware mecánico

En este capítulo se hablará como fueron realizados los procesos de diseño y construcción del prototipo, luego de buscar información acerca de los robots exploradores se observó que el modelo Rhex satisface las expectativas para lograr el propósito de este proyecto de investigación.

Se empezará por la parte mecánica que es muy importante para el buen funcionamiento del robot. Las medidas fueron tomadas del modelo del Robot X-RHex de la Universidad de Pensilvania. (Pensilvania, 2010)

Para el diseño de las partes mecánicas se usó la herramienta Google Sketchup 8.0 el cual es un software libre para diseño en 3D, este es fácil de utilizar y muy interesante porque permite hacer animaciones de los movimientos del robot.

3.1.1 Chasis

El chasis es la parte mecánica que no se ve, si se hace una analogía con el cuerpo humano el chasis viene a ser el esqueleto. Es la parte que soporta todo el peso del prototipo, por lo que el chasis debe ser de un material resistente pero liviano. Si se construye un robot muy pesado se tendrá que buscar motores de mayor potencia y subirá el costo de construcción, lo que se busca es ahorrar recursos y hacer un robot que cumpla las expectativas del proyecto con la menor cantidad de recursos.

3.1.1.1 Diseño del chasis.

Antes de empezar con el diseño del chasis se tuvo que modelar el motor que se va a usar en el prototipo. Los motores escogidos son motores DC de 12Volts usados como

elevadores de vidrio en automóviles, a continuación se muestra su modelado en 3D hecho en Google Sketchup 8.0.

Diseño del motor DC

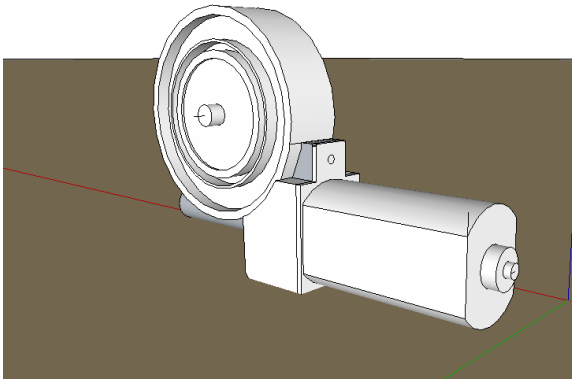


Figura 29.

Elaborado por: Heinerth Romero Macas

Se empezó por modelar el motor ya que es necesario conocer el espacio exacto que va a ocupar cada elemento dentro del robot.

Una vez que se tiene el diseño del motor se procede con el diseño del chasis.

Diseño de chasis

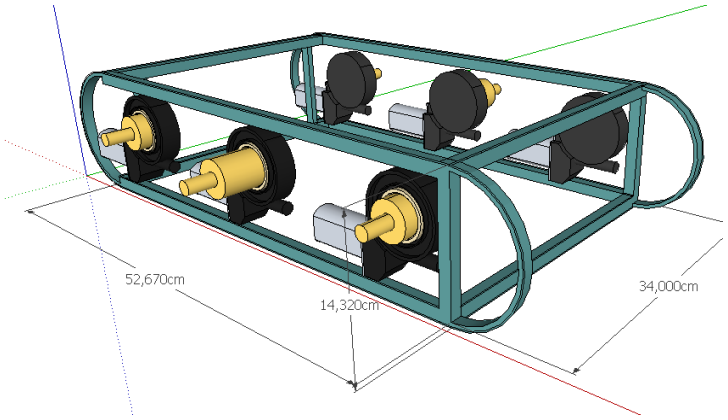


Figura 30.

Elaborado por: Heinerth Romero Macas

El chasis fue diseñado de tal manera que quepan todos los motores y demás componentes, el tamaño de los motores dificultó un poco el diseño del chasis, ya que

se buscaba que quede lo más parecido a los modelos de los prototipos investigados previamente. En la figura 30 se muestra el primer diseño y distribución de los 6 motores que se usaron en el robot.

3.1.1.2 Construcción del chasis.

Luego de tener el diseño en 3D se procedió a la construcción del mismo. Todo el chasis fue construido con tubo cuadrado de media pulgada, alineado a escuadra y soldado perfectamente.

Construcción del chasis

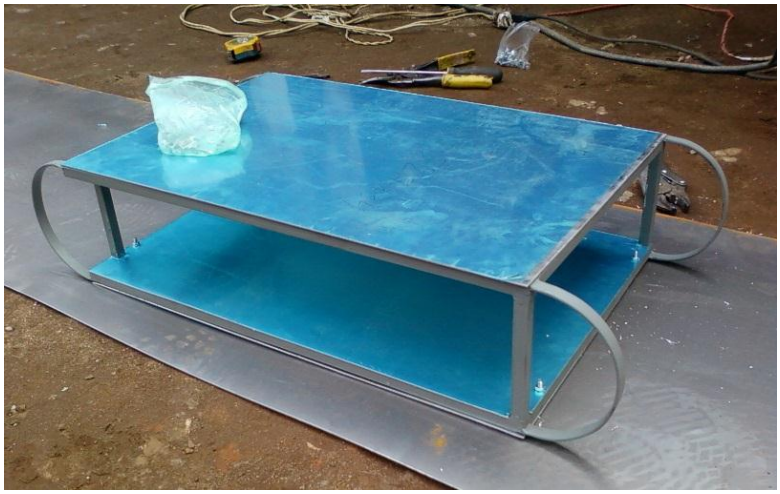


Figura 31.

Elaborado por: Heinerth Romero Macas

En la figura 31 se muestra el chasis construido y las tapas superior e inferior, las tapas son planchas de aluminio de 1mm de espesor.

Luego de tener listo el chasis y las tapas superior e inferior se procedió a construir las tapas laterales, para esto se hizo orificios redondos lo suficientemente grandes para que pueden salir los ejes de los motores.

Tapas laterales con agujeros

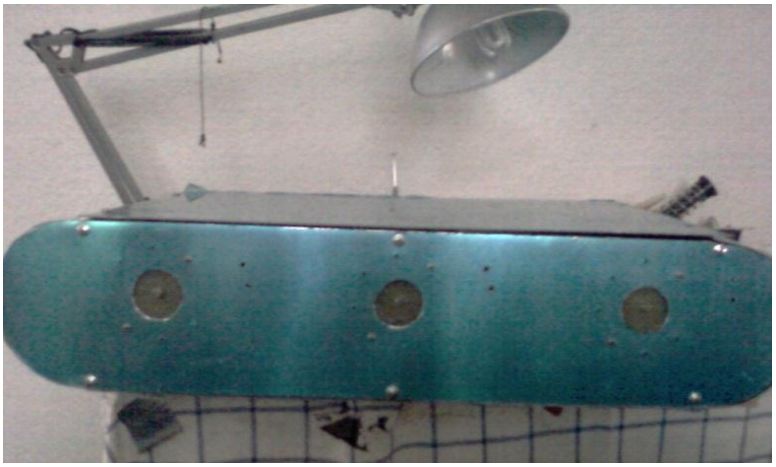


Figura 32.

Elaborado por: Heinerth Romero Macas

Por último se construyó la tapa frontal y posterior, el primer diseño fue redondo, pero luego de ciertas pruebas se decidió que tanto la tapa frontal como posterior debían ser planas para que permita que las patas sobresalgan de la estructura y tengan mayor contacto con superficie irregulares.

Estructura final

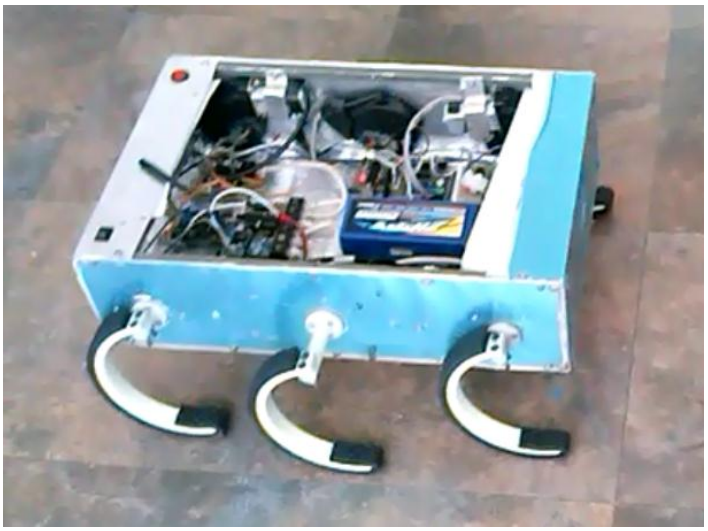


Figura 33.

Elaborado por: Heinerth Romero Macas

En la figura 33 se muestra la estructura final. A continuación se hablará sobre el diseño y construcción de las patas del robot.

3.1.2 Patas

Las patas son el sistema de locomoción del prototipo, por lo que se debe tener en cuenta algunos aspectos que permitan que se desplace de manera fluida en el ambiente donde tenga que desplazarse.

3.1.2.1 Diseño de las patas.

El mejor diseño para este tipo de robots hexápodos, son las patas en semicírculo, deben tener el mayor diámetro posible. Para este prototipo en particular tendrán un diámetro de veinte centímetros como máximo, el diámetro de la pata determinará la altura de los obstáculos que puede sobrepasar.

Antes de empezar con el diseño de las patas se tuvo que diseñar seis ejes que permitan acoplar el eje del motor con las patas, dos de estos ejes deben ser más grandes, esto es porque las patas de la mitad deben estar salidas en comparación con las patas delanteras y traseras del robot.

Ejes para acoplar las patas con el motor

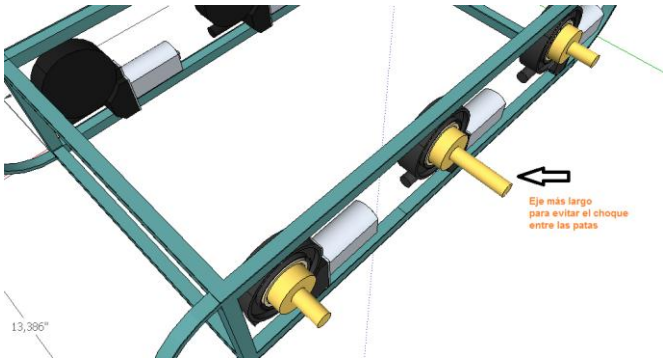


Figura 34.

Elaborado por: Heinerth Romero Macas

Las patas irán ajustadas con dos pernos hexagonales con muesca para evitar que se aflojen con el movimiento. Los ejes son los de color amarillo que se muestran en la figura 34, como se mencionó antes, los ejes de la mitad tienen tres centímetros más de largo para evitar que las patas choquen entre sí.

Ejes para acople

Figura 35.

Elaborado por: Heinerth Romero Macas

Para el diseño de las patas se consideró la adherencia, la resistencia al calor, flexibilidad, peso y coste. Todos estos factores hicieron que se escoja al Duralón como material para la construcción de las mismas, el cual es un material más liviano que el aluminio, y resistente a altas temperaturas, es flexible y no se rompe fácilmente, este material se usa para hacer piñones, bocines en remplazo del acero en aplicaciones industriales.

Material de las patas

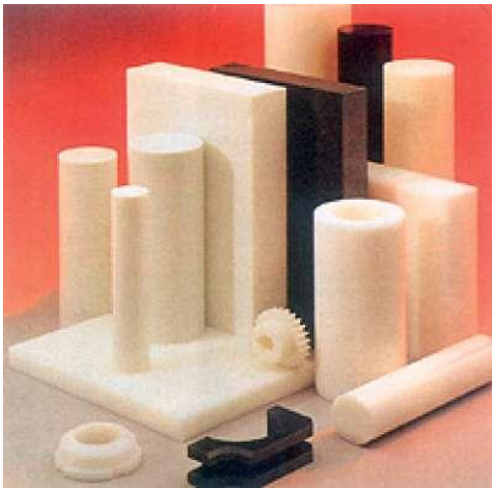


Figura 36.

Fuente: Aceros Industriales HGB.

Y el prototipo que se construirá será como el diseño mostrado en la figura 37:

Diseño de patas en 3D

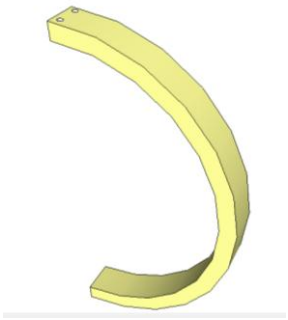


Figura 37.

Elaborado por: Heinerth Romero Macas

Las medidas de las patas son: 20 cm de diámetro, 1cm de espesor, y dos de ancho, en la figura falta el caucho que se puso para que tenga mayor tracción, eso se explicará en la en el apartado 3.1.2.2.

3.1.2.2 Construcción de las patas.

Para la construcción de las patas se tuvo que comprar un pedazo redondo de 22 cm de Duralón, todas las patas fueron maquinadas en torno, con las medidas exactas para no tener inconvenientes a la hora del armado.

Pero como el Duralón es liso se recubrió las patas con caucho para que puedan tener mayor fricción sobre el piso, el caucho usado fue el que se ocupa para la fabricación de suelas de zapatos.

Patas con recubrimiento de caucho

Figura 38.

Elaborado por: Heinerth Romero Macas

En la figura 38 se muestra las patas terminadas. A continuación se hablará del aislante térmico.

3.1.3 Aislante térmico

Por las altas temperaturas que debe soportar el prototipo, se buscó un sistema que mantenga frío su interior, para que no cause daños en las tarjetas electrónicas y demás componentes.

Toda la estructura del robot fue cubierta con termo-espuma, para lograr así que el robot se desplace por ambientes inflamables sin sufrir daños al menos durante su labor de rescate.

Instalando la termo-espuma

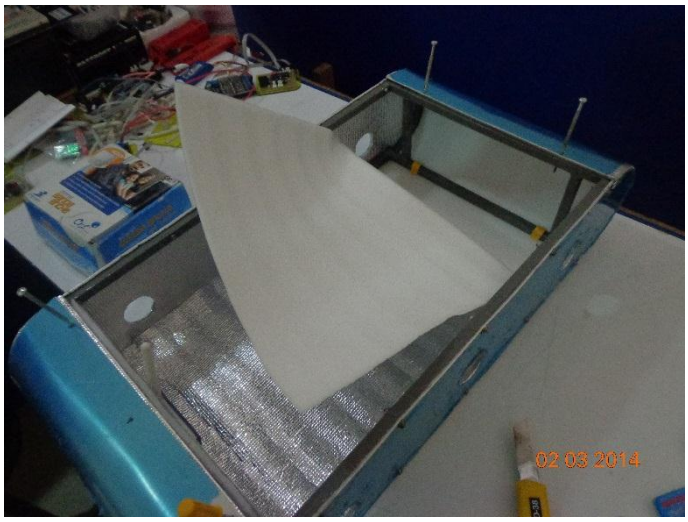


Figura 39.
Elaborado por: Heinerth Romero Macas

El diseño y construcción se realizó en base a prueba y error, de acuerdo a los problemas que iban surgiendo se iba corrigiendo. Como estudiantes de ingeniería electrónica se dificultó el diseño y modelamiento de las partes mecánicas, sin embargo se investigó a fondo la forma, tamaño y materiales que se utilizaron para la construcción del prototipo, es por esto que solo se da una breve explicación del diseño y construcción del hardware mecánico.

3.2 Hardware electrónico

El hardware electrónico es en lo que se enfocará este proyecto de titulación, por lo que este apartado se extenderá un poco para explicar cómo se diseñó y construyó las placas electrónicas del robot.

3.2.1 Tarjeta de control master

La tarjeta de control master es muy importante ya que será la encargada de sincronizar todo el trabajo de los esclavos, deberá estar en constate comunicación con cada uno de ellos, intercambiando datos y verificando el buen funcionamiento del sistema de control.

También será la encargada de procesar información de los sensores de temperatura y enviar de manera inalámbrica esta información hacia la estación de control, en este caso a una computadora portátil.

3.2.1.1 Diseño de la tarjeta de control master.

Para el diseño de las tarjetas de control se usó Proteus, el cual es una herramienta de simulación y diseño de tarjetas muy potente, tiene una gran cantidad de librerías que permite simular cualquier tipo de circuito electrónico.

Para empezar con el diseño de la tarjeta se debe determinar qué microcontrolador se va a usar y tener claro cuántos pines se va a necesitar, cuantas entradas, cuantas salidas, cuantos puertos de comunicación, si tiene o no entradas análogas, es decir dimensionar el microcontrolador.

En la Tabla 1 se muestra el número de pines que debe tener el microcontrolador de la tarjeta master.

Tabla 1.

Requerimientos básicos del microcontrolador master

MASTER:	
Entradas	**Detalle**
1	ADC, Sensor de temperatura 1
2	ADC, Sensor de temperatura 2
3	Sensor pos. Inicial 1
4	Sensor pos. Inicial 2
5	Sensor pos. Inicial 3
6	Sensor pos. Inicial 4
7	Sensor pos. Inicial 5
8	Sensor pos. Inicial 6
9	Reset
Salidas	**Detalle**
1	Ventilador 1
2	Ventilador 2
3	Ventilador 3
4	Ventilador 4
5	Led indicador
Comunicación	**Detalle**
1	Rx1 Comunicación pc
2	Tx1 Comunicación pc
3	Rx2 Comunicación esclavos
4	Tx2 Comunicación esclavos
Reloj	**Detalle**
1	Crystal 1
2	Crystal 2
Alimentación	**Detalle**
1	Vcc
2	Gnd
3	Aref
4	Avcc
TOTAL PINES	**24**

Nota. Elaborado por: Heinerth Romero Macas

Tomando en cuenta el número de pines necesarios se escogió un microcontrolador Atmega164p, de Atmel de cuarenta pines.

Distribución de pines Atmega164p

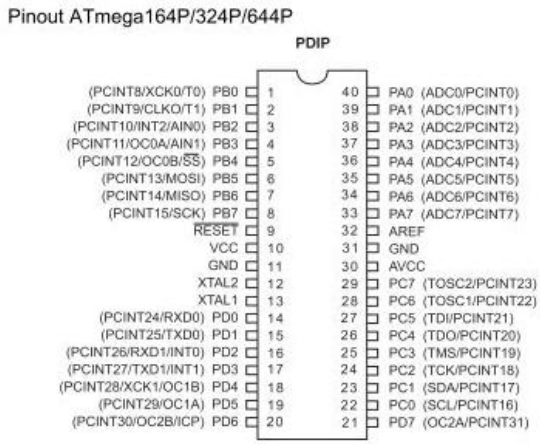


Figura 40.

Fuente: Hoja de datos Atmega164P.

Estas son sus principales características:

Tabla 2.

Características Micro-controlador Master Atmega164p

MASTER:	
Memoria de código	16Kbyte de memoria flash
Memoria de datos	512 bytes de EEPROM
Terminales para entrada/salida	32
Frecuencia máxima de trabajo	20 Mhz
Reloj de tiempo real	Sí
Temporizadores	2 de 8bits y 1 de 16 bits
Canales PWM	4
Canales ADC	8 de 10 bits
USART	2
Interfaz SPI	Sí
Oscilador interno programable	Sí
Watchdog timer	Sí

Nota. (Corporation, 2010) **Elaborado por:** Heinerth Romero Macas

Luego de haber escogido el microcontrolador se procede a hacer la simulación en ISIS de Proteus. Para esto se empezará por la alimentación del microcontrolador y en sí de todo el sistema, para esto se ha usado una batería de litio-polímero o polímero de litio recargable de 11.1V a 1A, conectado a un conversor de voltaje dc-dc step down, el cual proporcionará los 5 voltios que alimentarán a todos los integrados del circuito.

Conexión del conversor DC-DC

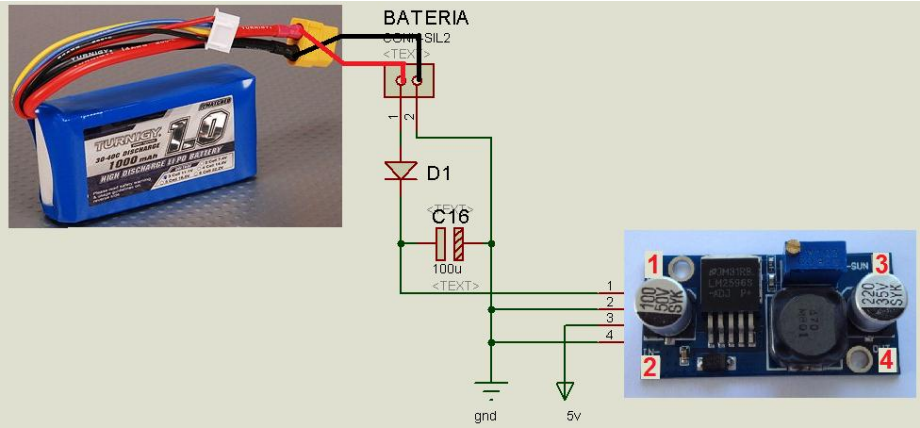


Figura 41.

Elaborado por: Heinerth Romero Macas

El conversor viene con 4 pines de conexión que están representados en la figura 41:

- 1: Entrada de la batería, que pasa por un diodo de protección de 3A, en caso de conectar al revés los terminales de la batería.
- 2: GND
- 3: Salida regulada a 5 voltios
- 4: GND

*Se conectó un capacitor de 100uf para mantener un voltaje estable en la entrada del conversor, en caso de que haya una sobretensión para proteger la batería.

La tarjeta master tiene que comunicarse con seis tarjetas esclavos, a continuación se explicará cómo se implementó la comunicación de estas tarjetas.

Para lograr que se comunique la tarjeta master con los esclavos, se prefirió implementar una comunicación serial ya que al inicio del proyecto se pensó en implementar la comunicación maestro-esclavo de forma paralela, pero por baja eficiencia y la gran cantidad de cables, se descartó esa idea y se prefirió realizar la comunicación con el protocolo RS485 half-duplex de dos hilos multipunto, esto quiere decir que tanto el emisor como el receptor pueden enviar y recibir datos pero uno a la vez, reduciendo la cantidad de cables y coste del proyecto.

Por lo que se conectará al USART1 del microcontrolador, pines 16 y 17, RX, y TX respectivamente, al transceptor MAX485. El USART0 será usado para la comunicación inalámbrica.

Conexión transceptor MAX485

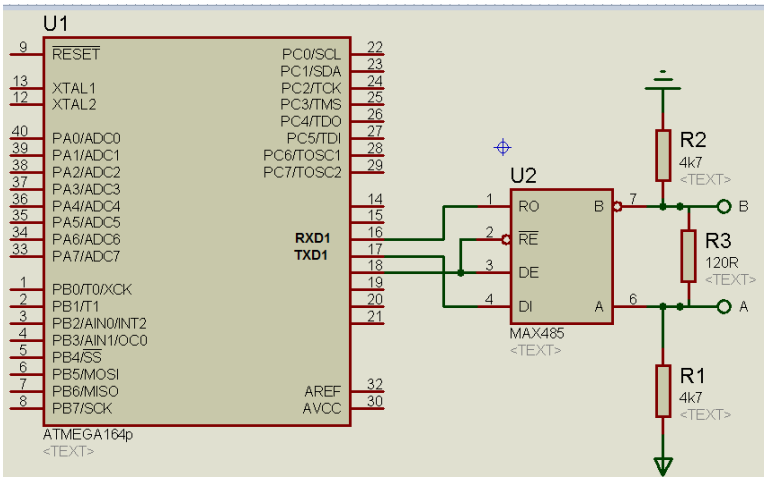


Figura 42.

Elaborado por: Heinerth Romero Macas

Con esto ya se tiene la salida para conectar la tarjeta master con cada uno de los esclavos, en cada una de las tarjetas esclavos habrá un circuito similar con dos entradas A y B, para realizar la comunicación, el pin 18 es el bit de control que está conectado al pin 4 del MAX485, este bit de control será el encargado de poner al master en modo escucha o modo envío, cuando el estado del bit de control esté en bajo el microcontrolador estará esperando un dato que venga de los esclavos pero si se pone en alto se pondrá en modo envío.

El USART0 se usará para realizar la comunicación inalámbrica y para esto solo se necesita conectar el TX0 y RX0 al módulo Zigbee, a continuación su conexión:

Conexión con el Xbee

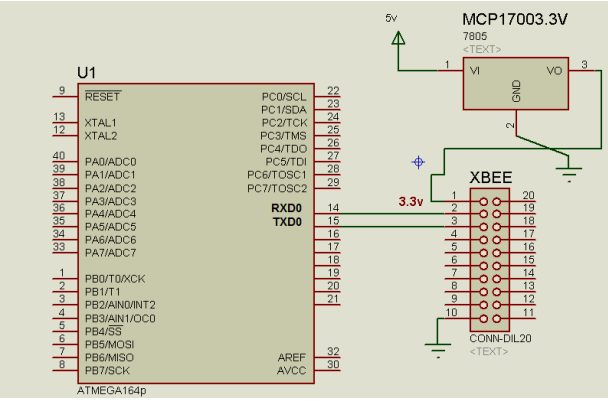


Figura 43.

Elaborado por: Heinerth Romero Macas

El módulo de comunicación inalámbrica Xbee Pro que se usará tiene 20 pines de conexión y para este proyecto únicamente se ocupará 4, dos son de alimentación y dos de comunicación, en la figura 43 se muestra como es su conexión con el microcontrolador.

- 1: VCC, se usa un regulador de voltaje de 3.3v
- 2: TX, se conecta al RX0 del microcontrolador
- 3: RX, se conecta al TX0 del microcontrolador
- 10: GND

Con esto ya se tiene diseñado la parte de comunicaciones tanto con el computador así como con los esclavos, solo faltan las entradas y salidas.

Como entradas tenemos dos sensores de temperatura y seis sensores infrarrojos de los encoders de posición inicial, esto es algo sencillo que se explica a continuación.

Conexión sensores

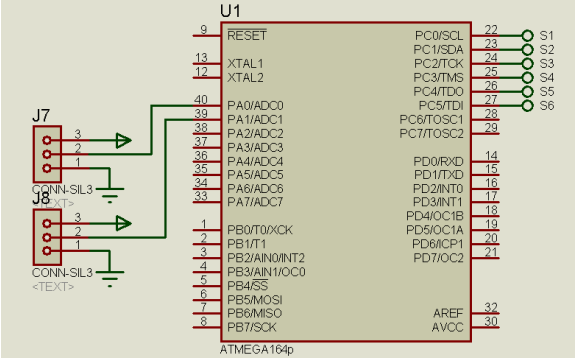


Figura 44.

Elaborado por: Heinerth Romero Macas

Los sensores de temperatura usados son los LM35, que proporcionan un rango de salida de -55°C a +150°C con un factor de escala lineal de +10mV/°C, por lo que se necesita conectarlos a las entradas ADC del microcontrolador. Los otros seis sensores son digitales por lo que se ha conectado al puerto C digitales, adicionalmente los sensores de temperatura necesitan de alimentación, por lo que se ha puesto conectores de tres pines, para VCC, GND y señal.

Las salidas que manejará la tarjeta de control master son para el encendido de un led indicador y para el encendido de cuatro ventiladores, para esto se usarán 5 pines del microcontrolador.

El led que se ha puesto es para realizar pruebas al momento de programar, ya que es una manera fácil de seguir los procesos y el estado del programa, Adicionalmente se dejó 4 salidas opcionales para ventiladores que se los conectará en caso de que el sistema de aislamiento térmico no mantenga estable la temperatura interna del robot, estos ventiladores funcionan a 12 voltios, por lo que se utilizó un driver para motores para el accionamiento de cada uno de ellos, su conexión se muestra en la figura 45.

Conexión de salidas

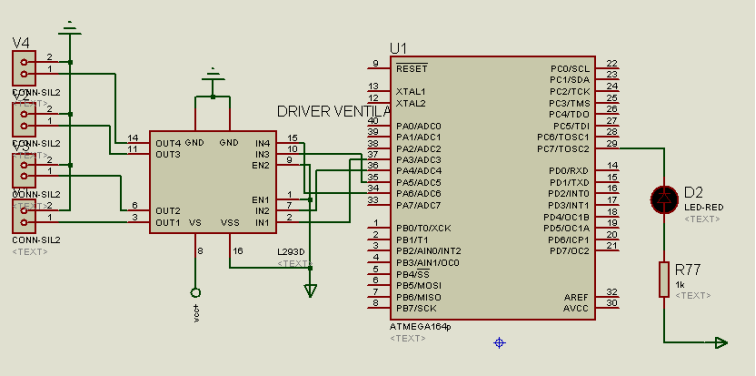


Figura 45.

Elaborado por: Heinerth Romero Macas

Por último pero no menos importante tenemos la conexión del cristal de cuarzo y el reset.

Conexión cristal y reset

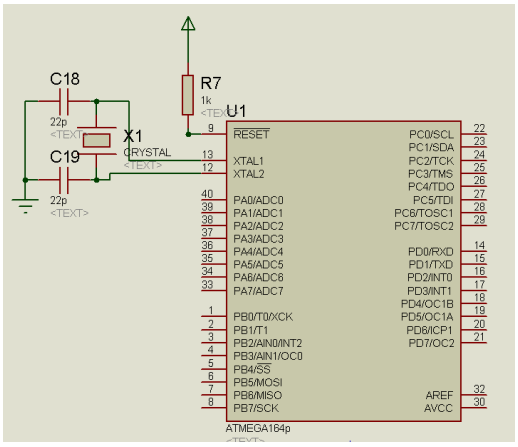


Figura 46.

Elaborado por: Heinerth Romero Macas

Al tener que trabajar con comunicación serial se debe utilizar un cristal externo que permita el intercambio de datos sin pérdida de información por esto se utilizó un cristal de cuarzo externo de 11.059200 Mhz de frecuencia, que es un valor estándar para este tipo de aplicaciones.

Al pin 9 del microcontrolador se ha conectado una resistencia pull-up para evitar que el microcontrlador se resetee por ruido.

Luego de haber explicado la conexión y funcionamiento de cada una de los componentes electrónicos que se conectarán en la tarjeta de control master se muestra como se verán todas las partes interconectadas.

Diagrama final de la tarjeta de control master

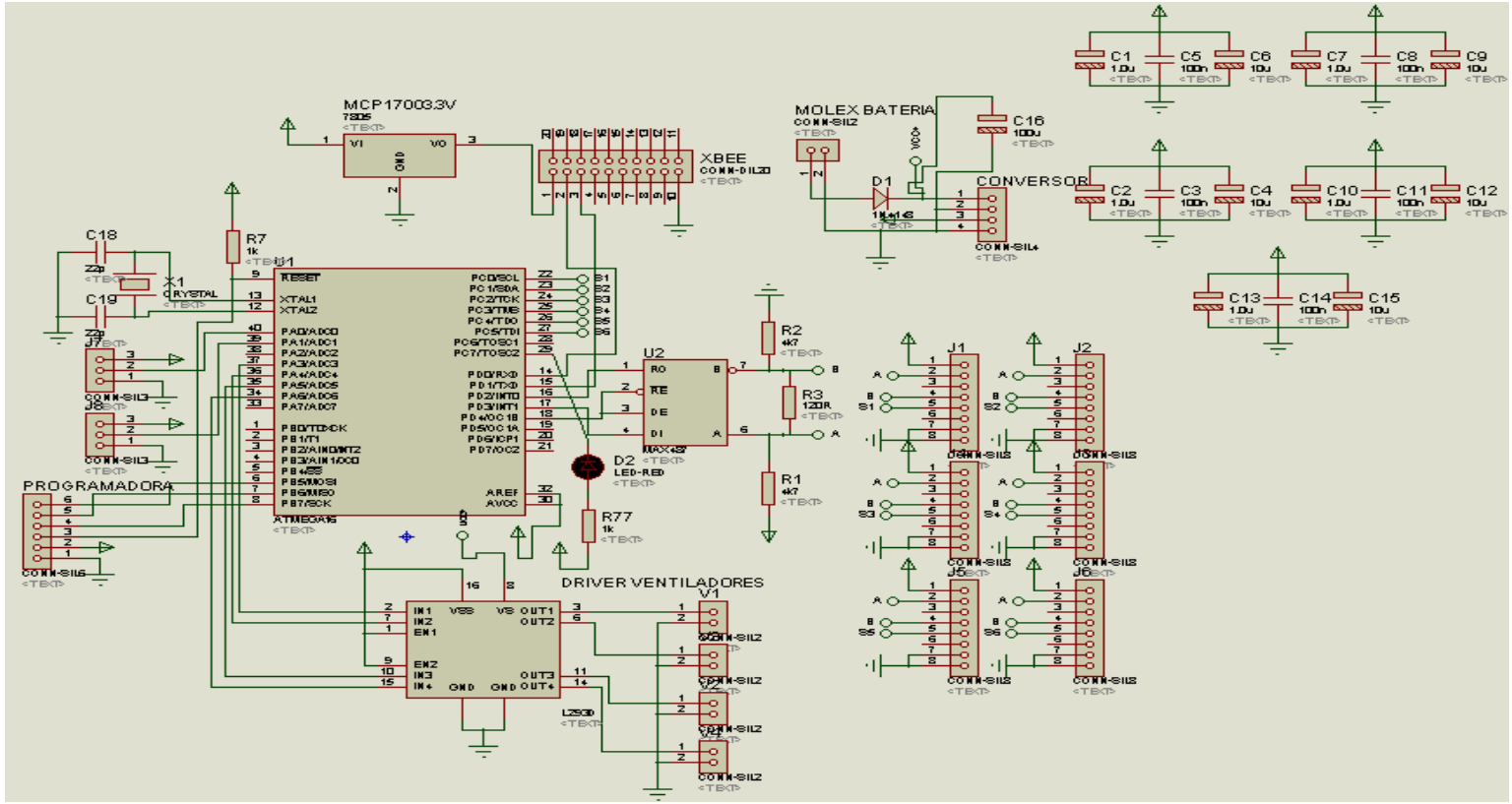


Figura 47.

Elaborado por: Heinerth Romero Macas

Se han incluido seis conectores RJ45 para optimizar el cableado a cada uno de los esclavos, capacitores para eliminar cualquier tipo de ruido, 3 capacitores por cada integrado usado, y los pines de programación del microcontrolador.

Todo la simulación y verificación del funcionamiento del circuito se lo hizo en ISIS de Proteus, ahora lo que falta es hacer el diseño de la PCB, esto se lo hará en ARES del mismo Proteus, solo es cuestión de presionar el botón que hay en la barra de tareas de ISIS y nos enviará directamente a una plantilla en blanco donde se tendrá que ir ubicando cada elemento y haciendo las conexiones correspondientes.

Botón para pasar a ARES

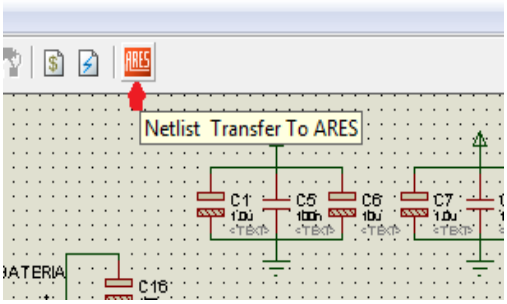


Figura 48.

Elaborado por: Heinerth Romero Macas

Luego de tener todos los componentes ubicados se procede al ruteo de la tarjeta, se puede hacer de forma manual o automática con la opción auto-router de ARES.

Diseño final PCB tarjeta master

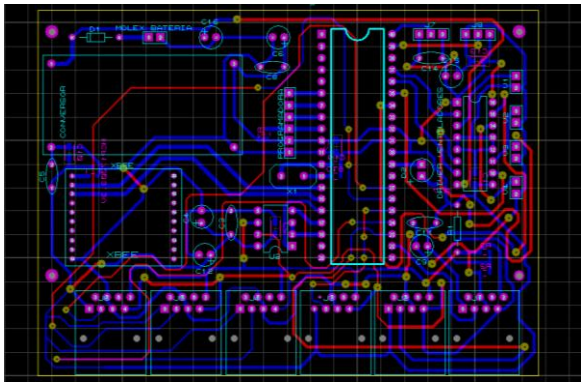


Figura 49.

Elaborado por: Heinerth Romero Macas

Debido a que se tiene gran cantidad de puentes, se escogió hacer el diseño a doble lado, es decir la tarjeta de control master tendrá soldado a ambos lados los componentes electrónicos.

Tarjeta de control master en 3D

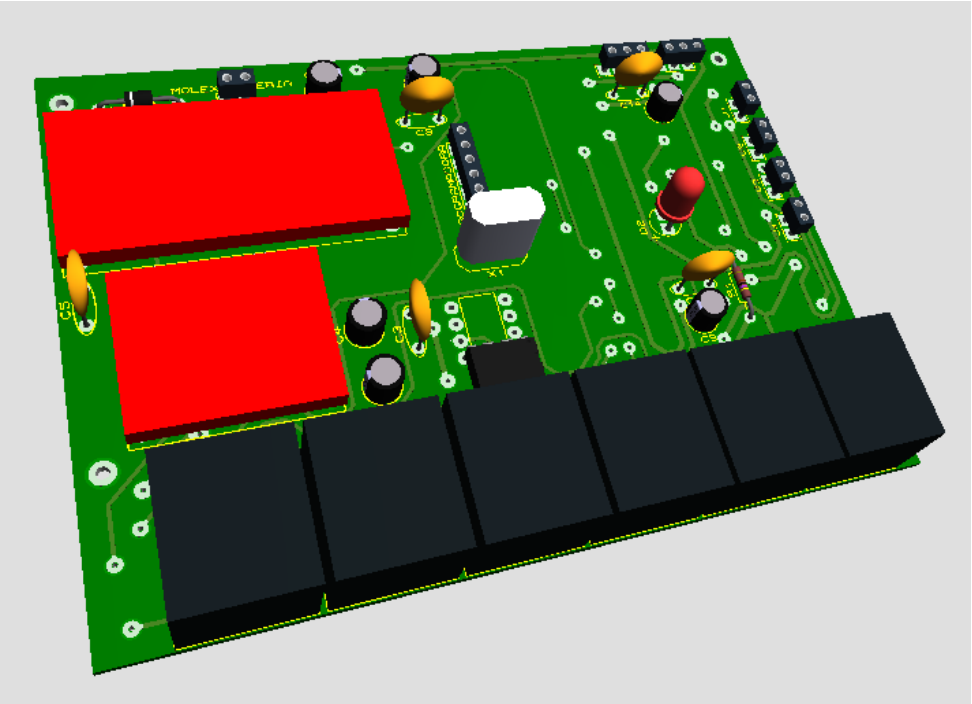


Figura 50.

Elaborado por: Heinerth Romero Macas

3.2.1.2 Diseño de la tarjeta de control esclavo.

De la misma manera como se hizo con la tarjeta de control master se empezará a analizar que microcontrolador es el adecuado para las tarjetas de los esclavos.

Tabla 3.

Requerimientos básicos del microcontrolador esclavo

SLAVE:	
Entradas	**Detalle**
1	Dipswitch1
2	Dipswitch2
3	Dipswitch3
4	Dipswitch4
5	Sensor encoder1
6	Sensor encoder2
7	Sensor encoder3
8	Reset
Salidas	**Detalle**
1	PWM1
2	PWM2
3	Enable
4	Led indicador
Comunicación	**Detalle**
1	Rx Comunicación master
2	Tx Comunicación master
Reloj	**Detalle**
1	Crystal 1
2	Crystal 2
Alimentación	**Detalle**
1	Vcc
2	Gnd
3	Aref
4	Avcc
TOTAL PINES	**20**

Nota. Elaborado por: Heinerth Romero Macas

A diferencia de la tarjeta master que necesitaba dos USART's el esclavo solo necesita uno, esto hace que se busque un microcontrolador más pequeño y de acuerdo al número de pines necesarios, se ha escogido usar para las tarjetas de los esclavos el microcontrolador Atmega8 de 28 pines.

Distribución de pines microcontrolador esclavo

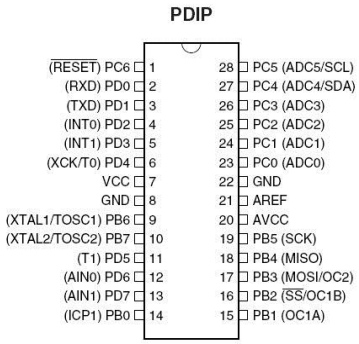


Figura 51.

Fuente: Hoja de datos Atmega8.

El Atmega8 es un microcontrolador del fabricante Atmel de 8 bits y tiene arquitectura Harvard, en la tabla 4 se enlista sus principales características:

Tabla 4.

Características Micro-controlador Esclavo, Atmega8

MASTER:	
Memoria de código	8Kbyte
Memoria de datos	512 bytes de EEPROM
Memoria de datos	1Kbyte de SRAM y 512 bytes de EEPROM
Pines para entrada/salida	23
Frecuencia máxima de trabajao	20Mhz con oscilador externo
Voltaje de alimentación	de 2.7 a 5.5 voltios
Temporizadores	2 de 8 bits y 1 de 16 bits
Canales PWM	3
Fuentes de interrupción	19
Interrupciones externas	2
Canales de conversión analógico/digital	8 de 10 bits
Reloj de tiempo real	sí
Interfaz SPI maestro/esclavo	sí
USART	1
Interfaz serial de dos hilos	sí
Programación “in system”	sí
Oscilador interno configurable	sí
Watchdog timer	sí

Nota. (Espinosa, 2012) **Elaborado por:** Heinerth Guillermo Romero Macas

Luego de escoger el microcontrolador se debe diseñar la tarjeta, que incluye todas las conexiones de todos los elementos electrónicos y conectores que conformarán el circuito, como se hizo con la tarjeta master se usará ISIS de Proteus para la simulación y diseño de la PCB.

A continuación se explicará como fue el proceso de diseño de la tarjeta de control esclavo:

La primera etapa del diseño es la comunicación serial (USART), para esto se usó transceptores MAX485 para montar la red RS-485, como el robot tendrá un esclavo por cada pata, se diseñaron las tarjetas de tal manera que se le asigne una dirección a cada esclavo, para esto se usó un dipswitch de 4 pines, en la figura 52 se puede observar su conexión.

Conexión MAX485 con microcontrolador esclavo

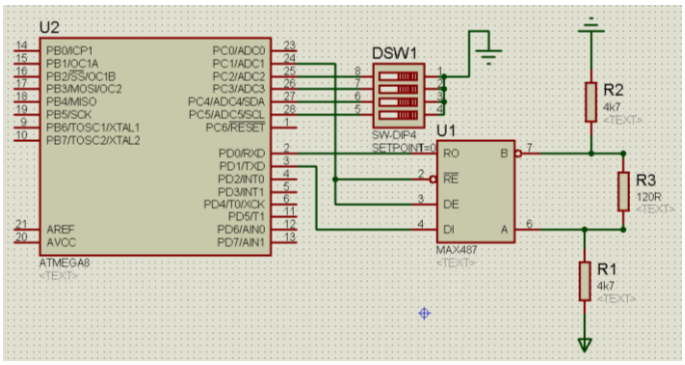


Figura 52.

Elaborado por: Heinerth Romero Macas

También se muestra la manera en que se conectó el transceptor MAX485 con el microcontrolador así como el dipswitch que permitirá dar una dirección a cada uno de los esclavos.

El dipswitch permitirá el paso de GND, por lo que se debe activar a los pines 2, 3, 4 y 5 del puerto C como entradas con resistencia pull-up mediante software.

Lo que se hizo después fue el diseño de las entradas de los sensores, para esto se utilizó una compuerta Smith Trigger 74LS04. La señal que proviene de los encoders

son valores digitales (unos y ceros) pero de acuerdo a la variación de luz del ambiente estos valores pueden variar, es por esto que la señal de los sensores antes de ingresar al microcontrolador pasa por la Smith Trigger; lo que hace esta compuerta es comparar valores de voltaje y mediante histéresis define si el valor recibido se acerca a cualquiera de los dos extremos (5V o 0V), entonces niega y a su vez transmite los valores exactos.

Entrada de sensores infrarrojos (encoders)

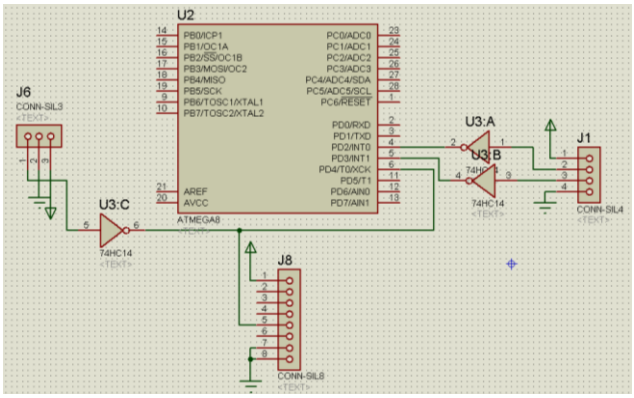


Figura 53.

Elaborado por: Heinerth Romero Macas

Como se observa en la figura 53, se tiene tres entradas de sensores que están conectadas a los pines PD2, PD3 Y PD4 del microcontrolador. J1 es un conector hembra de cuatro pines donde se conecta el encoder, dos pines son para alimentar al mismo (VCC y GND) y dos para las demás señales. Las señales pasan por un circuito integrado Smith Trigger LS7414 antes de los pines PD2 y PD3 que son interrupciones externas del microcontrolador. J6 es el conector para un solo sensor en este caso se conectará el sensor de posición inicial, de igual forma pasa por el Smith Trigger antes de ser conectado al pin PD4. Por último J8 es el conector RJ45 que va conectado a la tarjeta Master, con esto lo que se está haciendo es enviar la señal del sensor de posición inicial a una entrada del microcontrolador esclavo y master a la vez.

La mayor parte del tiempo se necesita conocer la posición exacta de cada pata del robot, es por esto que la tarjeta esclavo fue diseñada para que las señales del encoder ingresen a las dos interrupciones externas del microcontrolador INT0 e INT1, esto

permite monitorear la posición de las seis patas del robot ya que si una sola se desiguala el robot automáticamente perderá estabilidad. El sensor de posición inicial fue conectado a un pin digital normal ya que solo se usa para saber el momento en que se inicia las rutinas que hacen caminar al robot.

Lo siguiente es hacer la conexión para las salidas, la tarjeta de control esclavo, tiene tres salidas hacia la tarjeta de potencia: dos salidas PWM y una para activar el enable del driver de motores.

Otra salida que se tiene, es el led indicador que no debe faltar en los circuitos, esto sirve para realizar pruebas de programación en caso de ser necesario.

Conexión de salidas de la tarjeta esclavo

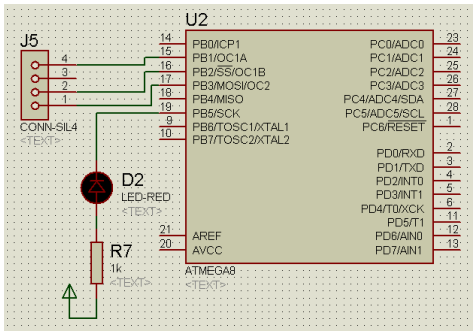


Figura 54.

Elaborado por: Heinerth Romero Macas

Para la conexión de las salidas de PWM es necesario fijarse en la hoja de datos del microcontrolador cuáles son las salidas para generar señales PWM, en este caso se ha usado las salidas del temporizador 1; como se puede observar en la figura 54 son los pines 1 y 2 del puerto B, OC1A Y OC1B respectivamente.

La distribución de las salidas en el conector J5 es debido a que en el otro extremo es decir en la tarjeta de potencia los pines están distribuidos de esa manera, y esto facilitará el cableado hacia la misma.

El led se ha conectado a cualquier pin digital del microcontrolador que no se esté usando, en este caso al pin 5 del puerto B.

Por último se han realizado las conexiones del cristal externo y el reset. Tanto la tarjeta maestro como los seis esclavos deben trabajar a la misma frecuencia para que se puedan comunicar entre ellos, como se hizo con la tarjeta maestro los esclavos tienen un cristal de cuarzo de 11.059200 MHZ, y el reset debe estar conectado a una resistencia pull-up para garantizar que le llegue los 5 volts, se ha decido no poner un pulsador de reset ya que se lo hará mediante software, en caso de que sea necesario.

Conexión cristal de cuarzo y reset

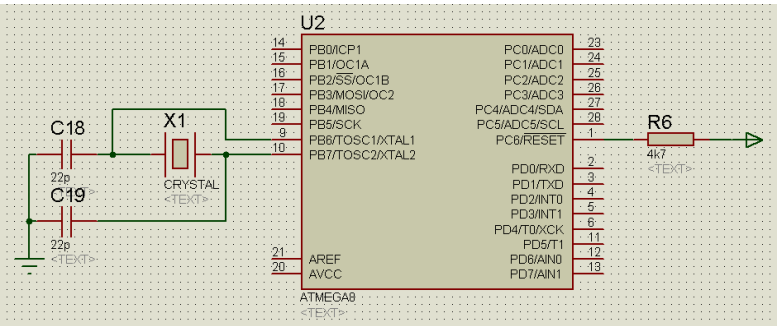


Figura 55.

Elaborado por: Heinerth Romero Macas

Una vez terminado el diseño de las conexiones de la tarjeta se procedió a simular todo el circuito para verificar su funcionamiento. Ya unido todos los esquemas, el circuito final es el mostrado en la figura 56.

Diseño de la tarjeta esclavo hecho en ISIS de Proteus

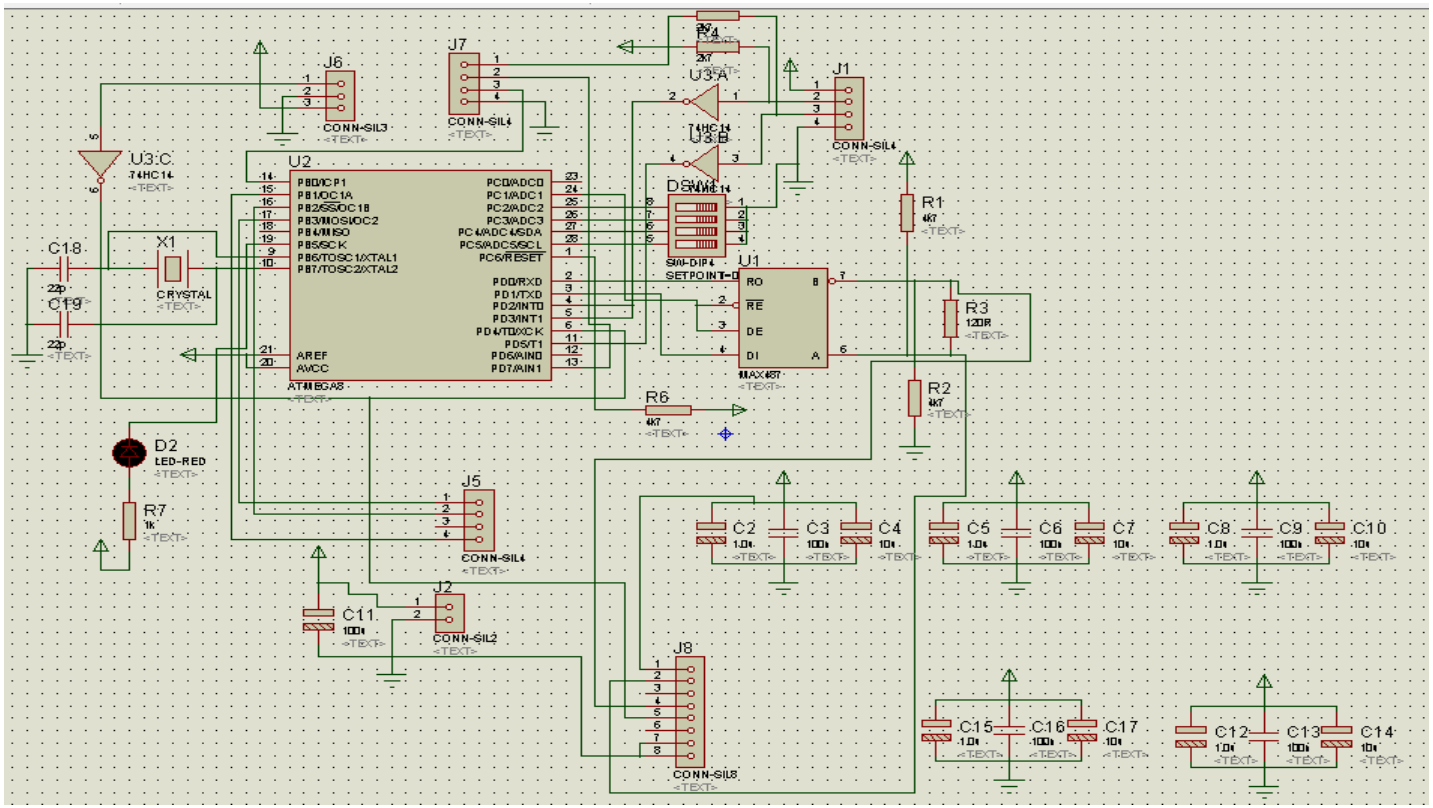


Figura 56.

Elaborado por: Heinerth Romero Macas

Lo que resta es pasar el diseño hecho en ISIS para routear las pistas, esto es un proceso que depende de la habilidad y paciencia de los diseñadores, y al final ha quedado la tarjeta de los esclavos como se observa en la figura 57 y 58.

Diseño final tarjeta de control esclavo

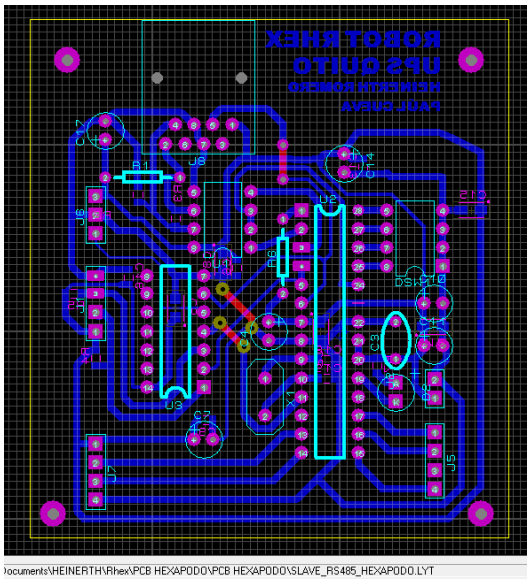


Figura 57.

Elaborado por: Heinerth Romero Macas

Tarjeta de control esclavo en 3D

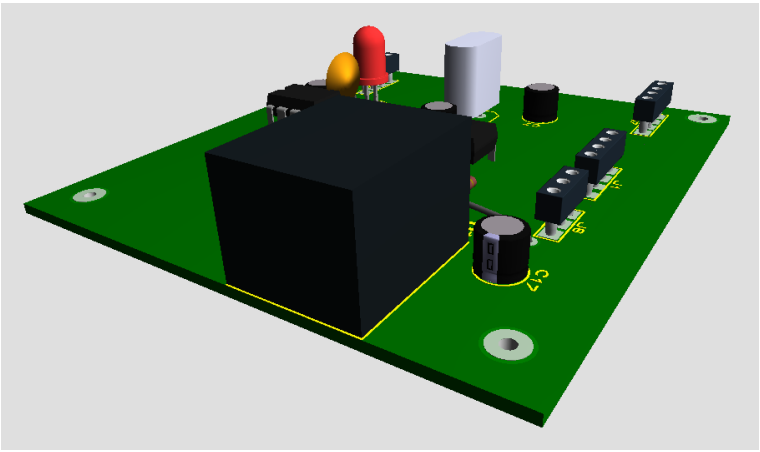


Figura 58.

Elaborado por: Heinerth Romero Macas

3.2.1.3 Construcción de las tarjetas de control.

A diferencia de la construcción del hardware mecánico, que se tuvo que enviar a construir algunas piezas, el hardware electrónico fue completamente construido por los desarrolladores del proyecto.

Luego de hacer el diseño de la tarjeta de control, se procede al grabado y erosionado de la misma.

Para esto se imprimió a laser el diseño hecho en ARES en una hoja de papel fotográfico, esto con el fin de poder estampar el diseño en la baquelita.

La baquelita usada fue una baquelita de fibra de vidrio a doble lado, es decir que ambos lados tiene el recubrimiento de cobre. Luego del proceso de estampado se procede a sumergir la baquelita en un recipiente plástico, con agua y Cloruroférrico (ácido), luego de unos minutos se saca la baquelita y habrá quedado solo las pistas que se necesitan para interconectar todos los elementos del circuito.

Fabricación de las tarjetas

Figura 59.
Elaborado por: Heinerth Romero Macas

Una vez terminado el proceso de erosión del cobre, solo queda por perforar, soldar todos los elementos electrónicos y verificar que no existan cortos circuitos entre las pistas.

Con esto se termina el proceso de diseño y construcción de las tarjetas de control del robot.

Tarjetas terminadas

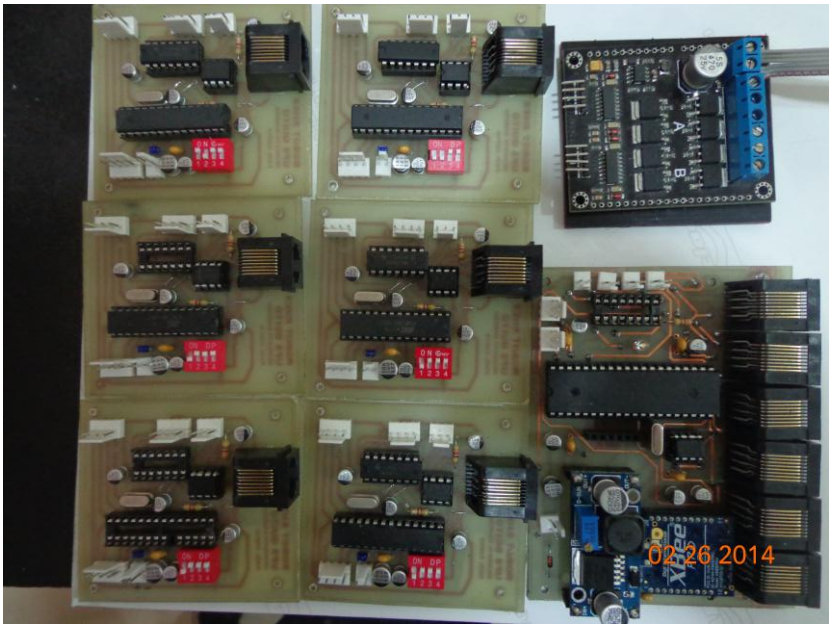


Figura 60.

Elaborado por: Heinerth Romero Macas

3.3 Encoders

El diseño, construcción y montaje de los encoders es muy importante, ya que la tarjeta master debe tener completo control sobre cada una de las seis patas del robot, y la manera más fácil de tener el control por completo es montando encoders en los motores, con esto se garantiza que el microcontrolador master tenga la información exacta de la posición de cada una de las patas.

A continuación se explica el proceso de diseño, construcción y montaje de los encoders.

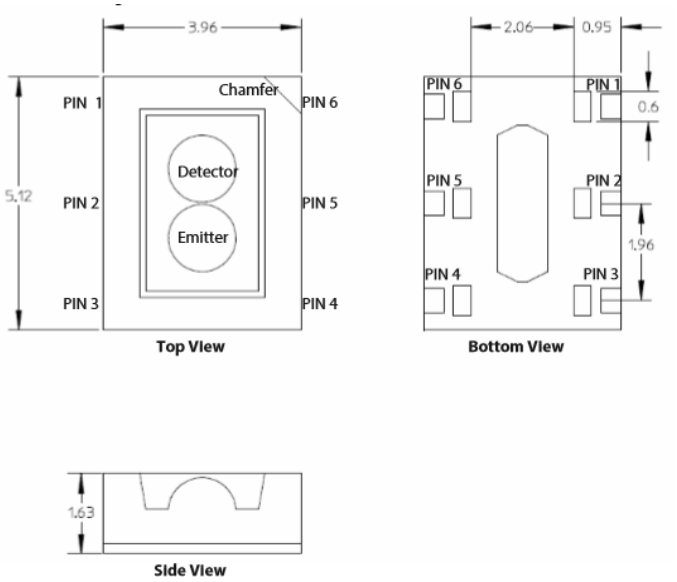
Tomando en cuenta el espacio que se tiene para montar los encoders en el eje del motor, se descartó a los encoders de sensores infrarrojos de barrera o en U y a los encoders con sensores magnéticos o de efecto Hall. Se decidió usar sensores infrarrojos reflectivos, los sensores usados para este fin son de la marca AVAGO Thechnologies, son los que se muestran en la figura 61.

Sensores AEDR-8300-1Wx encoders

Figura 61.
Fuente: Mouser.

Estos sensores son de montaje superficial y tienen 6 pines de conexión, en la figura 62 se muestra la distribución de pines y su conexión.

Pines de conexión sensor AEDR-8300-1Wx



Notes:
1. All dimensions in millimeter.
2. Tolerance X.XX ± 0.15mm.
3. For ease of reference, a chamfer is incorporated on the detector side (pin 6), as shown in the above diagram.

Encoder Pin Configuration

Encoder option	Pin 1	Pin2	Pin3	Pin4	Pin5	Pin6
AEDR-8300-1Wx	Ch B	Gnd	V_{LED}	Gnd	Ch A	Vcc

Figura 62.

Fuente: Mouser.

Ya identificados los pines de conexión se procedió al diseño de una pequeña tarjeta donde se va a soldar el sensor y los componentes recomendados por el fabricante, para el buen funcionamiento del sensor, así como conectores para la alimentación del sensor y la salida de las señales.

Tarjeta del sensor reflectivo del encoder

Figura 63.

Elaborado por: Heinerth G. Romero M

Luego de tener listas las tarjetas, con los sensores soldados y los cables de conexión, se fijó los sensores al motor, para eso se usó un pegamento llamado “super bonder” y se le añadió bicarbonato en polvo, estos dos componentes crean una masa que al solidificarse se vuelve muy resistente haciendo que el pegado sea duradero. En la figura 64 se muestra el acoplamiento del sensor en la parte posterior del motor.

Sensores AEDR-8300-1Wx fijado a la base del motor

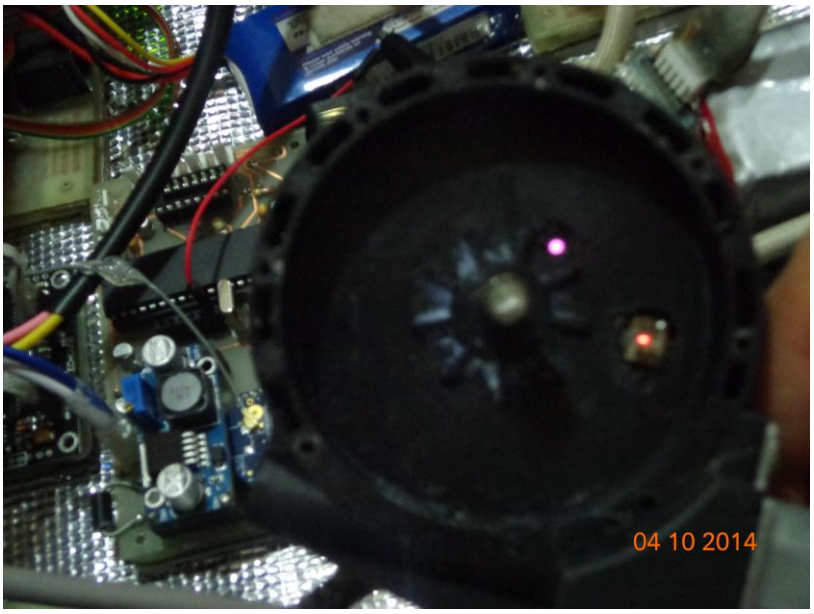


Figura 64.

Elaborado por: Heinerth G. Romero M

Se puede observar dos luces, la luz de color rojo es la luz emitida por el sensor AEDR-8300-1Wx. El siguiente paso es hacer un disco que genere los pulsos. Este disco debe ir fijo en el eje del motor, es decir tendrá que girar junto con el eje para que genere el cambio de color y así el sensor pueda generar la señal que se necesita para el control de las patas.

Para el diseño del disco se usó la herramienta Autocad, con la que se dibujó un círculo con el diámetro correspondiente al del eje del motor, en este caso el eje al cual se va a fijar el disco del encoder es un piñón.

Piñón donde se debe pegar el encoder.

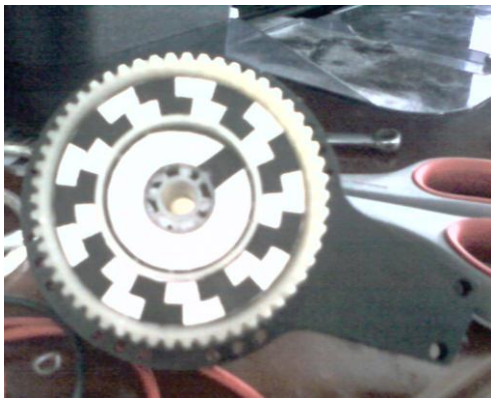


Figura 65.

Elaborado por: Heinerth G. Romero M

En la figura 65, se muestra como debe ser adherido el disco del encoder dentro del piñón, el diseño que se muestra fue impreso en una cartulina blanca sin embargo el haz de luz emitido por el sensor no se reflejaba sobre el color blanco, por lo que se tuvo que buscar una hoja que sea capaz de reflejar el haz de luz infrarrojo y después de varias pruebas con diferentes tipos de papel se encontró que el papel aluminio es capaz de reflejar la luz del sensor, en la papelería que se lo compró lo llaman papel aluminio adhesivo.

Diseño final pegado en el piñón y/o eje del motor

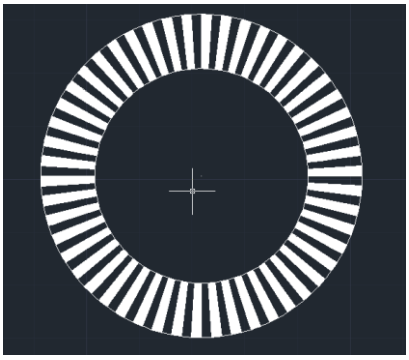


Figura 66.

Elaborado por: Heinerth G. Romero M

El diseño final que se acopló en el interior del piñón del motor es el mostrado en la figura 66, y fue impreso en una lámina de papel aluminio adhesivo.

3.4 Desarrollo de software

En este capítulo se detallan las etapas que se realizaron para la elaboración e implementación de los algoritmos de control para el microcontrolador maestro, los microcontroladores esclavos, y el computador que contiene la interfaz de usuario.

Debido a que los diseñadores poseen conocimientos de programación en lenguaje C, para la edición y compilación de código de los microcontroladores se decidió usar el entorno de desarrollo de la empresa MikroElektronika, mikroC Pro For AVR el cual es un compilador con todas las funciones ANSI C para los dispositivos AVR de Atmel® y con gran cantidad de bibliotecas de software, entre sus funcionalidades cabe destacar que trae incorporado un depurador de código y un cronógrafo (MikroElektronika, 2013).

Para el desarrollo de la interfaz gráfica de usuario o GUI se decidió usar la herramienta Matlab, debido a que, entre sus muchas prestaciones, permite crear interfaces gráficas, manejar puertos seriales, manipular matrices, y codificar de manera fácil algoritmos matemáticos, características adecuadas para el desarrollo del proyecto tomando en cuenta que se necesita implementar el algoritmo de reconocimiento de personas sobre la imagen de la cámara de video.

Las etapas efectuadas son las siguientes: diseño de software, y codificación e implementación de software.

3.4.1 Diseño de software.

Los movimientos que debe lograr el robot serán codificados en los microcontroladores y su ejecución será comandada por el operador a través de la interfaz gráfica de usuario.
Para comprender el funcionamiento del sistema, las tareas de cada uno de los componentes a ser programados, se describen a continuación.

3.4.1.1 Tareas del microcontrolador esclavo.

- Recibir y procesar datos provenientes del microcontrolador maestro;
- Leer señales de encoders para determinar la posición de los motores;
- Controlar velocidad de motores según el valor que establezca el maestro;
- Enviar datos de confirmación al microcontrolador maestro.

3.4.1.2 Tareas del microcontrolador maestro.

- Recibir y procesar los comandos provenientes del computador;
- Adquirir datos de sensores analógicos de temperatura y de inclinación;
- Enviar datos de lectura de sensores al computador;
- Enviar comandos, a los microcontroladores esclavos;
- Recibir y procesar datos de confirmación provenientes de cada uno de los microcontroladores esclavos;
- Ejecutar algoritmos de giro y desplazamiento.

3.4.1.3 Tareas de la interfaz gráfica de usuario.

La interfaz gráfica de usuario o GUI deberá cumplir tres tareas: visualización de datos, envío de comandos, y ejecución del algoritmo de reconocimiento de personas.

- **Datos a visualizar:**
  - Temperatura en grados Celcius al interior y al exterior del robot
  - Inclinación del robot
  - Video en tiempo real de la perspectiva frontal del robot

- **Comandos a enviar:**
  - Adelante
  - Girar izquierda
  - Girar derecha
  - Detenerse

- Recostarse
- Levantarse
- Petición de temperatura al interior y al exterior del robot
- Petición de Inclinación del robot
- Iniciar o detener transmisión de video de la cámara
- Reiniciar el sistema

Cada comando está asociado a un botón específico dentro de la interfaz de usuario.

- **Reconocimiento de personas:**

Sobre las imágenes capturadas con la cámara de video se implementa un algoritmo diseñado para detectar porciones de color de piel de personas, mostrando con un rectángulo de seguimiento el área de color más grande detectado.

3.4.1.4 Mensajes de datos.

Son cadenas de caracteres ASCII que contienen información que se transmite entre la interfaz de usuario y los microcontroladores maestro y/o esclavos. Existen cuatro modelos: el mensaje para el maestro (MM), el mensaje para el esclavo (ME), el mensaje de confirmación de los esclavos al maestro (ACK), y el mensaje de respuesta del maestro a el computador (MRC).

3.4.1.4.1 Mensaje para el maestro (MM).

Contiene la información de una orden que se envía al robot como tal; se transmite solamente al microcontrolador maestro; su estructura está compuesta por el comando de operación y el finalizador de mensaje, tal como se muestra en la figura 67.

Estructura mensaje de datos maestro

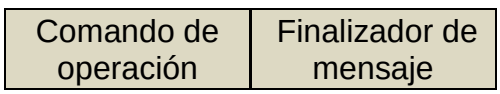


Figura 67.

Elaborado por: Paúl Cueva Castillo

- Comandos de operación

Corresponden a los enumerados en la sección “Comandos a enviar”; están representados por caracteres ASCII referentes al rango de valores comprendido entre 64 y 96, es decir las letras mayúsculas. A continuación se muestran en la tabla 5:

Tabla 5.

Comandos de operación en el mensaje para el maestro

Comando	Carácter
Adelante	'W'
Girar izquierda	'A'
Girar derecha	'D'
Detener movimiento / Levantarse	'X'
Acostarse	'J'
Petición de temperatura al interior del robot	'L'
Petición de temperatura al exterior del robot	'I'
Petición de Inclinación del robot	'U'
Reiniciar sistema	'R'

Nota. Elaborado por: Paúl Cueva Castillo

El MM permanece oculto para el operador ya que a cada comando le corresponde un botón dentro de la interfaz de usario, es decir, si por ejemplo se presiona el botón adelante, automáticamente se envían el carácter del comando correspondiente para dicha orden mas el carácter finalizador. El comando *Reiniciar sistema* pone en condiciones iniciales todos los registros de los MCU’s maestro y esclavos.

El finalizador de mensaje se configuró para ser el carácter ‘m’.

3.4.1.4.2 Mensaje para el esclavo (ME).

Contiene la información de una orden dirigida para un esclavo o un grupo de ellos. Está compuesto por el identificador de esclavo (ID), el comando de operación, y el carácter finalizador de mensaje como se observa en la figura 68.

Estructura mensaje de datos esclavo

ID	Comando de operación	Finalizador de mensaje

Figura 68.

Elaborado por: Paúl Cueva Castillo

- Identificador de esclavo (ID)

El ID es el carácter que permite especificar el esclavo o los esclavos a los que estará dirigido el mensaje. Puede ser: bien la ubicación física (por fila o columna) de la pata a la que está asociado; bien la dirección de esclavo; bien la paridad (calidad de par o impar) que tiene su dirección; o bien el carácter de broadcast al cual responden todos los esclavos.

La ubicación, la dirección, y la paridad de los esclavos se muestran en la figura 69.

Ubicación dirección y paridad de los esclavos según la pata a la que pertenecen

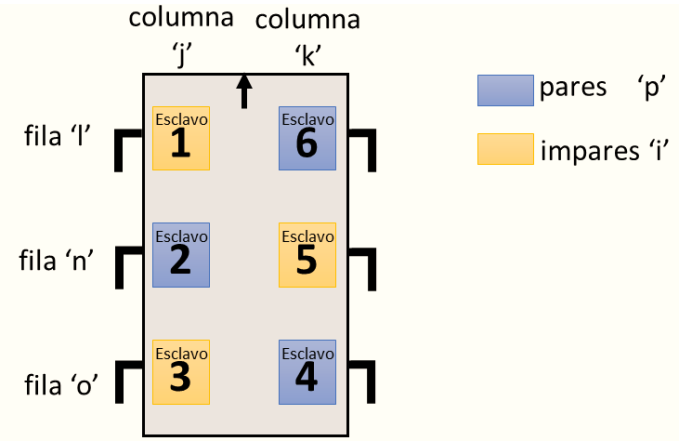


Figura 69.

Elaborado por: Paúl Cueva Castillo

Lo primero que hace el software del esclavo es identificar la dirección de esclavo, la fila y la columna a la que pertenece, y evaluar si su dirección es par o impar; de esta forma a cada esclavo le corresponde más de un ID mediante el cual se puede acceder al mismo.

En la tabla 6 se resumen las formas de identificar a un esclavo:

Tabla 6.

Formas de identificar a un esclavo

ID	Carácter
Por dirección	'1', '2', '3', '4', '5', '6'
Por paridad	'p', 'i'
Por fila	'l', 'n', 'o'
Por columna	'j', 'k'
Por broadcast	'b'

Nota. Elaborado por: Paúl Cueva Castillo

Por ejemplo si se invoca a la columna ‘k’ responderán los esclavos 6 ,5 y 4; si se invoca a los pares ‘p’ responderán los esclavos 2, 4 y 6; si se invoca a la dirección ‘6’ sólo responderá el esclavo 6; y así sucesivamente. También hay la posibilidad de comunicarse con todos a la vez enviando el identificador de broadcast ‘b’.

Estas distinciones permiten optimizar código a la hora de programar las secuencias de desplazamiento debido a que por lo general dichas secuencias consisten en dar pasos alternando las patas de dos en dos o de tres en tres. Por ejemplo, para la secuencia de caminar los pasos se alternan entre las patas pares e impares, de modo que requiere menos líneas de código enviar una orden directamente a todos los pares y luego a todos los impares en lugar de hacerlo esclavo por esclavo.

- Comandos de Operación

Están representados por caracteres ASCII cuyos valores están fuera del rango comprendido entre 64 y 96, es decir todos los caracteres que no son letras mayúsculas. Se dividen a su vez en comandos de configuración de parámetros, comandos de posicionamiento, y comando de petición de confirmación.

- Comandos de configuración de parámetros

Se utilizan para configurar la velocidad de una o más patas y para establecer el número de pulsos de encoder que deberán recorrer los motores durante las rutinas de movimiento. Se almacenan en la memoria EEPROM del MCU con el fin de recuperarlos en caso de reinicio. Los valores de velocidad son enviados dentro del comando del mensaje para el esclavo (ME) en cantidades porcentuales y es el software del esclavo el encargado de convertirlos a valores de PWM para ser enviados a los drivers. Los comandos de configuración y el carácter que corresponde a cada uno se muestran en la tabla 7.

- Comandos de posicionamiento

Sirven para situar una o varias patas a una distancia específica respecto a la posición inicial, esto se logra moviendo el motor un número n de pulsos contados desde la posición inicial. Los comandos de posicionamiento y el carácter que corresponde a cada uno se muestran en la tabla 7.

- Comando de pedido de confirmación

Se utiliza cuando el maestro necesita saber si un esclavo ha concluido una tarea. Este comando se envía especificando a uno solo de los esclavos como receptor; es decir que el ID debe ser únicamente por dirección puesto que en una red RS-485 solamente responde un esclavo a la vez. Cuando un esclavo recibe el comando, responde con el mensaje de confirmación o ACK (el cual se verá más adelante). El comando de pedido de confirmación se muestra en la tabla 7.

Tabla 7.

Comandos de operación en el mensaje para el esclavo

Tipo	Comando	Carácter a enviar
Configuración de parámetros	Establecer velocidad de tracción	'>' + [% de pwm]
	Establecer velocidad de avance	'v' + [% de pwm]
	Establecer pulsos de tracción	'#' + [núm. de pulsos]
	Establecer pulsos de avance	'@' + [núm. de pulsos]
	Establecer pulsos de giro	'*' + [núm. De pulsos]
Posicionamiento de patas	Mover pata a la posición 0 en sentido horario con velocidad de avance	'a'
	Mover pata a la posición 0 en sentido horario con velocidad de tracción	'c'
	Mover pata pulsos avance en sentido horario con velocidad de avance	'd'
	Mover pata pulsos de tracción en sentido horario con velocidad de tracción	'e'
	Mover pata a posición 0 en sentido anti horario con velocidad de avance	'g'
	Mover pata a posición 0 en sentido anti horario con velocidad de tracción	'h'
	Mover pata pulsos avance en sentido anti horario con velocidad de avance	'q'
	Mover pata pulsos de tracción en sentido horario con velocidad de tracción	's'
	Mover pata pulsos de giro en sentido horario con velocidad de avance	't'
	Mover pata pulsos de giro en sentido anti horario con velocidad de avance	'u'
	Mover pata un número n de pulsos en sentido horario con velocidad de avance	'y'
	Mover pata un número n de pulsos en sentido anti horario con velocidad de avance	'z'
Confirmación	Pedir confirmación	'r'

Nota. La diferencia entre la velocidad y tracción, y la utilidad de cada una de estas se comprenderá más adelante.

Elaborado por: Paúl Cueva Castillo

El finalizador en el mensaje para el esclavo se configuró para ser el carácter 'f'.

3.4.1.4.3 Mensaje de confirmación (ACK).

Es el mensaje de retorno de un esclavo al maestro confirmando que ha concluido una tarea. El ACK es el mensaje que envía un esclavo cuando ha recibido el comando de pedido de confirmación, de tal modo que no es automático, sino que se remite sólo cuando el maestro hace la petición.

Para reconocer cuál de los esclavos envía el ACK, la confirmación es la propia dirección del esclavo. En la figura 70 se muestra la estructura de éste mensaje.

Estructura del mensaje de confirmación

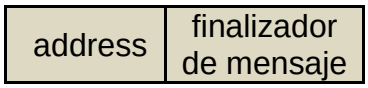


Figura 70.

Elaborado por: Paúl Cueva Castillo

El finalizador en el mensaje de confirmación se configuró para ser el carácter 'f'.

3.4.1.4.4 Mensaje de respuesta del maestro al computador (RMC).

Es el mensaje que envía el maestro al computador, contiene la información de la lectura de alguno de los sensores del robot ya convertida a un valor de temperatura o inclinación (según el caso), que se envía para ser visualizado en la GUI. La figura 71 muestra su estructura:

Estructura del mensaje de confirmación

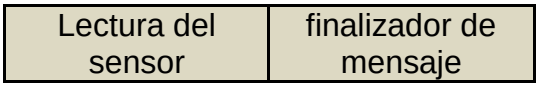


Figura 71.

Elaborado por: Paúl Cueva Castillo

El finalizador de mensaje se configuró para ser el carácter 'm'.

El funcionamiento de la red de comunicación dentro del sistema, origen y destino de cada uno de los mensajes de datos, se muestran en la figura 72.

Mensajes de datos dentro de la red de comunicación

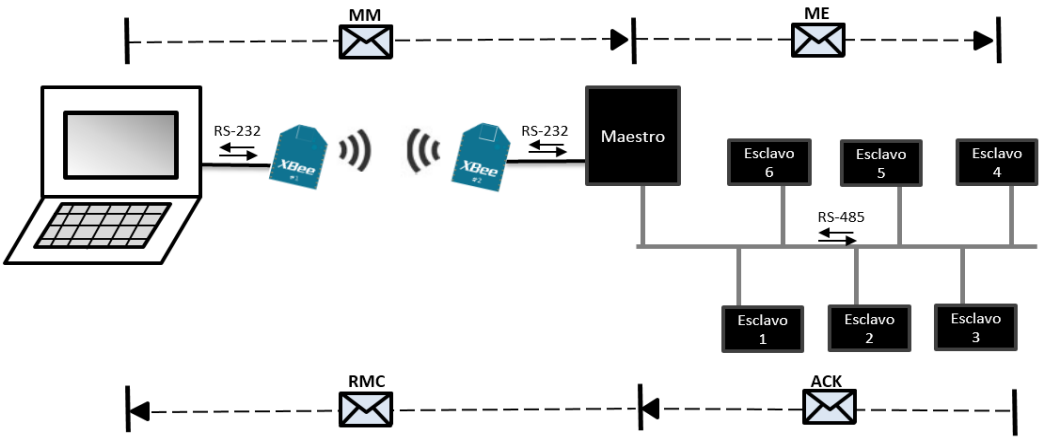


Figura 72.

Elaborado por: Paúl Cueva Castillo

En la red RS-485, el maestro y los esclavos, siempre están en modo escucha de datos, únicamente cuando alguno debe enviar un mensaje cambia su estado a modo envío, e inmediatamente después de realizar esta acción regresa al modo escucha.

3.4.2 Codificación e implementación de software.

La estructura general de la programación en los microcontroladores maestro y esclavos se muestra en la figura 73.

Estructura general del software en el MCU maestro y el MCU esclavo

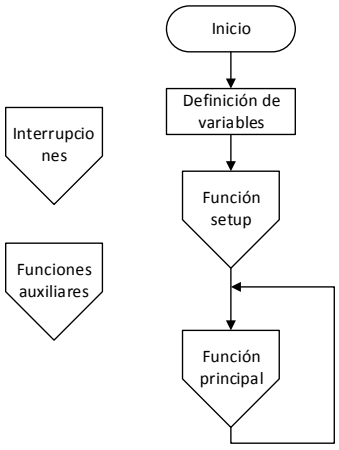


Figura 73.

Elaborado por: Paúl Cueva Castillo

3.4.2.1 Software microcontrolador esclavo.

La programación es la misma para cada uno de los esclavos, se distinguen solamente por su dirección de esclavo.

3.4.2.1.1 Definición de variables.

Tabla 8.

Definición de variables de programación en microcontrolador esclavo

Nombre	Tipo	Descripción
address	entero	Almacena la dirección del esclavo, del 1 al 6.
paridad	caracter	Almacena la paridad del esclavo, 'p' si es par, 'i' si es impar.
fila	caracter	Almacena la fila a la que pertenece el esclavo, 'l', 'n', 'o'.
columna	caracter	Almacena la columna ‘j’ o ‘k’ a la que pertenece el esclavo.
dataRxMaster[ ]	arreglo de caracteres	Almacena carácter por carácter el mensaje proveniente del maestro; la posición cero corresponde al ID mientras la uno al comando de operación.
velocidadTraccion	entero	Almacena el valor de pwm que tendrá la pata al hacer tracción durante el desplazamiento.
velocidadAvance	entero	Almacena el valor de pwm que tendrá la pata cuando esté en el aire durante el desplazamiento.
pulsosTraccion	entero	Almacena el número de pulsos que recorrerá el encoder de la pata al hacer tracción durante el desplazamiento.
pulsosAvance	entero	Almacena el número de pulsos que recorrerá el encoder de la pata cuando esté en el aire durante el desplazamiento.
pulsosGiro	entero	Alamacena el número de pulsos que recorrerá el encoder durante las rutinas de giro.
pulsosEncoder	entero	Almacena el número de pulsos que ha contado el encoder de posicionamiento angular mediante la interrupción *INT0.*
mensajeRecibido	bit/bandera	Se activa dentro de la interrupción para recepción de datos cuando se ha recibido un mensaje proveniente del maestro; se desactiva en la función principal.
posicionInicial	bit/bandera	Se activa cuando la interrupción detecta la posición inicial de la pata; se desactiva en la función principal.

Nota. Elaborado por: Paúl Cueva Castillo

3.4.2.1.2 Función setup.

En esta función se preparan los parámetros que se van a utilizar a lo largo del programa: se inicializan módulos; se configuran puertos de entrada/salida; se configuran el uso de interrupciones; se extraen los parámetros de la memoria EEPROM; y se identifica la paridad (si la dirección es par o impar), la fila y la columna a la que pertenece el esclavo. El diagrama de flujo de la función se muestra en la figura 74.

Función setup en microcontrolador esclavo

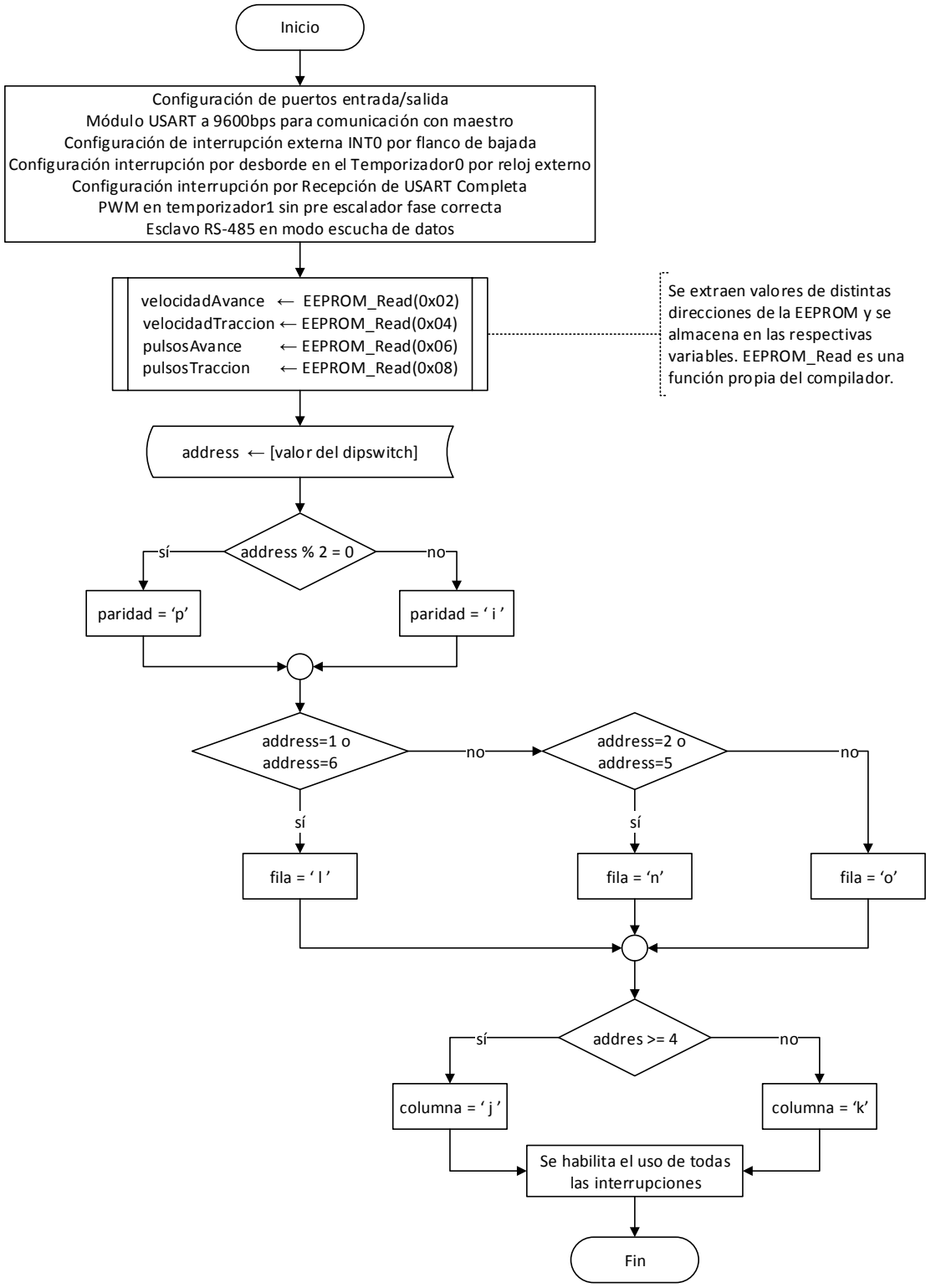


Figura 74.

Elaborado por: Paúl Cueva Castillo

3.4.2.1.3 Funciones por interrupción.

- Función *Recepción USART Completa*

Para evitar la pérdida de información, la captura de datos seriales se realizó usando la interrupción correspondiente al evento de *Recepción USART Completa* que se ejecuta cuando existe un dato no leído en el buffer de recepción (buffer Rx USART). Dentro de la función se almacenan los caracteres uno a uno en el arreglo *dataRxMster[ ]* a medida que van llegando, esto se hace hasta encontrar el finalizador de mensaje, el cual una vez recibido se activa una bandera que permitirá saber cuándo se tiene un mensaje completo. Además si el dato es el carácter 'R', se activa el Watchdog y se resetea el microcontrolador después de 16ms. En la figura 75 se observa el diagrama de flujo de esta función.

Función por interrupción *Recepción USART Completa* en el microcontrolador esclavo

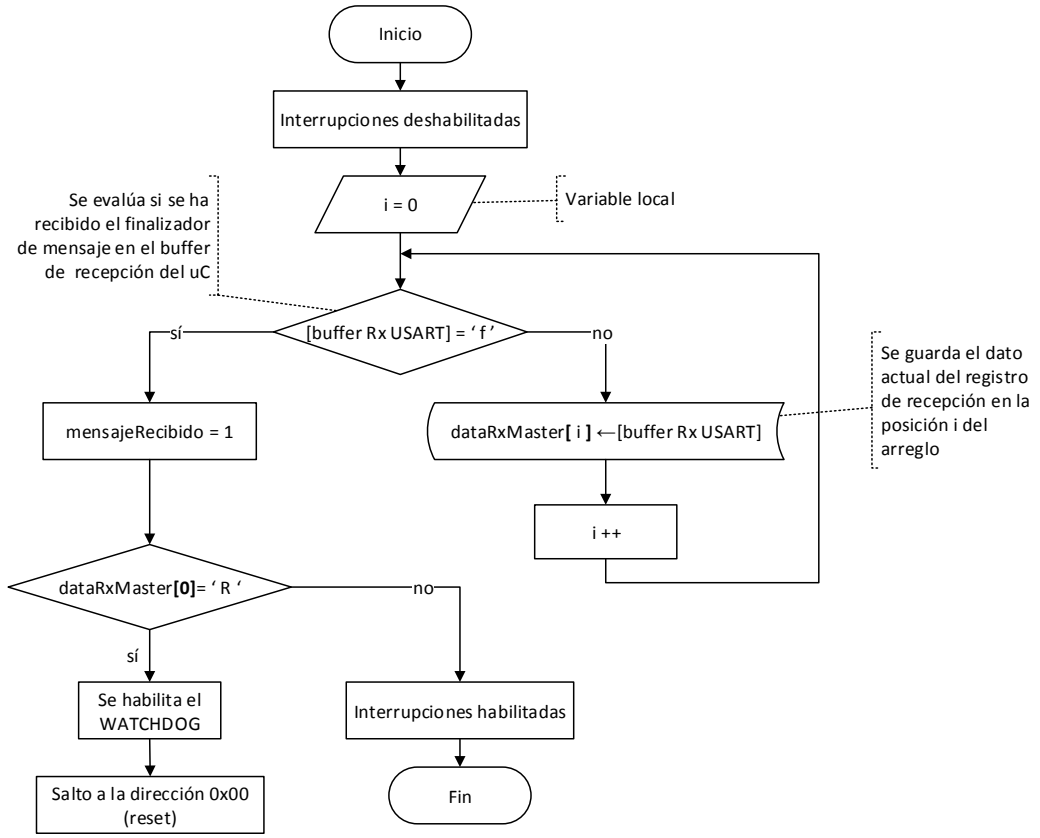


Figura 75.

Elaborado por: Paúl Cueva Castillo

- Función *INT0*

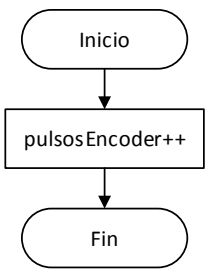
Para detectar los pulsos del encoder de posicionamiento angular se utilizó la interrupción externa *INT0* configurada para que se active al detectar un flanco de bajada. En la función se incrementa la variable encargada de registrar el conteo de pulsos. En la figura 76.a se observa el diagrama de esta función.

- Función *Desborde del Temporizador0*

Para detectar el pulso del encoder de posicionamiento inicial se utilizó la interrupción correspondiente al evento *Desborde del Temporizador0*, que se activa cuando el registro de conteo TCNT0 pasa de 255 a 0 (temporizador de 8 bits), dicho registro se incrementa automáticamente por cada ciclo de un reloj externo o interno; en este caso se configuró para reloj externo (pulso del encoder), y se cargó el registro de conteo directamente con el valor de 255, de modo que al primer pulso detectado se produzca el desborde y se dispare la interrupción. Dentro de la función se recarga el valor de TCNT0 a 255 y se activa una bandera que permitirá saber cuándo el motor ha llegado a la posición inicial. En la figura 76.b se observa el diagrama de esta función.

Funciones por interrupción para la lectura de encoders en microcontrolador esclavo

a) *INT0* para lectura de encoder de posición inicial



b) *Desborde del T0* para lectura de encoder de posicionamiento angular

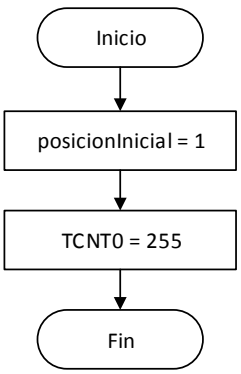


Figura 76.

Elaborado por: Paúl Cueva Castillo

3.4.2.1.4 Funciones auxiliares.

- Función *StrToNum*

Transforma una cadena de caracteres a un número entero decimal. Se utiliza para extraer un valor numérico de una trama de datos, como el caso del valor de pwm. Recibe como parámetro la cadena de caracteres y la posición inicial desde la cual va a empezar a extraer los datos. En la figura 77 se muestra el diagrama de flujo para esta función:

Función auxiliar *StrToNum* en microcontrolador esclavo

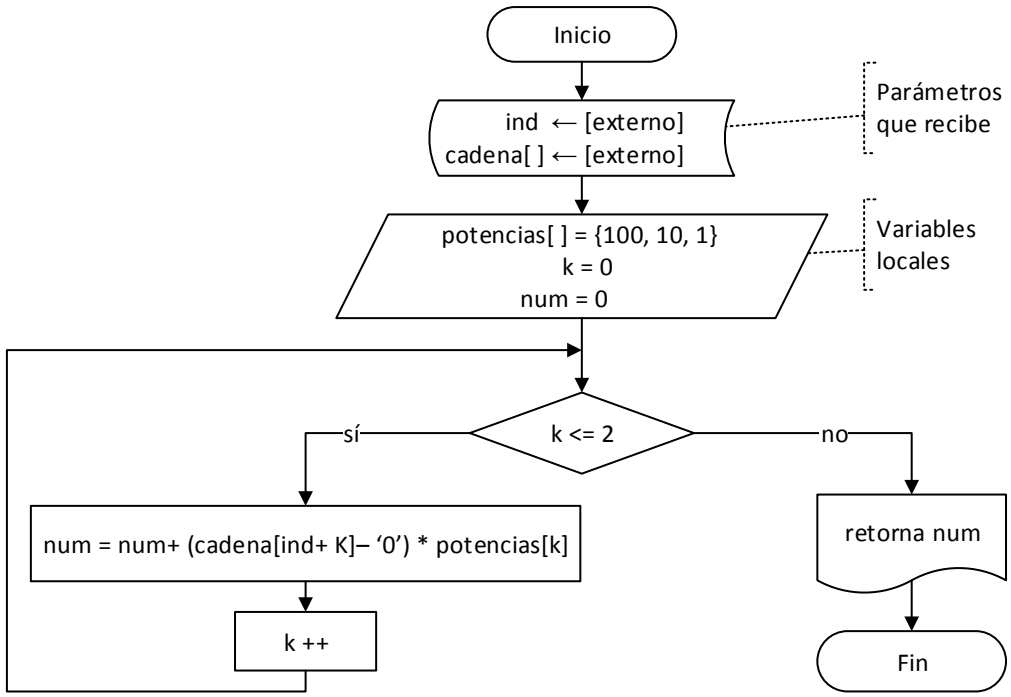


Figura 77.

Elaborado por: Paúl Cueva Castillo

- Función *enviar ACK*

Se usa para enviar el mensaje de confirmación al maestro, en la figura 78 se muestra su diagrama de flujo.

***Función enviarACK* para enviar mensaje de confirmación al maestro**

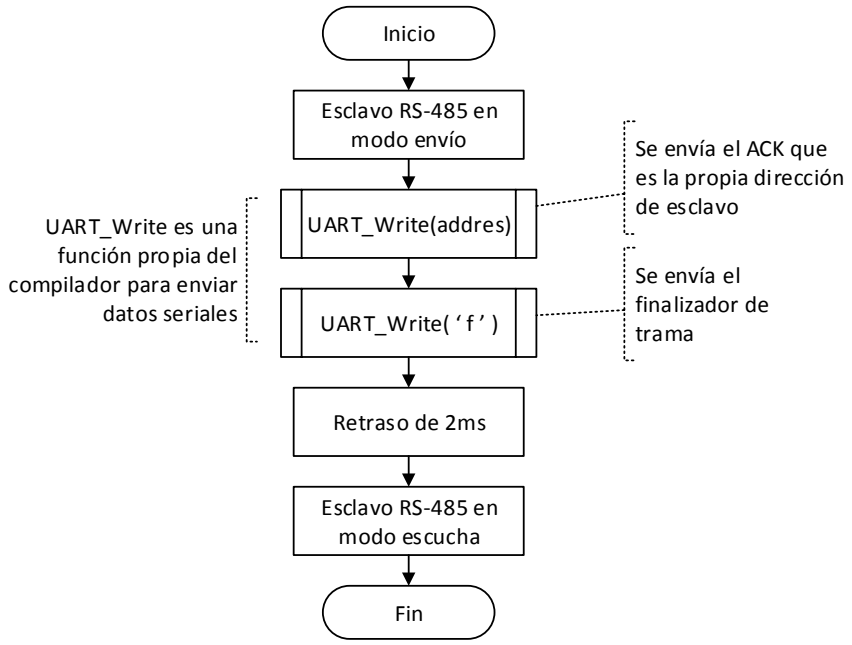


Figura 78.

Elaborado por: Paúl Cueva Castillo

- Función *adelante*

Mueve el motor en sentido horario, recibe como parámetro el valor para la modulación por ancho de pulso (PWM), se envía este valor al canal A del driver y al canal B se envía un 1 lógico. Su diagrama de flujo se muestra en la figura 79.a.

- Función *atrás*

Mueve el motor en sentido anti horario, recibe como parámetro el valor para la modulación por ancho de pulso (PWM), se envía este valor al canal B del driver y al canal A se envía un 1 lógico. Su diagrama de flujo se muestra en la figura 79.b.

- Función *freno*

Se detiene el motor enviando un 1 lógico a ambos canales del driver. Su diagrama de flujo se muestra en la figura 79.c.

Funciones auxiliares *adelante, atrás,* y *freno* en microcontrolador esclavo

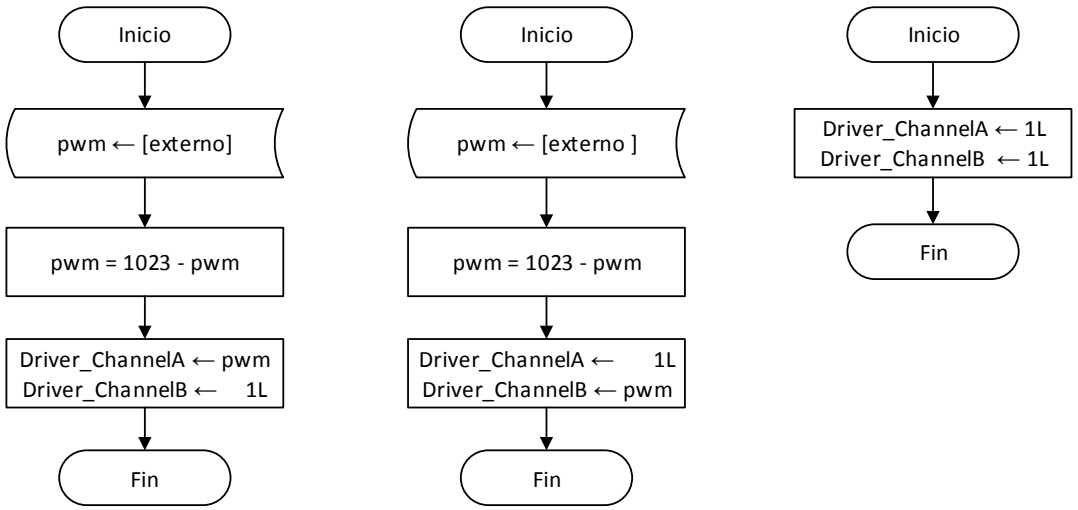


Figura 79.

Elaborado por: Paúl Cueva Castillo

3.4.2.1.5 Función principal.

La función principal consiste en un bucle infinito en el cual constantemente se consulta si se ha recibido un mensaje proveniente del maestro, si la respuesta es afirmativa, se evalúa el ID del mensaje (posición 0 del arreglo), si dicho ID es válido se procede a evaluar el comando (posición 1 del arreglo), y según éste el esclavo realiza la acción solicitada.

Para una mejor comprensión en el diagrama de flujo se ha sustituido la variable *dataRxMaster[0]* por la palabra *ID*, y la variable *dataRxMaster[1]* por la palabra comando, que son los objetos a los que hacen referencia.

Función *principal* en microcontrolador esclavo

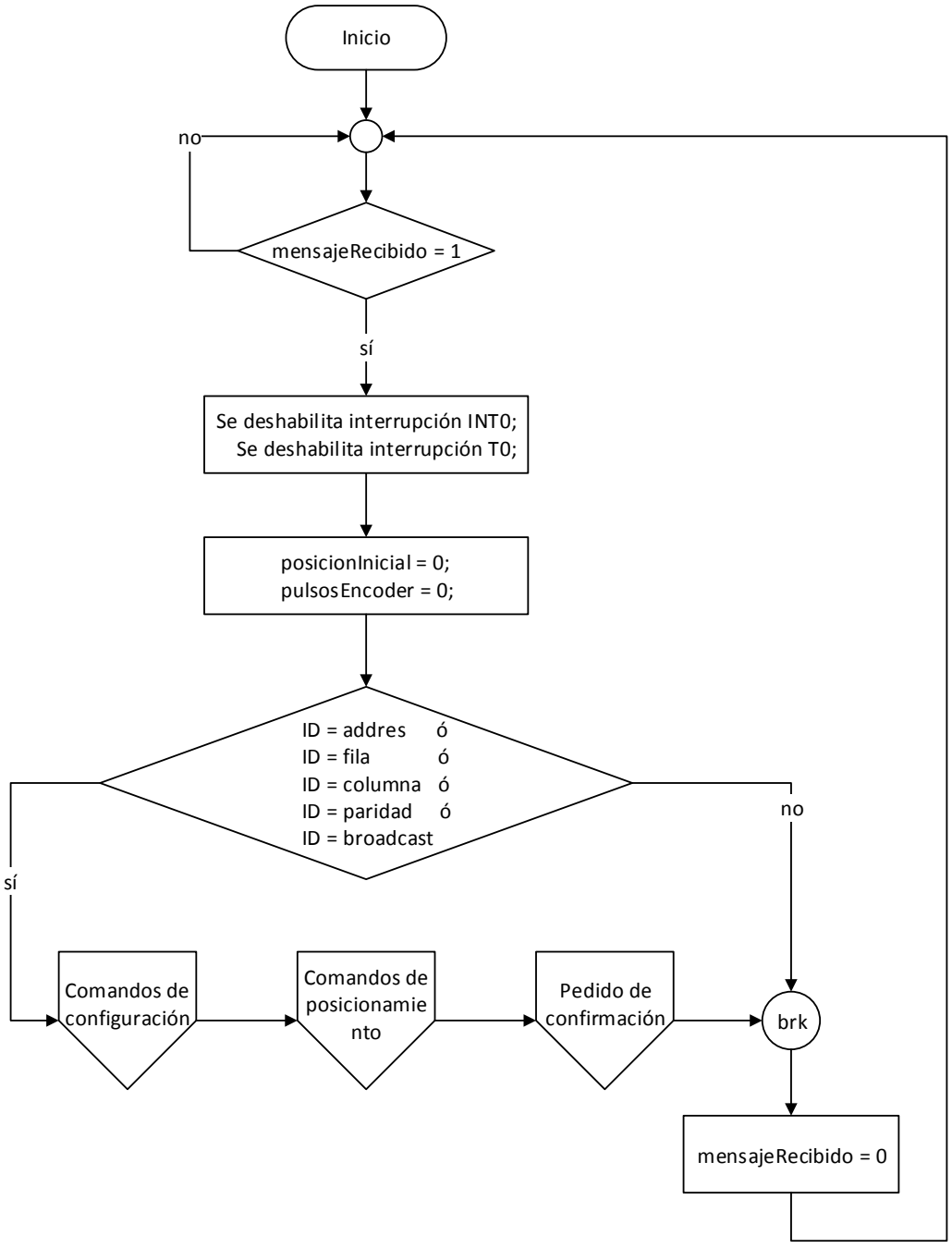


Figura 80.

Elaborado por: Paúl Cueva Castillo

Comandos de configuración para la función *principal* en el MCU esclavo

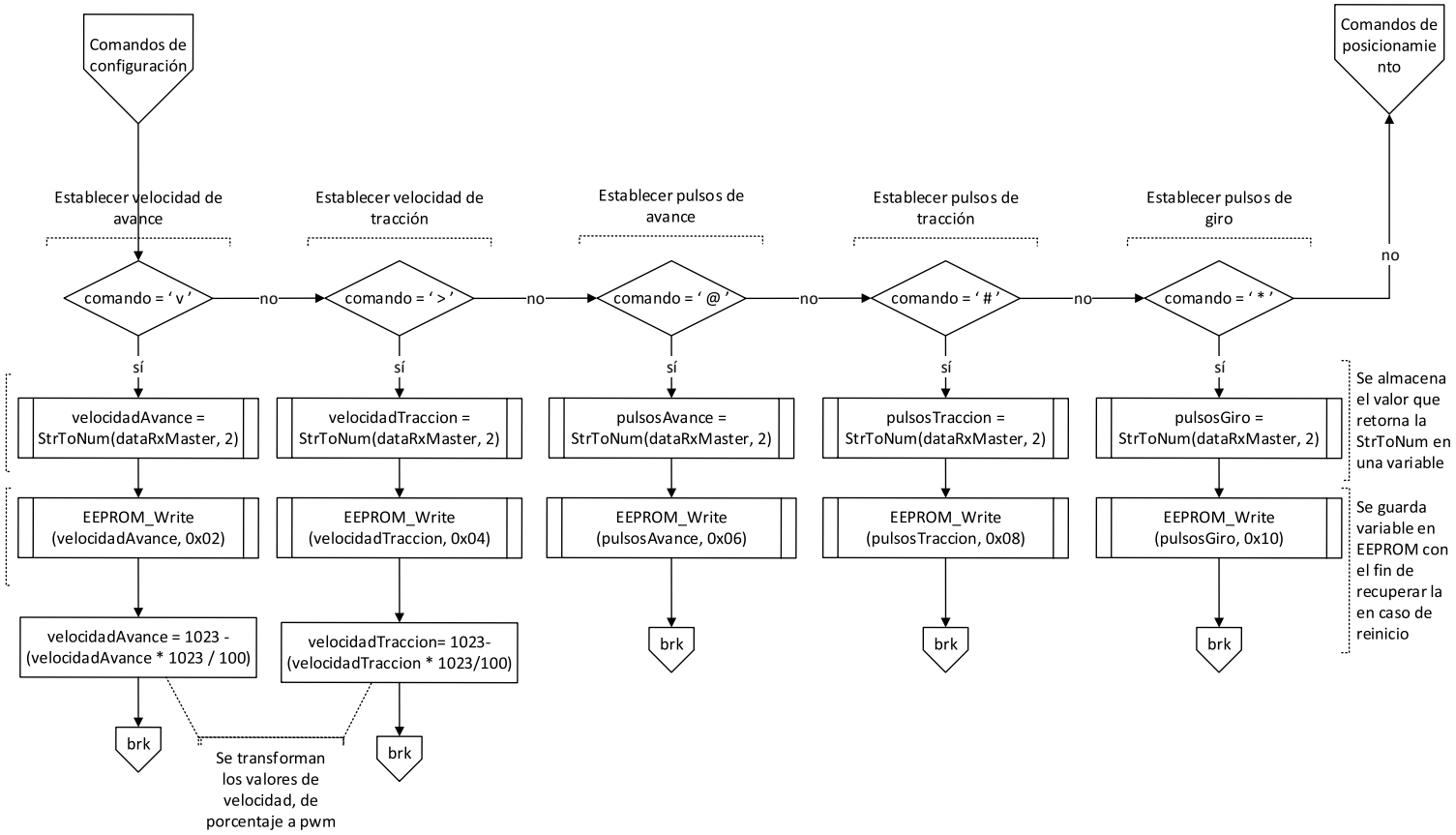


Figura 81.

Elaborado por: Paúl Cueva Castillo

Comandos de posicionamiento sentido horario en función *principal* MCU esclavo

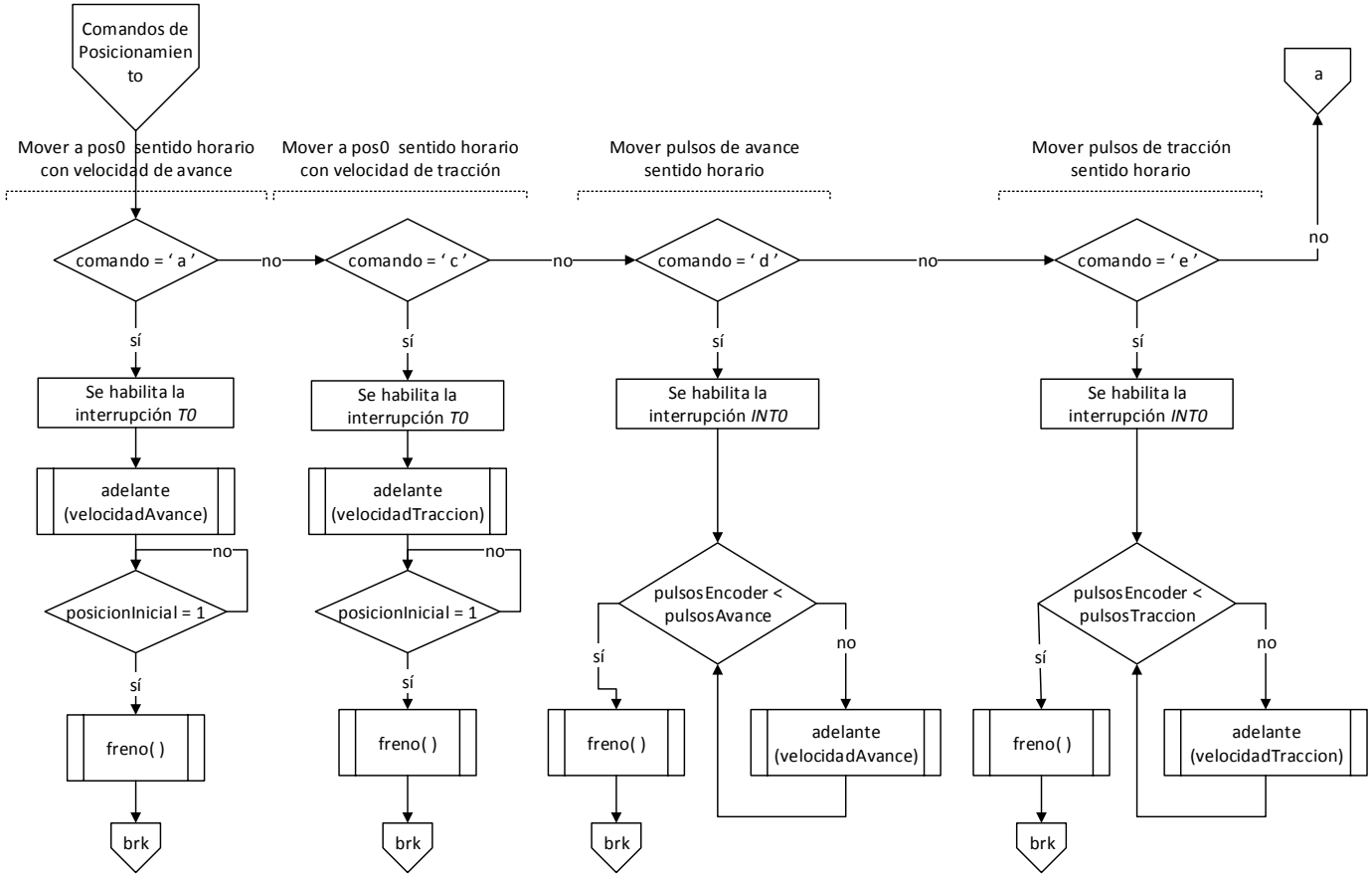


Figura 82.

Elaborado por: Paúl Cueva Castillo

Comandos de posicionamiento sentido anti horario en MCU esclavo

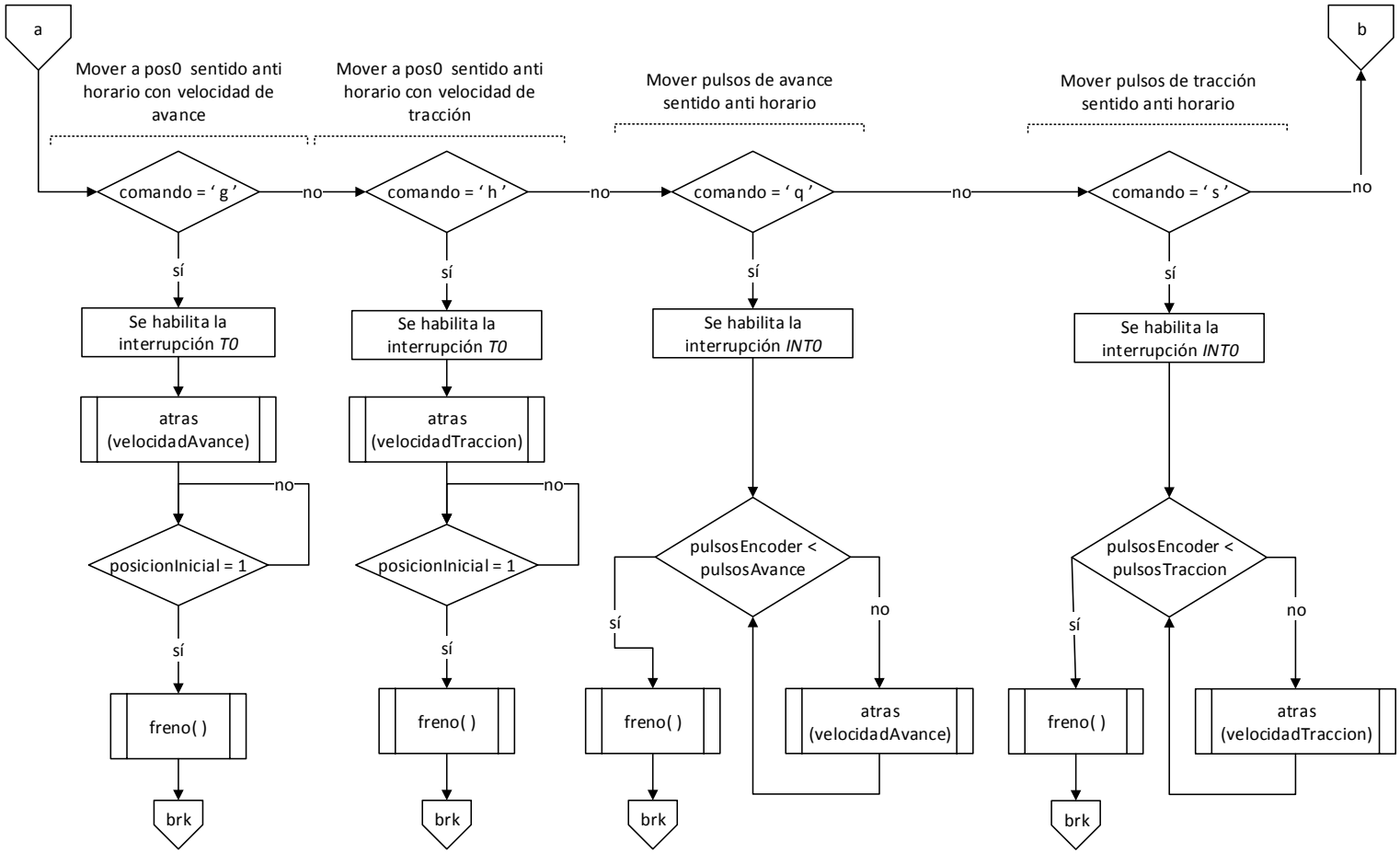


Figura 83.

Elaborado por: Paúl Cueva Castillo

Comandos de posicionamiento para manejo individual de patas en MCU esclavo

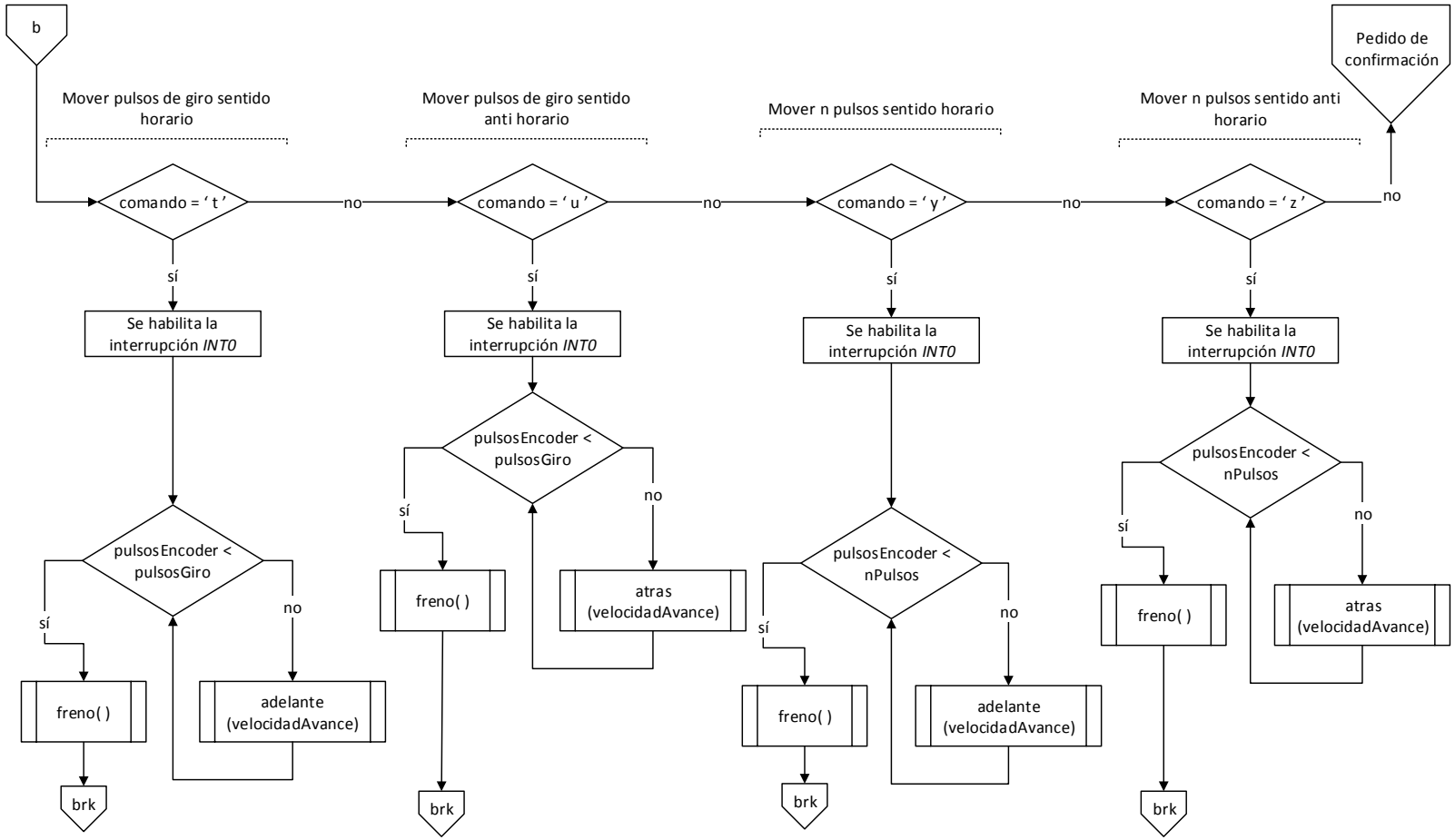


Figura 84.

Elaborado por: Paúl Cueva Castillo

Comandos de posicionamiento para pulsos de giro y n pulsos en el MCU esclavo

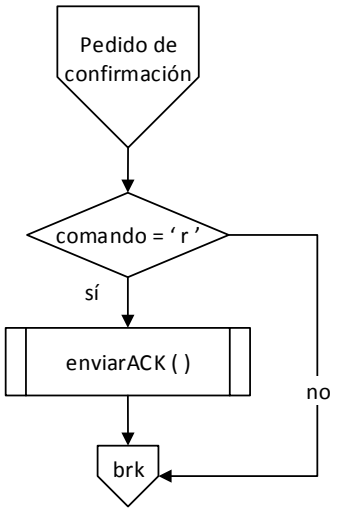


Figura 85.

Elaborado por: Paúl Cueva Castillo

3.4.2.2 Software microcontrolador maestro.

3.4.2.2.1 Definición de variables.

Tabla 9.

Definición de variables de programación en microcontrolador maestro

Nombre	Tipo	Descripción
dataRxGUI[ ]	arreglo de caracteres	Almacena el mensaje proveniente de la interfaz gráfica de usuario (GUI).
dataRxSlave[ ]	arreglo de caracteres	Almacena el mensaje proveniente de un esclavo.
msjeGUIrecibido	bit/bandera	Se activa cuando se recibe un mensaje proveniente de la interfaz gráfica de usuario; se desactiva en la función principal.
msjeSlvRecibido	bit/bandera	Se activa cuando se recibe un mensaje de los esclavos.
sensor	Entero	Almacena el valor de los sensores de temperatura o de inclinación

Nota. Elaborado por: Paúl Cueva Castillo

3.4.2.2.2 Función setup.

Función *setup* en microcontrolador maestro

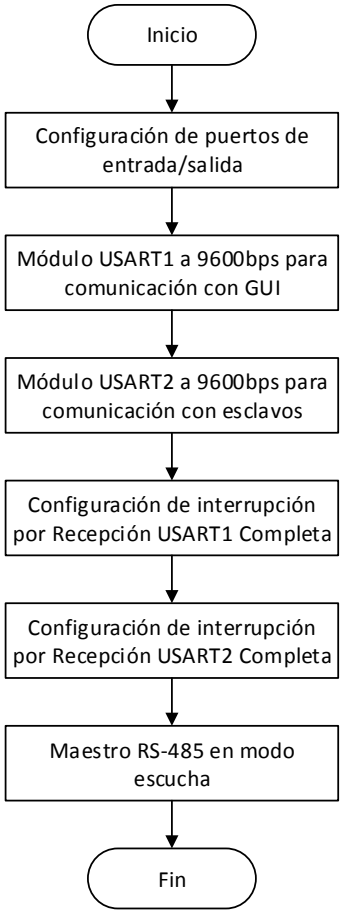


Figura 86.

Elaborado por: Paúl Cueva Castillo

3.4.2.2.3 Funciones por interrupción.

- Función *Recepción USART1 Completa*

Esta interrupción se utiliza para la recepción de datos seriales provenientes de la interfaz gráfica de usuario cuya trama corresponde al *Mensaje para el maestro*. El mensaje se almacena en el arreglo *dataRxGUI[ ]*. Si el mensaje es el carácter 'R', quiere decir que el operador desea resetear el sistema, por lo tanto se envía el comando de reseteo hacia los esclavos y luego se activa el Watchdog propio del MCU maestro para que este se también se resetee. En la figura 87 se observa el diagrama de flujo de esta función:

Función por *Recepción de USART1 Completa* en MCU maestro

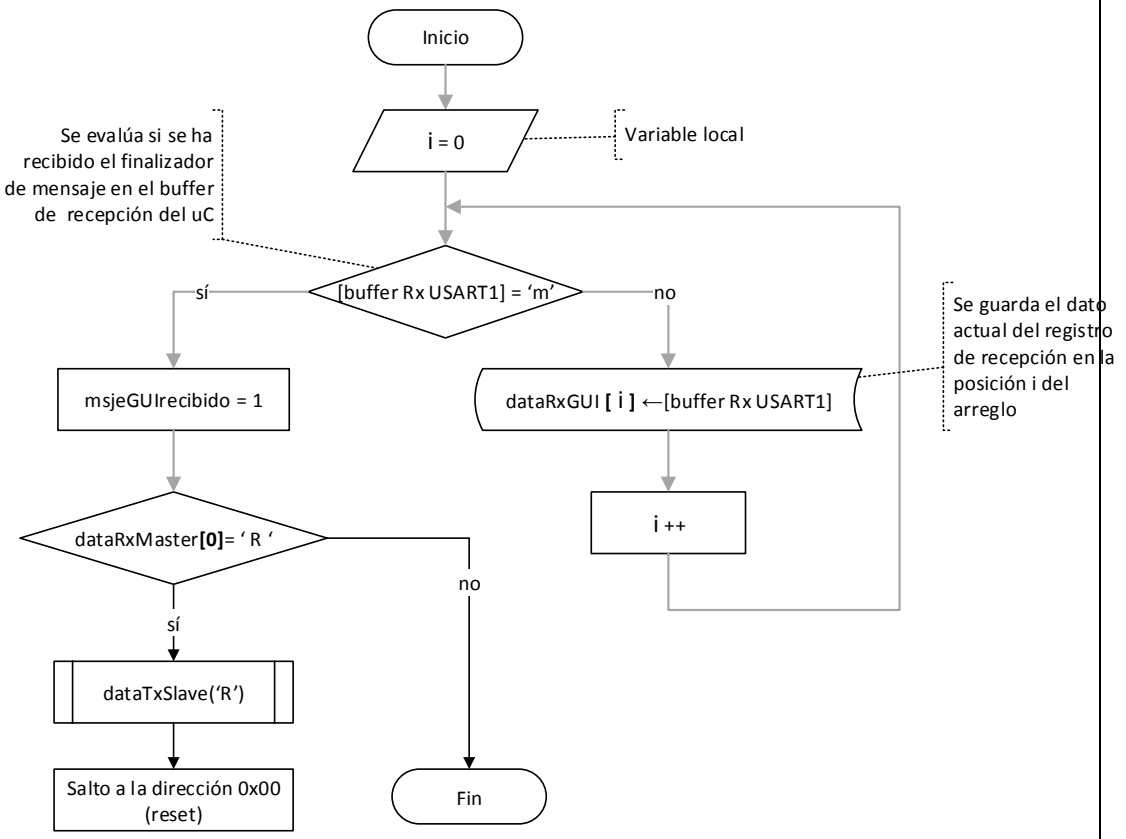


Figura 87.

Elaborado por: Paúl Cueva Castillo

- Función *Recepción USART2 Completa*

Se utiliza para la recepción del mensaje de confirmación proveniente de los esclavos.

Función por *Recepción de USART2 Completa* en el MCU maestro

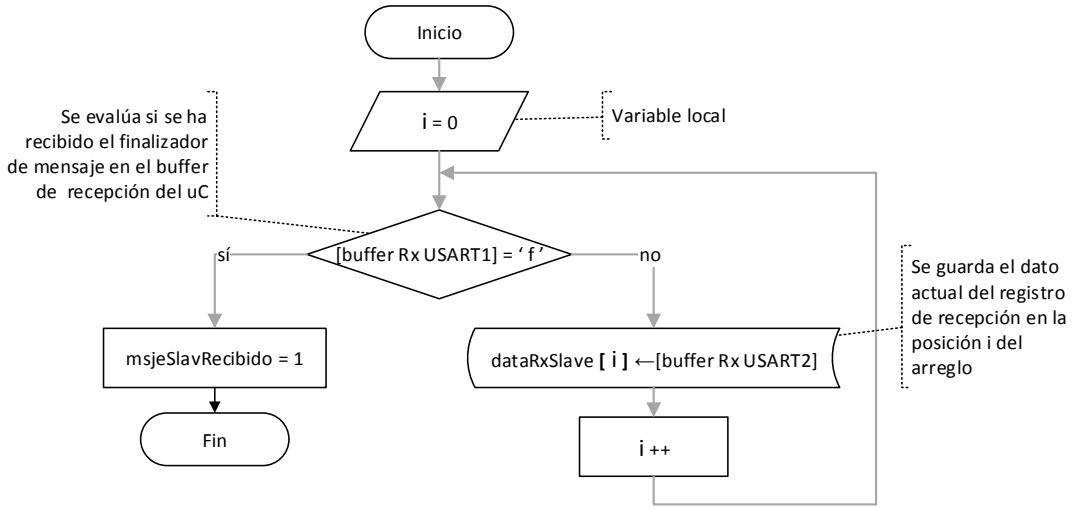


Figura 88.

Elaborado por: Paúl Cueva Castillo

3.4.2.2.4 Funciones auxiliares.

- Función *dataTxSlave*

Esta función se usa para enviar el mensaje a los esclavos, recibe como parámetro el mensaje que se enviará.

Función *dataTxSlave* para envío de mensajes a esclavos

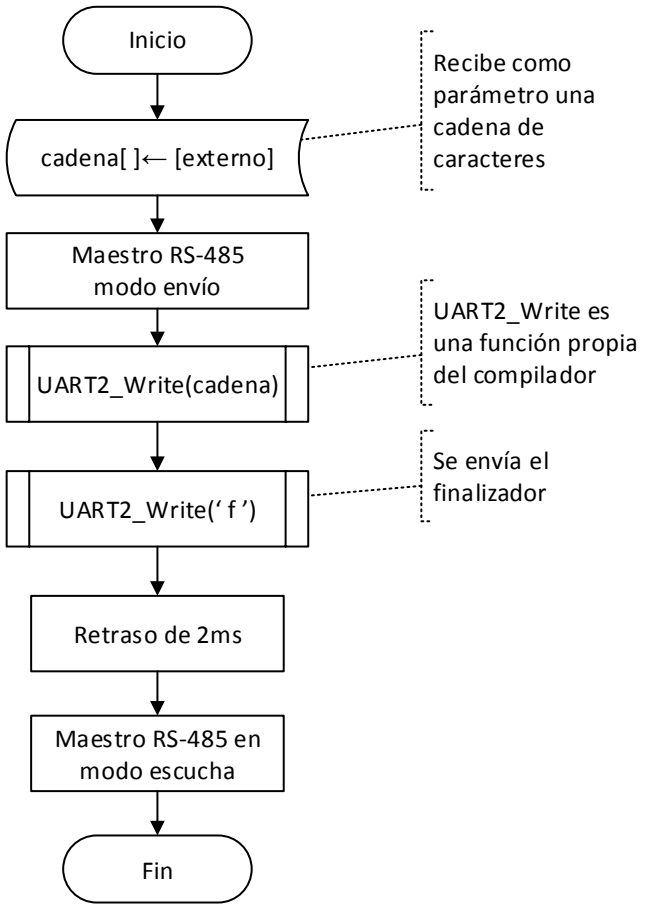


Figura 89.

Elaborado por: Paúl Cueva Castillo

- Función *pedirACKimpar*

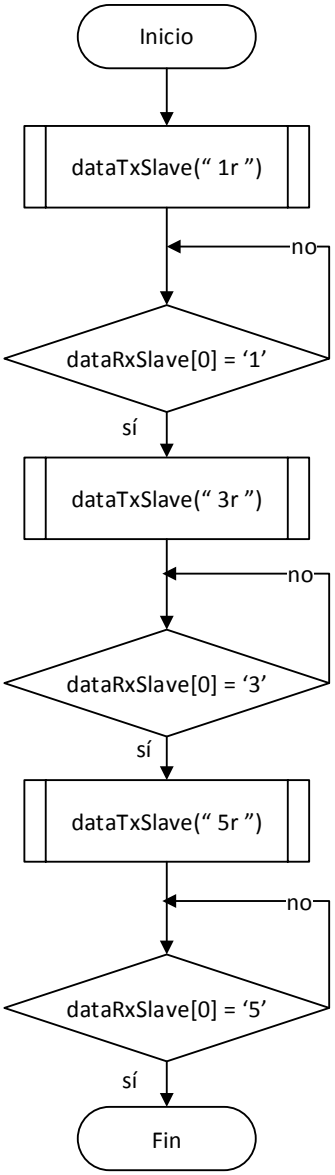
Se usa para enviar el comando de pedido de confirmación a los esclavos impares de uno en uno. La función envía el comando a un esclavo y se mantiene en espera hasta que éste responda, para después hacer lo mismo con el siguiente esclavo. Su diagrama de flujo se observa en la figura 90.a.

- Función *pedirACKpar*

Tiene la misma finalidad que la función *pedirACKimpar* pero dirigida a los esclavos pares. Su diagrama de flujo se observa en la figura 90.b.

Funciones para pedido de confirmación a los esclavos pares e impares

a) Función *pedirACKimpar*



b) Función *pedirACKpar*

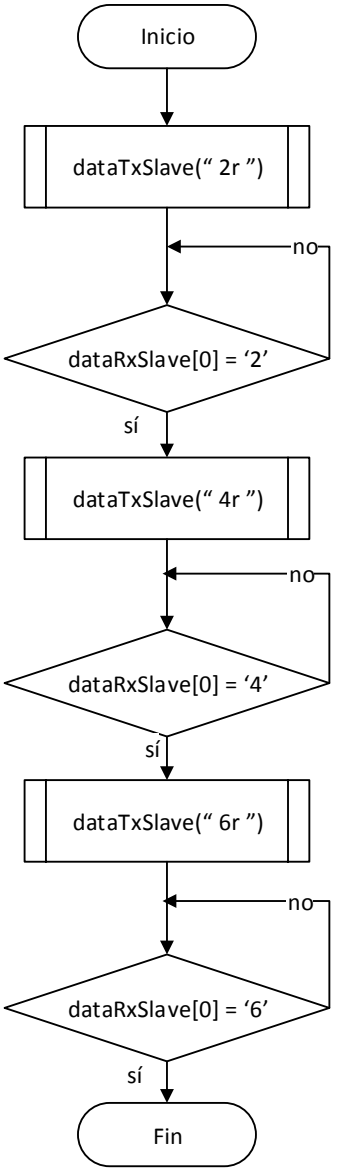


Figura 90.

Elaborado por: Paúl Cueva Castillo

- Función *posicionInicial*

Sirve para situar todas las patas del robot en la posición inicial de tal modo que el robot se ubica en la postura inicial. El diagrama de flujo de la función y la postura inicial del robot se observan en la figura 91.

Función *posicionInicial* para situar al robot en la postura inicial

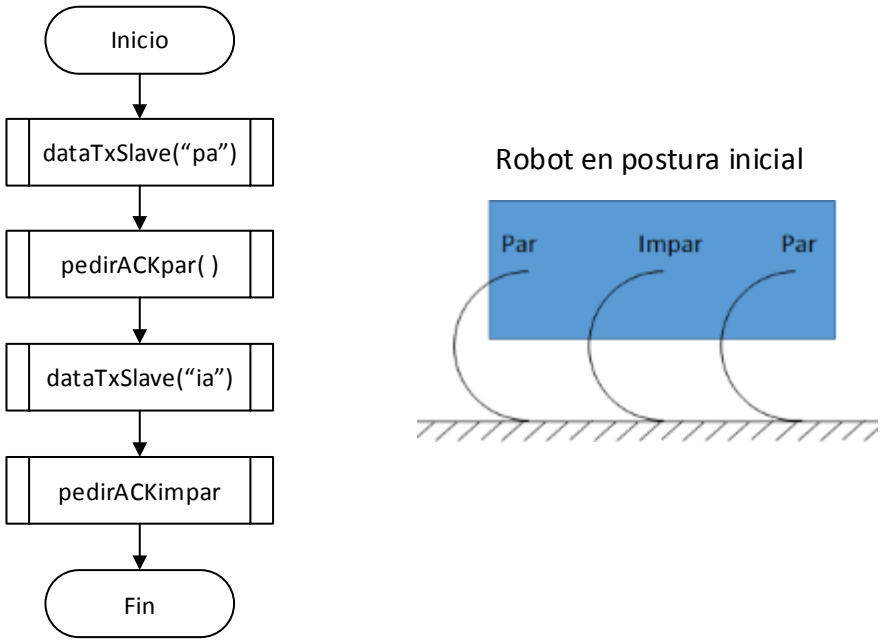


Figura 91.

Elaborado por: Paúl Cueva Castillo

3.4.2.2.5 Función *principal.*

Se encarga de reconocer si el mensaje que recibe es un Mensaje para el Maestro o un Mensaje para el Esclavo evaluando si la trama recibida contiene o no caracteres de letras mayúsculas. En caso afirmativo, se ejecuta una acción correspondiente a las tareas del MCU maestro; caso contrario, el mensaje se envía íntegramente a los esclavos.

En esta función se implementan los algoritmos para las secuencias de movimiento adelante y giros del robot; debido a eso, para una mejor comprensión, en los diagramas de flujo correspondientes (figuras 94, 96 y 97) se ha situado a lado derecho el gráfico los movimientos que realizarán las patas a medida que se va implementado el código de dichos algoritmos.

Función *principal* en microcontrolador maestro

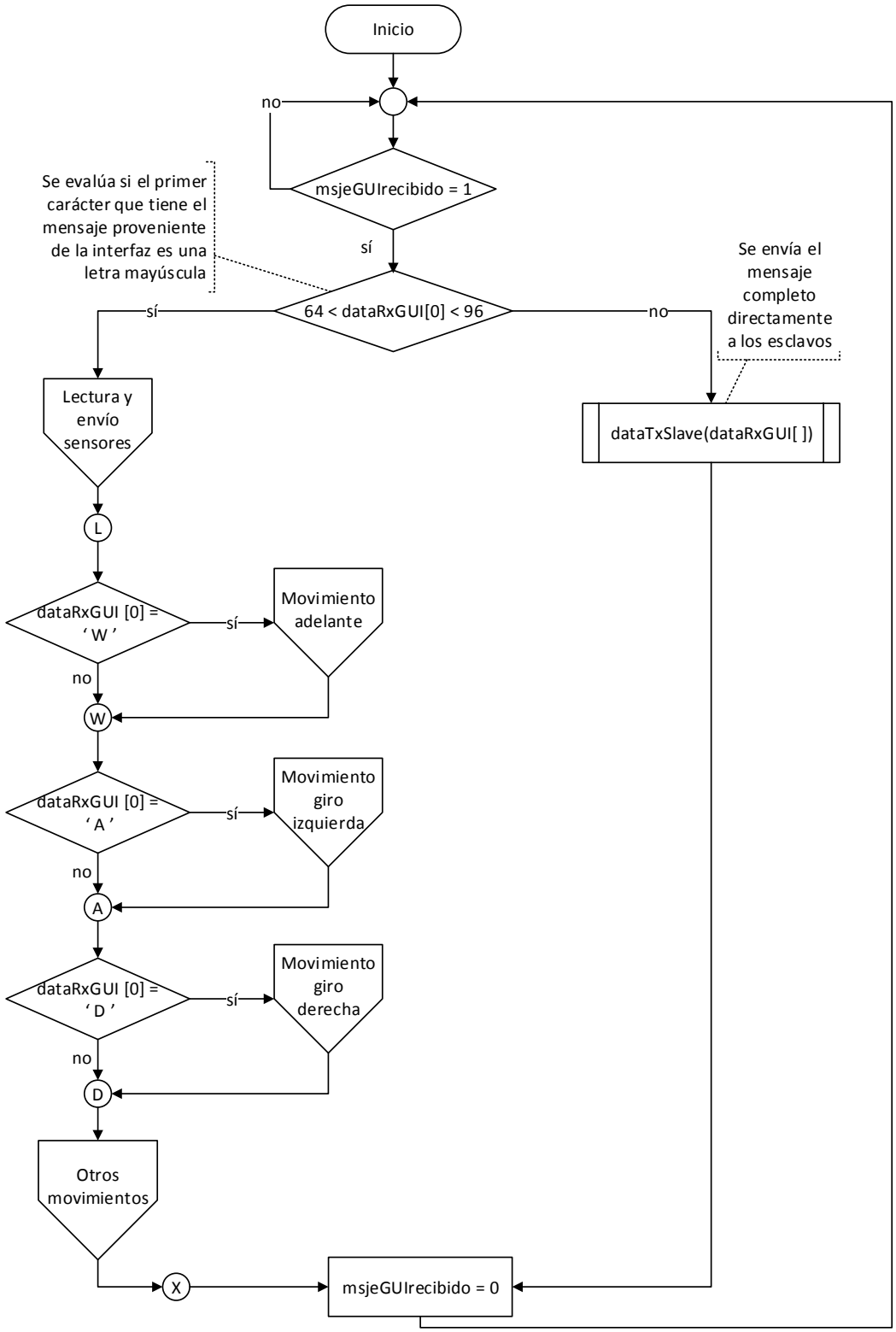


Figura 92.

Elaborado por: Paúl Cueva Castillo

Lectura y envío del valor de los sensores de temperatura e inclinación a la GUI

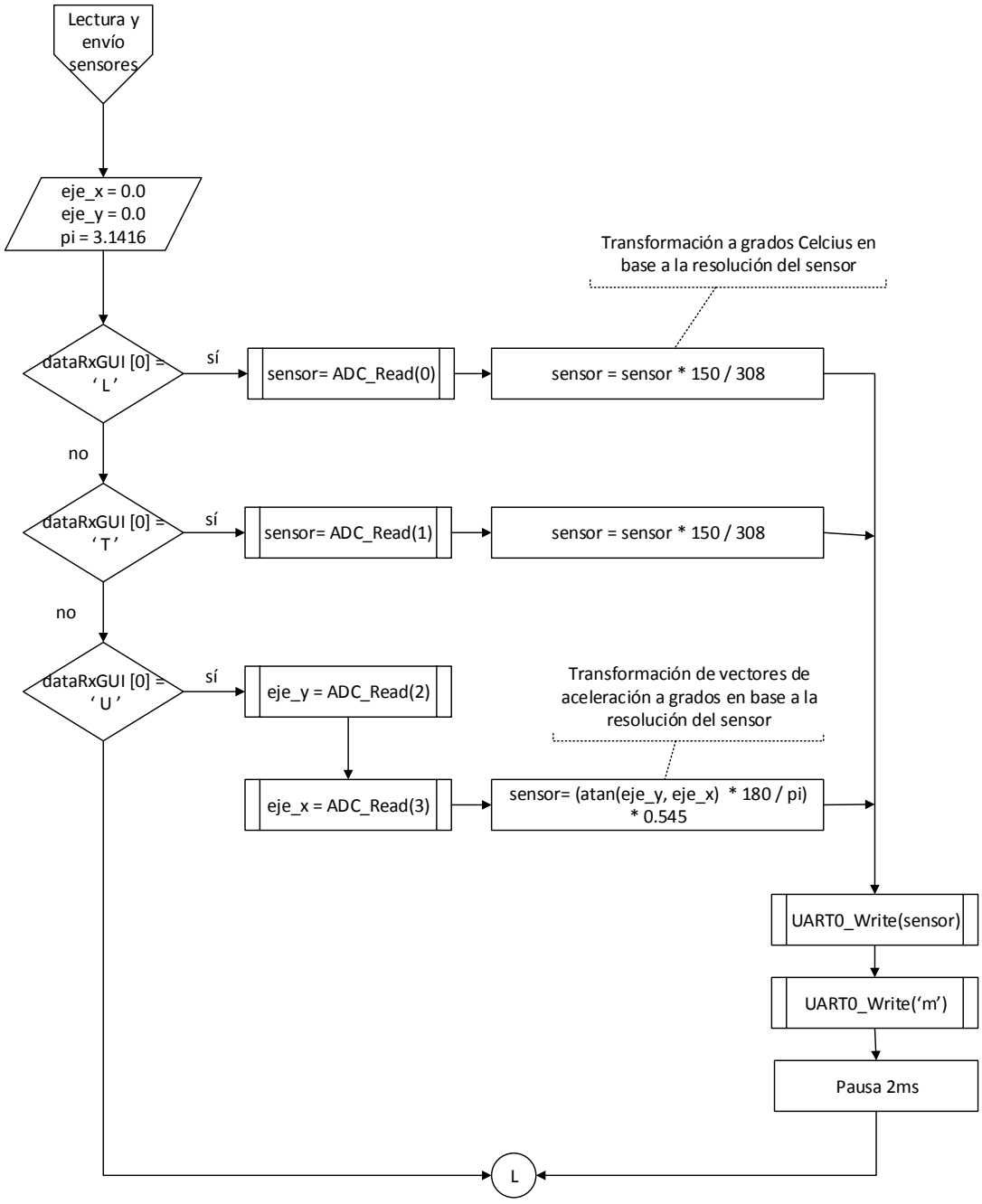


Figura 93.

Elaborado por: Paúl Cueva Castillo

La obtención de las fórmulas para hallar la temperatura en grados Celsius y la inclinación en grados se observan en el anexo 3.

Secuencia de comandos movimiento adelante en la función principal del MCU maestro

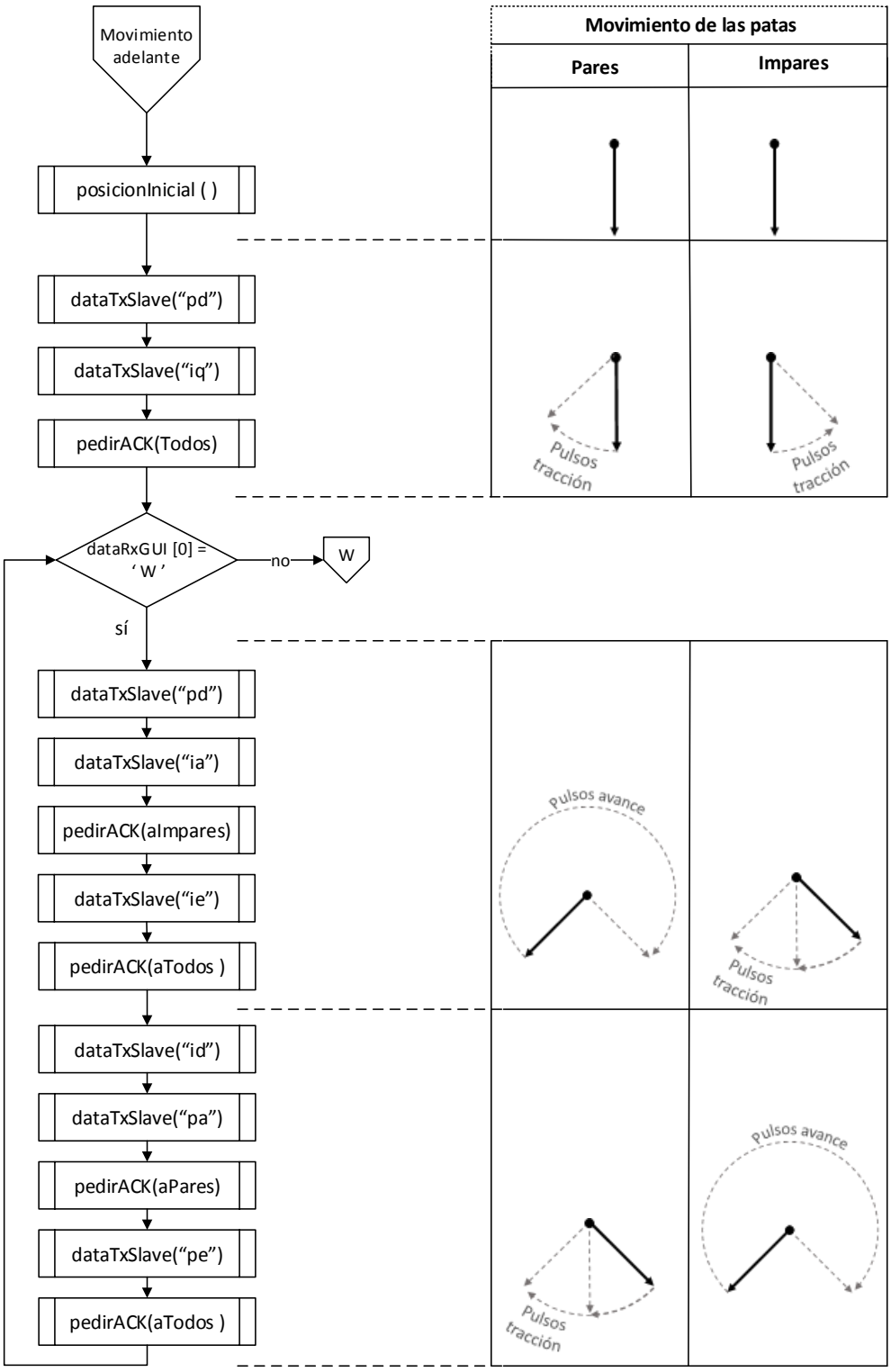


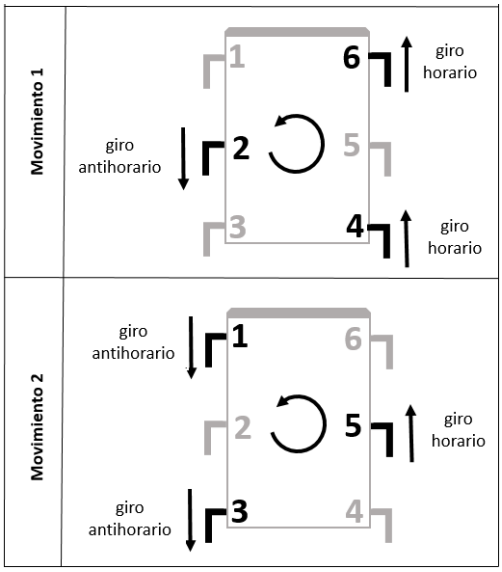
Figura 94.

Elaborado por: Paúl Cueva Castillo

Para llevar a cabo las rutinas de giro ya sea a la izquierda o a la derecha, las patas ejecutan dos secuencias distintas de movimientos que permiten que el robot gire sobre su propio eje. En la figura 93 se pueden observar dichas secuencias.

Secuencias de movimiento de las patas para rutinas de giro

a) Giro izquierda



b) Giro derecha

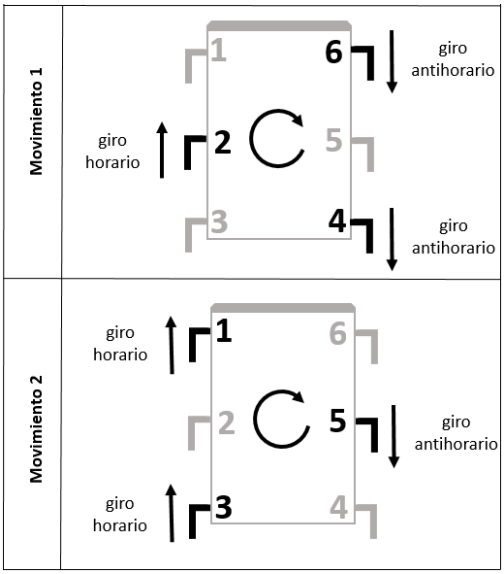


Figura 95.

Elaborado por: Paúl Cueva Castillo

Secuencia de comandos para movimiento giro izquierda en MCU maestro

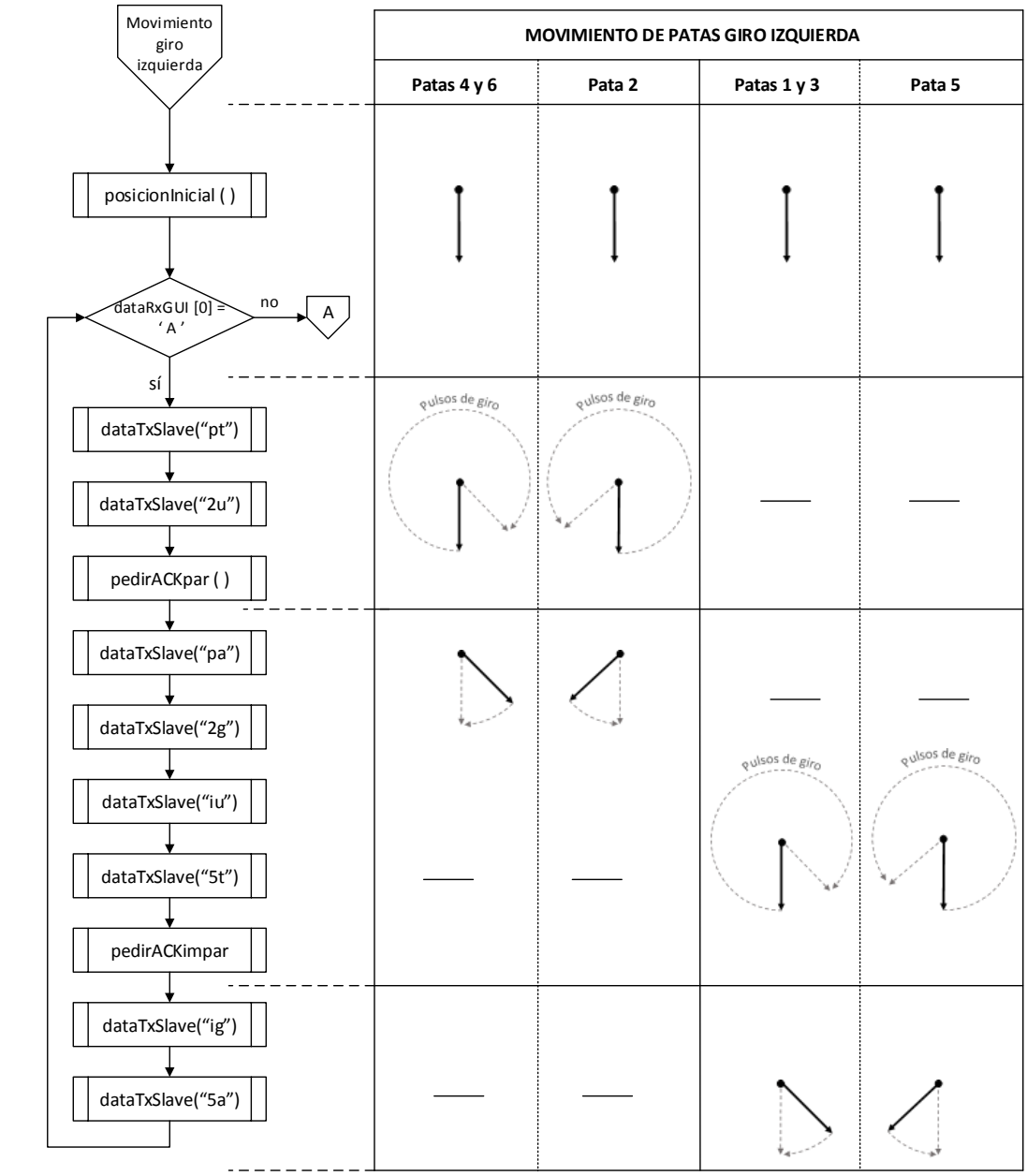


Figura 96.

Elaborado por: Paúl Cueva Castillo

Secuencia de comandos para movimiento giro derecha en MCU maestro

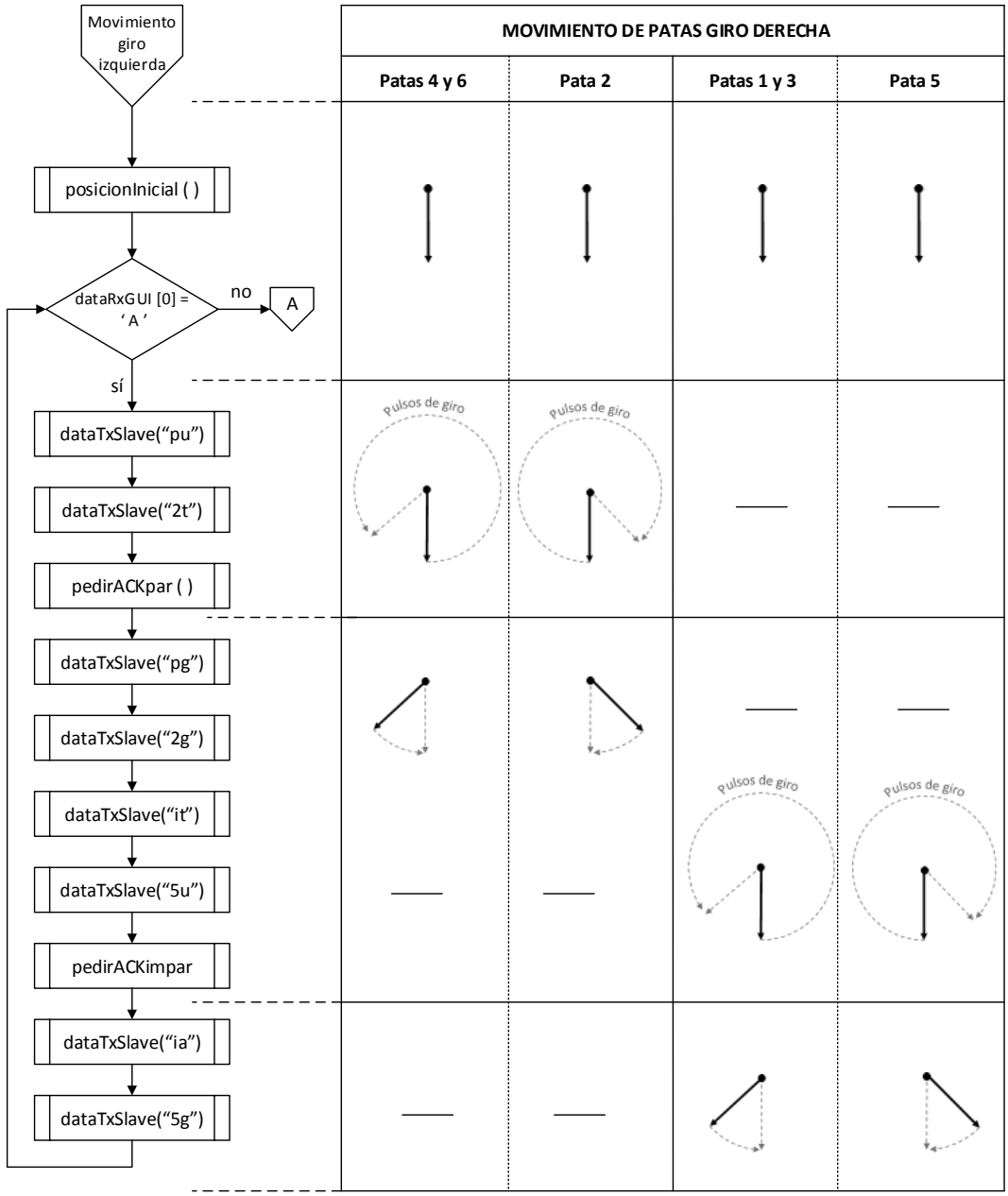


Figura 97.

Elaborado por: Paúl Cueva Castillo

Secuencia de comandos para detener, levantar y acostar al robot

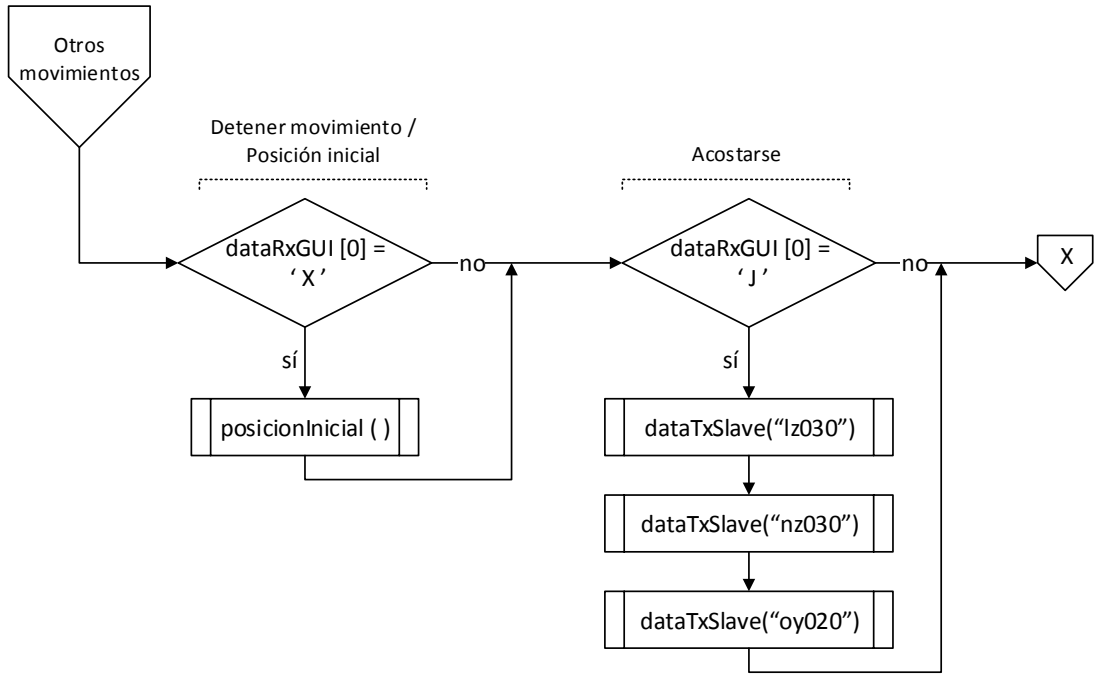


Figura 98.

Elaborado por: Paúl Cueva Castillo

3.4.2.3 Interfaz gráfica de usuario (GUI).

3.4.2.3.1 Comunicación entre Matlab y el microcontrolador maestro.

Para establecer el enlace de comunicación entre la interfaz y el MCU maestro es necesario crear un objeto serial en Matlab, el cual se asocia a su vez a un puerto de comunicación serial (puerto COM) del computador, y permite establecer parámetros tales como, número de puerto, velocidad, bit de paridad, etc.

Debido a que este proceso se debe realizar cada vez que se envíe y se reciba un dato (se presione un botón), se lo implementó en una función denominada *miObjSerial*. Su diagrama de flujo se muestra en la figura 99.

Función *miObjSerial* para crear un objeto serial para enviar y recibir datos en la GUI

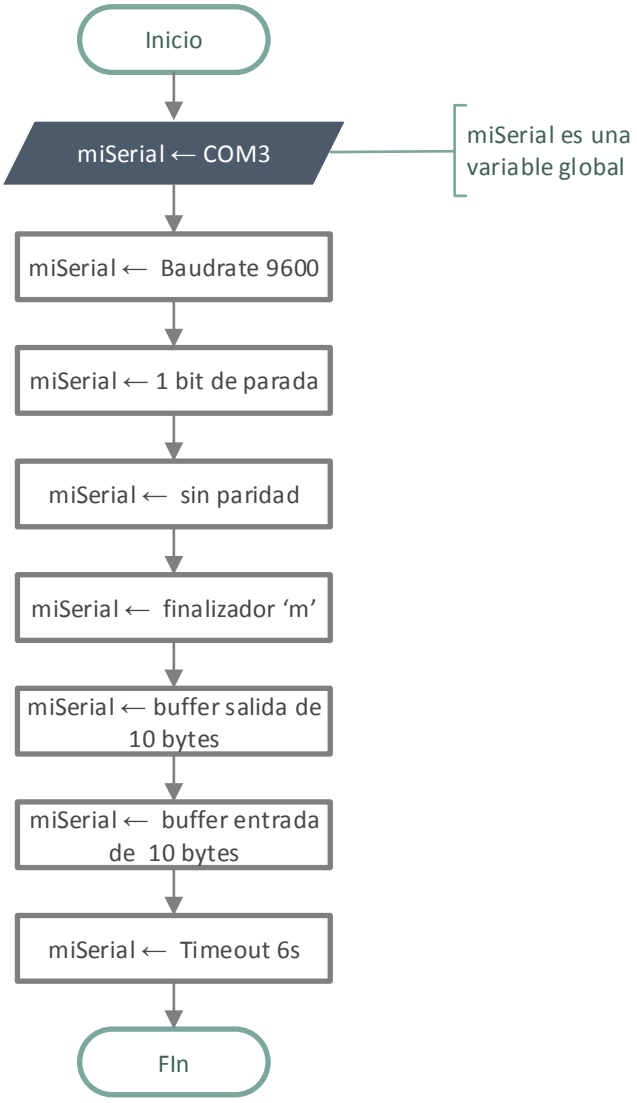


Figura 99.

Elaborado por: Paúl Cueva Castillo

Mediante el objeto serial es posible realizar las tareas de escritura, lectura, entre otras, en el puerto COM asociado. Para dichas tareas Matlab cuenta con sus propios comandos; en la tabla 10 se detallan los utilizados, y debido a que Matlab tiene muchas formas de uso para un mismo comando, se los definió según el propósito que concierne.

Tabla 10

Comandos de Matlab usados en el manejo del puerto serial

Función	Descripción
fopen(objeto)	Permite abrir el puerto serial.
fprintf(objeto, format, datos)	Permite escribir datos en el puerto especificando el nombre del objeto, el formato de los datos, y los datos
fscanf(objeto, format)	Permite leer datos del puerto especificando el nombre del objeto, y el formato de los datos.
fclose(objeto)	Permite cerrar el puerto de comunicación serial.
delete(objeto)	Permite eliminar el objeto serial.

Nota. Elaborado por: Paúl Cueva Castillo

La interfaz gráfica de usuario se compone de cuatro secciones: sección para enviar comandos de movimiento; sección para enviar comandos al esclavo; sección para visualizar datos de los sensores; y sección para visualizar la imagen de video. En la figura 100 se muestra la imagen de la interfaz junto con las distintas secciones que la componen, y a continuación se expone cada una de ellas.

Interfaz gráfica de usuario

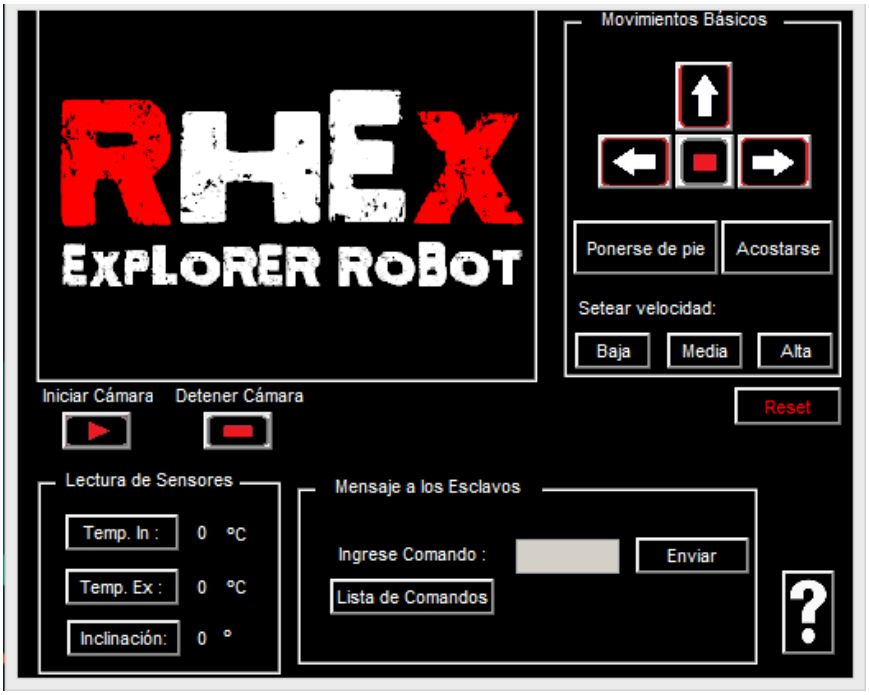


Figura 100.
Elaborado por: Paúl Cueva Castillo

3.4.2.3.2 Sección para enviar comandos de movimiento.

Está compuesta por botones que tienen la función de enviar, cada uno, los distintos comandos del *Mensaje para el Maestro*, los cuales se vieron anteriormente en la tabla 5: adelante, giro izquierda, giro derecha, acostarse, pedido de temperatura, etc. El diagrama de flujo del proceso que sucede al presionar uno de estos botones se muestra en la figura 101.

Proceso al presionar botón de la sección enviar comandos de movimiento en la GUI

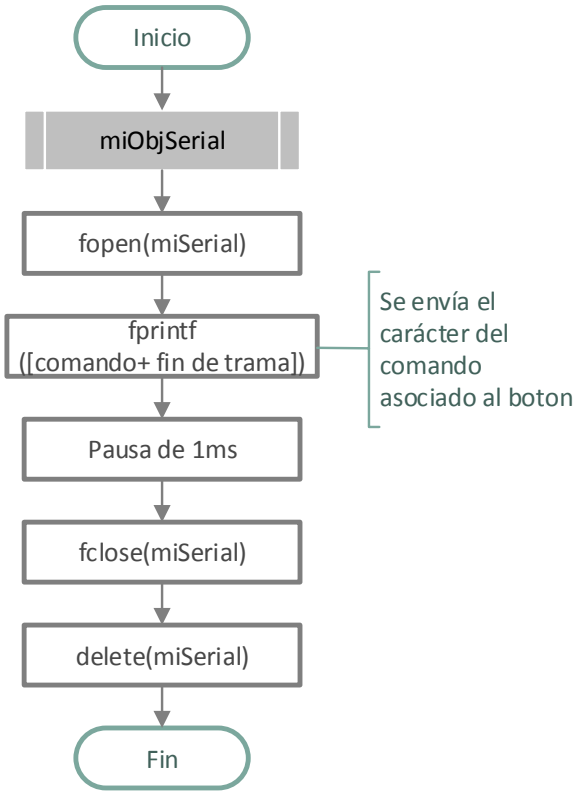


Figura 101.

Elaborado por: Paúl Cueva Castillo

3.4.2.3.3 Sección para enviar comandos para los esclavos.

Está compuesta por un elemento de ingreso de texto (edit text en Matlab) que permite escribir directamente los comandos del *Mensaje para el Esclavo*; por un botón de envío para dicha trama; compuesta además por un botón de ayuda el cual muestra la lista de los comandos y el carácter que les corresponde (tabla 7 en este documento).

Esta sección se usa principalmente para enviar los comandos de configuración, y en caso de ser necesario, los de posicionamiento de patas; ambos con la ventaja de que el *Mensaje para el Esclavo* permite al operador tener acceso a cualquiera de las patas al especificar el ID en la trama; por ejemplo se puede configurar a una determinada pata con distinta velocidad que las demás para realizar las rutinas de movimiento, función de mucha utilidad al momento de realizar las pruebas de campo. El diagrama de flujo del proceso que sucede al presionar el botón *enviar* se muestra en la fig. 102.

Presionar botón *enviar* de la sección comandos para los esclavos en la GUI

Figura 102.

Elaborado por: Paúl Cueva Castillo

3.4.2.3.3 Sección para visualizar datos de sensores.

Está compuesta por botones para adquirir la lectura de los distintos sensores (temperatura interna, temperatura externa e inclinación); y por los elementos conocidos como *static text* en Matlab los cuales permiten mostrar el valor de dichas lecturas. El diagrama de flujo del proceso que sucede al presionar uno de estos botones se muestra en la figura 103.

Proceso presionar botón de pedido de lectura de sensores en la visualización de datos en la GUI

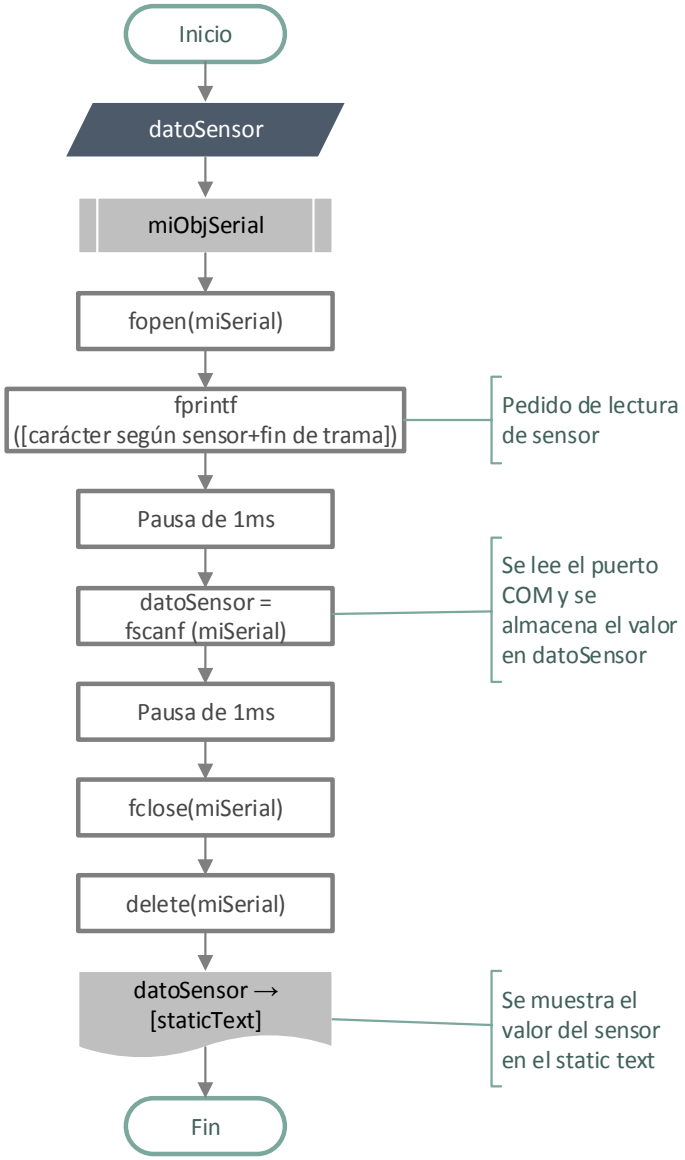


Figura 103.

Elaborado por: Paúl Cueva Castillo

3.4.2.3.4 Sección para visualizar el video.

Contiene el área para mostrar el video transmitido por la cámara. Dicha área está ajustada a la resolución de la propia cámara de video que es de 160x120 píxeles.

Sobre el video se aplica el algoritmo de reconocimiento de personas. La sección contiene además dos botones para detener y/o continuar la transmisión; si se detiene también se detiene la ejecución del algoritmo de reconocimiento de personas, lo que permite al operador disminuir la carga de procesamiento en caso de ser necesario.

3.4.2.3.5 Reconocimiento de personas.

Debido a que el procesamiento digital de imágenes no es el tema de investigación principal en el proyecto, a continuación se presenta una explicación elemental de conceptos referentes a dicho procesamiento manejados en el diseño del algoritmo de reconocimiento de personas.

a. **Espacios de color.**

   Son modelos matemáticos que permiten especificar, y manipular colores. El número de dimensiones del espacio depende del modelo que se utilice. Los colores se representan mediante puntos coordenados en estos espacios. Los modelos manejados en el diseño del algoritmo son el RGB y el HSV.

b. **Modelo RGB.**

   Está formado por tres planos de imagen independientes, donde cada uno corresponde a un color primario (rojo, verde, azul); para producir una imagen de color compuesta los tres planos se combinan en la pantalla (Francisco, 2002).

c. **Modelo HSV (Hue Saturation Value).**

   Permite reconocer un color en base al matiz (Hue), la saturación (Saturation) y el valor o brillo (Value). El matiz indica el color al que se refiere (rojo, verde, naranja, etc.); la saturación indica la cantidad de blanco que tiene dicho

color, siendo un color puro o completamente saturado el que no contiene blanco; y la componente brillo, es el grado de luminosidad del color (Noelia Sanjurju Dovigo, 2012).

d. Procesamiento morfológico de imágenes.

Técnica no lineal de tratamiento de las estructuras geométricas de una imagen basada en la teoría de conjuntos. Permite realizar operaciones de segmentación, restauración, detección de bordes, esqueletos, aumento de contraste, análisis de texturas, compresión, etc. El filtrado morfológico permite la atenuación de ruido y la extracción selectiva de estructuras y objetos en la imagen (Francisco, 2002).

El algoritmo está diseñado para detectar el color de piel de las personas; de tal forma que durante una búsqueda de rescate, el software será capaz de identificar a una persona al visualizarse cualquier parte de su cuerpo en la pantalla de la interfaz.

Debido a que el color de piel varía de persona a persona, el algoritmo reconoce distintos tonos de color naranja entre claros y oscuros.

Para reconocer el color de piel se realiza un tratamiento morfológico de la imagen, el cual permite resaltar las zonas convenientes en función de un rango de valores de color dados. Tal proceso se realiza en el espacio HSV para evitar medidas erróneas debidas a variaciones de iluminación del ambiente.

Como es muy probable que se detecten varias regiones de piel, para evitar confundir al operador se selecciona la región con mayor número de píxeles para realizar el seguimiento.

En la figura 104 se muestra el diagrama de bloques del algoritmo, y en la figura 105 el funcionamiento del mismo paso a paso.

Algoritmo de seguimiento de color de piel de las personas

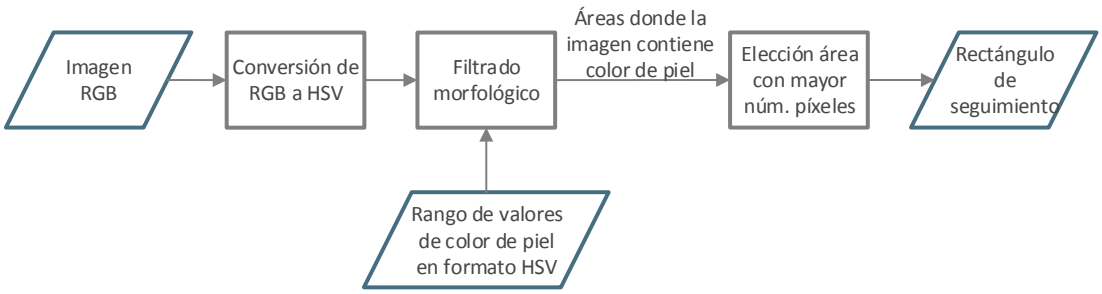


Figura 104.

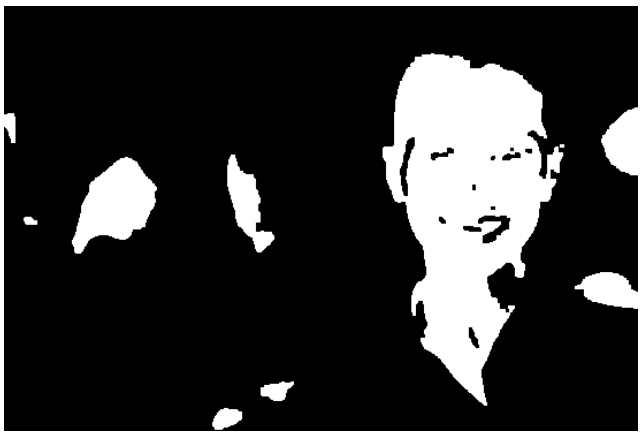
Elaborado por: Paúl Cueva Castillo

Funcionamiento del algoritmo de seguimiento de piel paso a paso.

a) Imagen original en modelo RGB obtenida de la cámara de video

b) Imagen convertida a modelo HSV

c) Imagen una vez realizado el filtrado morfológico



Se observa que el filtrado morfológico detecta varias regiones con color de piel.

d) Rectángulo de seguimiento sobre el área con mayor número de píxeles

e) Rectángulo de seguimiento sobre la imagen original

Figura 105.

Elaborado por: Paúl Cueva Castillo

CAPÍTULO 4
ANÁLISIS DE RESULTADOS

En este capítulo se explican las pruebas realizadas durante la construcción del robot así como las pruebas finales ejecutadas en campo. Luego de tener listo el hardware mecánico, con el hardware electrónico y el software, se procedió hacer pruebas de cada etapa de control del robot para verificar el buen funcionamiento de cada una de ellas, para ir corrigiendo errores en el transcurso del montaje, al finalizar la etapa de pruebas individual de cada módulo se procedió a las pruebas de campo y análisis de resultados.

4.1 Pruebas del hardware mecánico

Tomando como referencia un diseño original del robot Rhex, se hizo el primer diseño tal cual el robot original investigado, sin embargo, se tuvo que modificar las medidas y el aspecto final del robot. Con las modificaciones pasó de tener una forma redonda en la parte frontal y posterior de la estructura, a un aspecto rectangular, debido a que las patas no sobresalían lo suficiente como para subir gradas o pasar por sobre superficies irregulares. Se modificó su forma redondeada como también se aumentó el diámetro de las patas.

4.1.1 Chasis

Como se observa en la figura 106, el primer diseño era más pequeño que el diseño final, al no encontrar motorreductores pequeños con bastante torque, se tuvo que diseñar la estructura final en base a los motores que se encontró en el mercado.

Primer diseño vs diseño final

Figura 106.

Capturadas por: Heinerth G. Romero M

Luego de varias pruebas se decidió recortar los extremos dejando al robot con un aspecto rectangular. En las figuras 107 y 108 se muestra la evolución tanto del chasis como el de las patas.

Primer chasis y juego de patas

Figura 107.

Elaborado por: Heinerth G. Romero M

Chasis y juego de patas finales

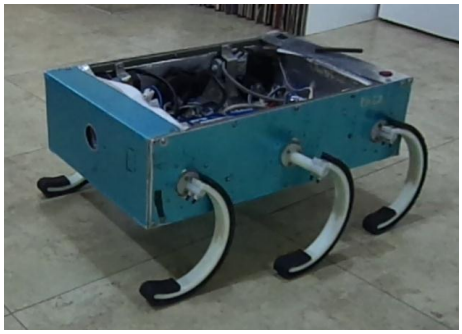


Figura 108.

Elaborado por: Heinerth G. Romero M

4.1.2 Patas

El primer juego de patas que se construyó fue de policloruro de vinilo o comúnmente llamado PVC, y tenían un diámetro de seis pulgadas. Este material es muy resistente, empieza a deformarse a los 80°C y se descompone a 140°C sin embargo no se consiguió un tubo con el diámetro deseado por lo que se descartó al tubo de PVC como material para la construcción de las patas.

El robot al ser todo terreno, tiene que poseer en sus puntos de apoyo sobre la superficie, un material que lo adhiera o que ejerza fricción sobre la misma. Las pruebas hechas con el primer juego de patas permitió observar que el desplazamiento lineal del robot era muy poco. Razón por la cual se aumentó el diámetro de las patas y se recubrió estas con un caucho especial, lo que permitió tener mayor coeficiente de fricción y por ende mayor desplazamiento.

Evolución de las patas

Figura 109.
Elaborado por: Heinerth G. Romero M

Otro problema que se corrigió en la etapa de pruebas fue el sistema de sujeción entre las patas y el eje del motor, el sistema de sujeción consta únicamente de dos tornillos Allen avellanados con tuercas de seguridad. Al inicio se utilizó tornillos y tuercas comunes, pero estos con el transcurso del tiempo por el movimiento y la fuerza que

tenían que soportar, se aflojaban, lo que hacía que las patas pierdan su alineación causando una desviación en la trayectoria del robot.

Pernos y tuercas utilizados

Figura 110.

Fuente: Palimpalem.

Como se observa en la figura 110, este tipo de tornillos poseen una parte cónica en su cabeza por lo que en el agujero de las patas tocó avellanar para que entre la cabeza de manera correcta y no permita que el perno se mueva y se afloje, a diferencia de los tornillos usados al inicio que tenían únicamente dos superficies de contacto para ajustar, los allen utilizados tienen seis superficies de contacto para que el ajuste sea más fuerte y las patas queden bien fijas al eje del motor. Por otro lado las tuercas de seguridad se llaman así porque poseen un recubrimiento de caucho como seguridad adicional en la parte interna de la rosca, para evitar que se aflojen.

4.2 Prueba encoders

Luego de la construcción de los encoders, se procedió a realizar las mediciones de las señales que estaban generando, con la ayuda de un osciloscopio se tomó las muestras de los seis encoders que se fabricó.

Lectura de señales de los encoders

Figura 111.

Capturadas por: Paul Cueva Castillo.

Este proceso de pruebas del encoder es muy importante ya que de esto dependerá la sincronización de las patas del robot, los primeros encoders que se fabricó no generaron ningún tipo de señal ya que el tipo de papel donde se imprimió los discos ranurados no reflejaban la luz infrarroja de los sensores, como se mencionó anteriormente en el apartado 3.3 se cambió el tipo de papel blanco por un papel especial de aluminio, con esto se logró generar los pulsos que se muestran en la figura 111.

Separación máxima del sensor del encoder

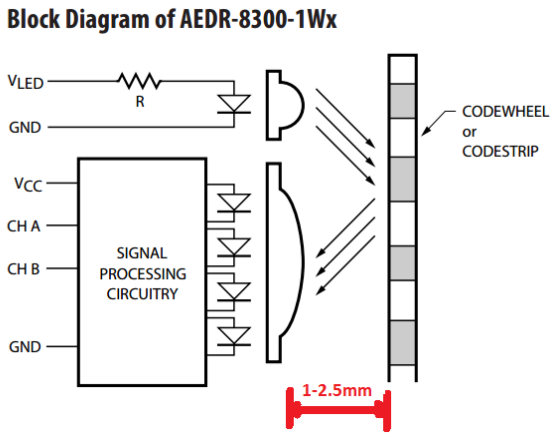


Figura 112.

Fuente: **A**vagotech.

Otro factor que se corrigió fue la distancia de separación entre el sensor y el disco ranurado, en el datasheet del fabricante se recomienda que los sensores deben estar a una distancia mínima de 1 milímetro y máxima de 2.5 milímetros. Si los sensores se encuentran a más de 2.5 milímetros de separación entre la superficie que va a reflejar la luz, la señal que genere el encoder puede verse afectada y no dará una señal confiable, por tal motivo se hizo pruebas sensor por sensor y en algunos motores se tuvo que mover el encoder ya que los pulsos generados se distorsionaban debido a la separación del sensor.

4.3 Pruebas de las tarjetas de control

Las pruebas que se realizaron a las tarjetas de control se han dividido en:

- Comunicación entre la tarjeta maestro y el computador
- Comunicación entre la tarjeta maestro y las tarjetas esclavos

4.3.1 Comunicación inalámbrica entre la tarjeta maestro y el computador

Para las pruebas de comunicación inalámbrica entre la tarjeta master y el computador se configuró los módulos X-bee, para esto se utilizó la herramienta de programación X-CTU que se la puede descargar gratis de la página del fabricante.

Configuración módulos X-bee

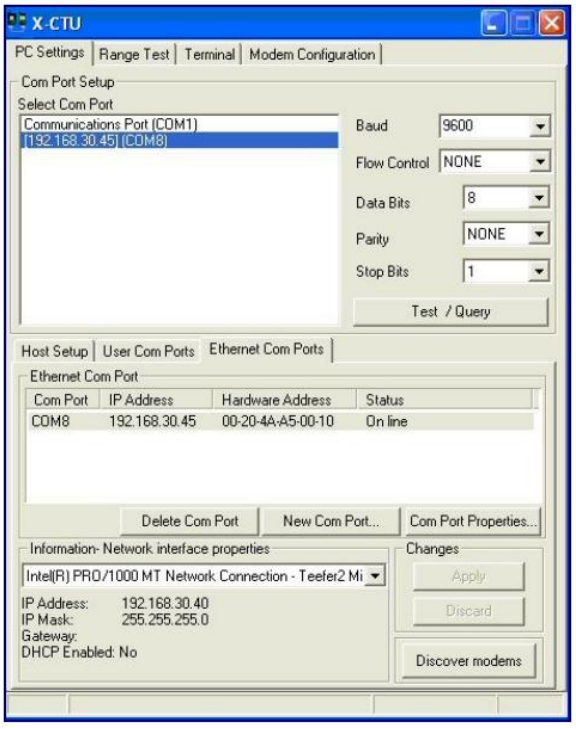


Figura 113.

Elaborado por: Heinerth Romero Macas

Estos módulo vienen configurados de fábrica sin embargo es mejor asegurase que ambos dispositivos estén trabajando a la misma velocidad, también X-CTU permite

dar nombre a los dispositivos para evitar que más módulos intenten comunicarse con los dos de este proyecto.

Luego de configurar y dar nombre a los dos módulos se procede a la prueba de comunicación inalámbrica entre los dos, y esto se lo hace con la misma herramienta X-CTU.

Prueba de comunicación entre los módulos X-bee

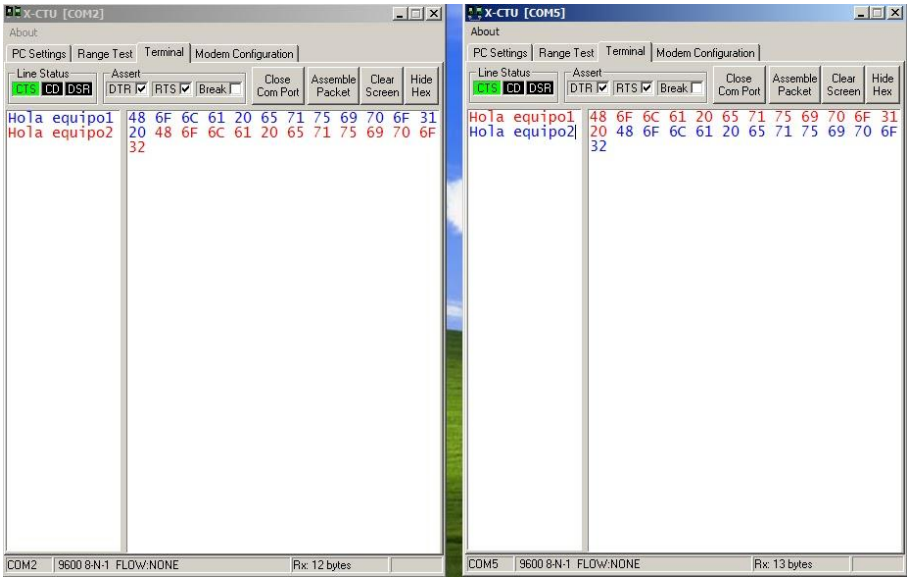


Figura 114.

Elaborado por: Heinerth G. Romero M

Una vez que los dos módulos X-bee están transmitiendo y recibiendo datos entre ellos, se realiza las pruebas de comunicación entre la cámara inalámbrica y el computador.

Para esto se tiene que abrir Matlab y en la ventana de comandos escribir >>imaqtool y se abre una ventana donde aparecen las cámaras conectadas en el computador, en este caso aparecen dos, una que es la cámara que viene incorporada en la laptop y otra es la cámara USB inalámbrica.

Reconocimiento de la cámara inalámbrica en Matlab

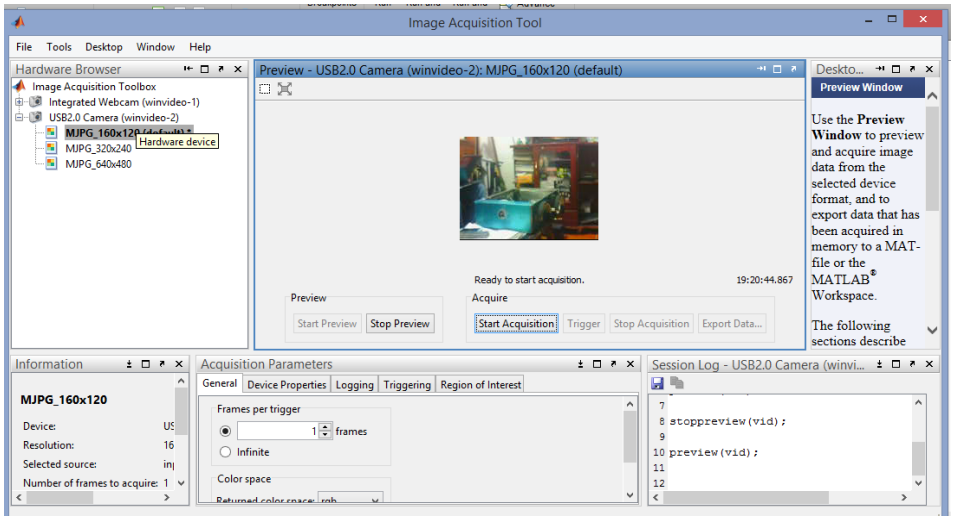


Figura 115.

Elaborado por: Heinerth G. Romero M

Se elige la resolución y se le da click en "Start Preview", la cámara empieza a funcionar. En la figura 116 se muestra la cámara funcionando y enviando imágenes de manera inalámbrica al computador.

Comunicación inalámbrica entre la cámara web y el computador.

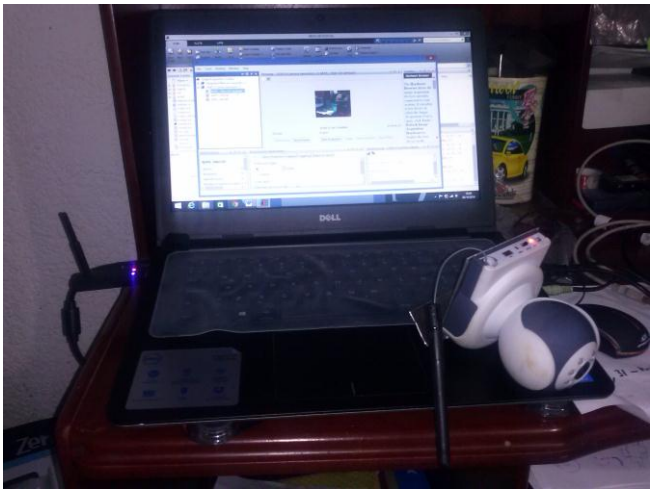


Figura 116.

Elaborado por: Heinerth G. Romero M

De igual forma desde Matlab se envían y reciben tramas de datos hacia el módulo Xbee que se encuentra en el robot, con esto se hace las pruebas de comunicación entre la tarjeta master y las tarjetas esclavos.

La tarjeta de control master recibe de manera inalámbrica tramas de datos desde el computador, la misma que se comunica de manera serial con todas las tarjetas esclavos, para comandar el robot.

4.3.2 Comunicación entre la tarjeta maestro y las tarjetas esclavos

Para las pruebas de comunicación serial entre la tarjeta maestro y las tarjetas esclavos se utilizó los leds indicadores que hay en cada tarjeta. Cada tarjeta esclavo tiene incorporado un dipswitch que permite direccionarlos para enviar las tramas de datos a cada uno de ellos, lo que se hizo fue mandar a encender y apagar los leds para verificar que los datos enviados desde el maestro le están llegando a cada uno de los esclavos.

Dipswitch para dar direcciones a los esclavos

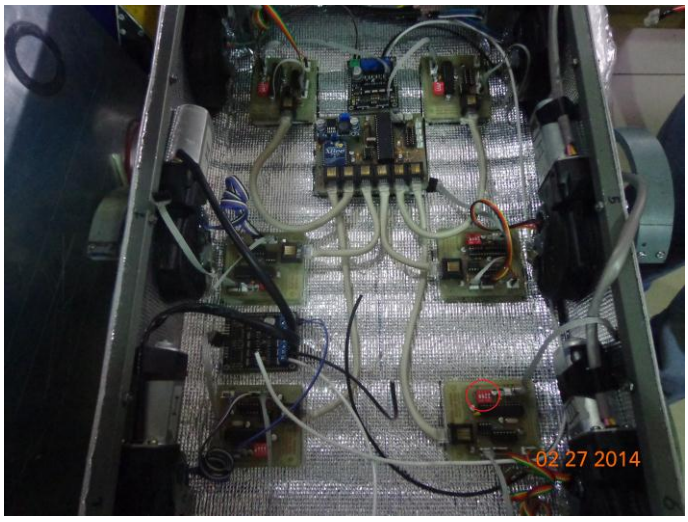


Figura 117.

Elaborado por: Heinerth Romero Macas

En la figura 118 se está probando a las tarjetas pares es por eso que se observa que los leds de estas tarjetas están encendidos.

Prueba de comunicación maestro-esclavos pares

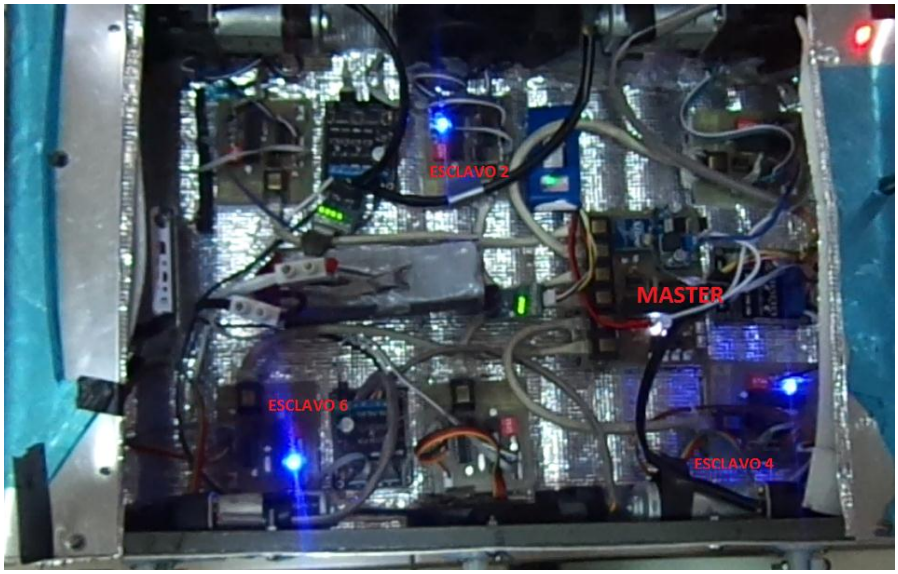


Figura 118.

Elaborado por: Heinerth Romero Macas

Luego se envió a apagar los leds pares y a encender los leds ipares.

Prueba de comunicación maestro-esclavos impares

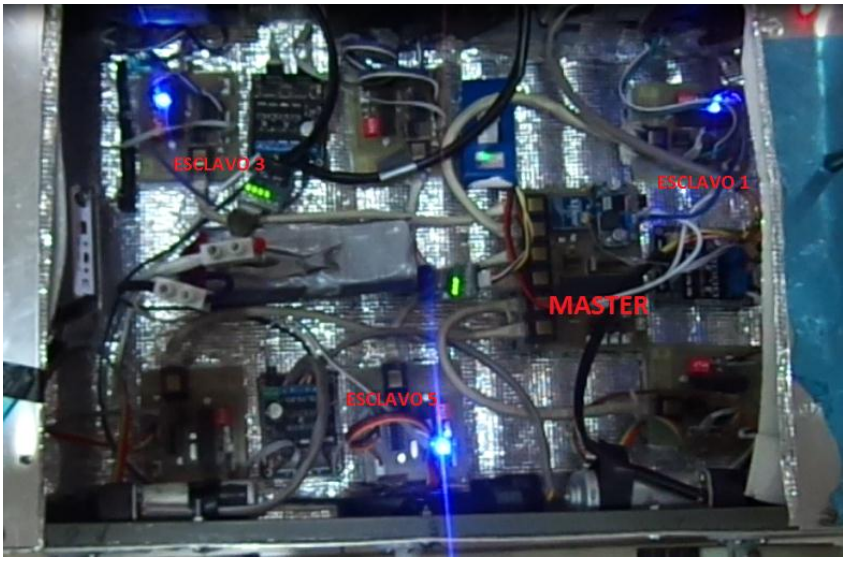


Figura 119.

Elaborado por: Heinerth Romero Macas

Con esto quedan probadas las tarjetas de control, sin embargo se deben hacer pruebas con todos los elementos conectados, para ver si no existe pérdidas de información por ruido, o por algún factor externo, en el apartado 4.4 se explica las modificaciones

que se tuvo que realizar para que se logre alcanzar la máxima distancia de comunicación.

4.4 Pruebas de comunicación inalámbrica (distancia)

Todo el diseño mecánico fue implementado para que el robot sea resistente a altas temperaturas, por lo que está sellado herméticamente y posee además termo espuma para repeler el calor y mantener la temperatura interna del robot baja.

En las primeras pruebas realizadas para medir la distancia máxima de comunicación inalámbrica, no alcanzaban más allá de 5 metros, esto se debía a que las antenas del módulo X-bee así como la de la cámara inalámbrica se encontraban dentro de la estructura del robot.

Razón por la cual se tuvo que hacer perforaciones en la estructura y utilizando extensiones de cables se las ubicó en la parte de externa del robot. Con esto se logró alcanzar las distancias señaladas en la hoja de datos de la cámara y del módulo X-bee.

Para la cámara inalámbrica se logró alcanzar una distancia máxima aproximada de 90 metros y para el módulo X-bee 600 metros, distancias que satisfacen las expectativas.

4.5 Pruebas del reconocimiento de imágenes

Cuando se presiona el botón “Iniciar Cámara” desaparece el logotipo del robot y la cámara empieza a generar imágenes en la pantalla.

Encendido de cámara

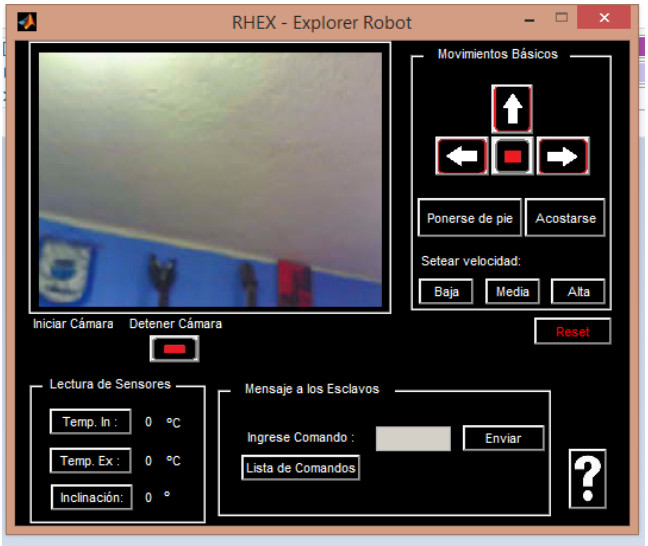


Figura 120.

Elaborado por: Heinerth Romero Macas

Si al iniciar la cámara esta no detecta ningún color que tenga una tonalidad parecida al color de piel, únicamente mostrará la imagen que está captando la cámara en ese momento, pero una vez que detecte este tipo de color, Matlab gracias al algoritmo de reconocimiento de imágenes, mostrará un recuadro de color rojo que indicará donde es posible que se encuentre una persona dentro del lugar donde el robot se encuentre explorando.

En la figura 121 se observa como detecta automáticamente el color del rostro de una persona y la enmarca en el recuadro. Si la persona se mueve o el robot a su vez sigue desplazándose el recuadro seguirá el color de la piel, siempre y cuando la persona se encuentre dentro del espectro que enfoca la cámara.

Prueba del reconocimiento de imágenes ambiente normal con y sin humo

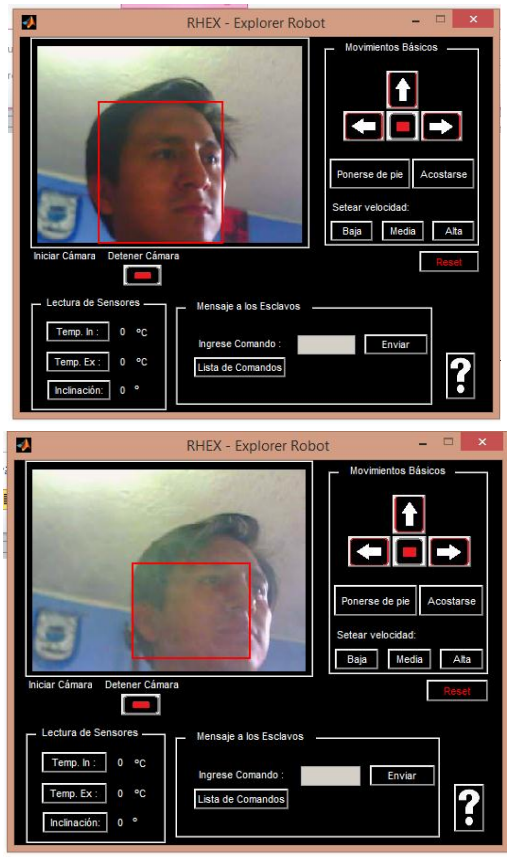


Figura 121.

Elaborado por: Heinerth Romero Macas

4.6 Pruebas de campo

Luego de verificar que el sistema mecánico, las tarjetas de control, el sistema de comunicación inalámbrica y el procesamiento de imágenes se encuentran funcionan correctamente, se realizó las pruebas de campo, donde el robot debe ser capaz de desplazarse con fluidez cobre cualquier superficie. A continuación se explicará cada una de estas pruebas.

4.6.1 Terreno plano

El robot cuenta con cuatro movimientos básicos:

1. Adelante
2. Atrás
3. Izquierda
4. Derecha

Para que el robot se desplace hacia adelante, las seis patas deben realizar una secuencia de trio esto consiste en que mientras tres se encuentran en movimiento las otras tres permanecen en el piso formando un trípode que mantienen al robot de pie, una vez que las primeras patas llegan a la posición cero las otras patas empiezan su movimiento; esta secuencia se repite hasta que el operador envíe el comando detenerse.

En la figura 122, se muestra el movimiento básico para que camine hacia adelante.

Secuencia hacia adelante

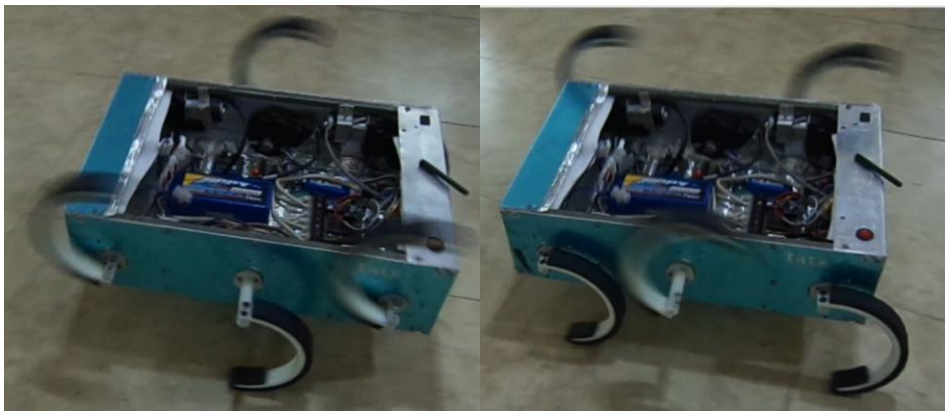


Figura 122.
Elaborado por: Heinerth Romero Macas

En las pruebas de campo se pudo observar que el robot no mantiene una trayectoria recta. Al momento de que se le envía el comando caminar hacia adelante tiene una desviación considerable hacia su lado derecho.

Gráfico de desviación al caminar

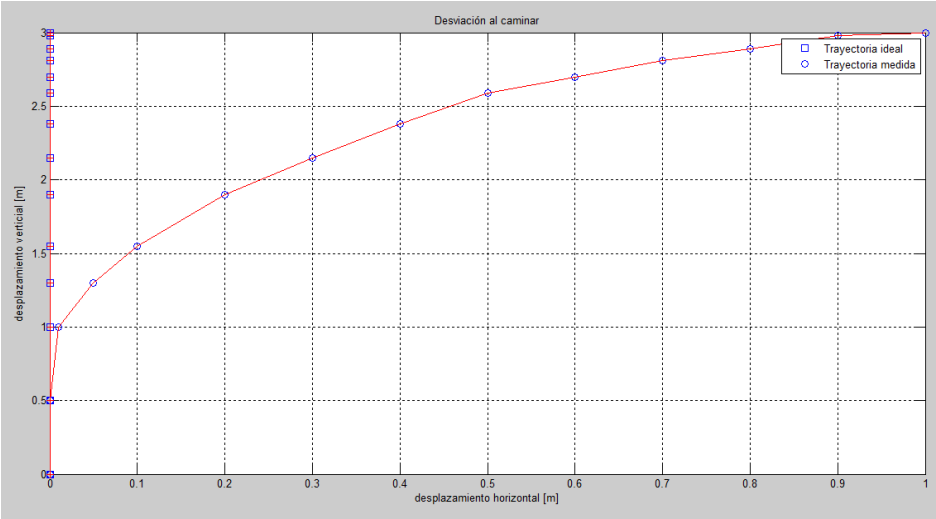


Figura 123.

Elaborado por: Heinerth Romero Macas

En la figura 123 se representa dicha desviación, esta gráfica fue realizada en el software Matlab, donde se observa una línea recta la cual representa la trayectoria ideal que debería seguir el robot, y una curva que es la desviación que tiene el robot al momento de caminar. Luego de verificar que el software esté correcto, y que las patas estén bien sujetas a los ejes de los motores, se determinó que esta desviación se debe al hardware mecánico.

Las patas deben estar bien alineadas, tanto horizontal como verticalmente, se usó una herramienta llamada nivelador de burbuja, para alinear cada una de las patas, y se las dejó lo más centradas posible. Sin embargo no se pudo corregir el error de forma mecánica.

4.6.1.1 Corrección de error para trayectoria recta.

Este error que existe de un metro hacia la derecha por cada tres metros desplazados hacia adelante, fue corregido por programación. Cada pata tiene cuatro parámetros que deben ser seteados antes de que el robot empiece a caminar, estos parámetros son: velocidad de avance, velocidad tracción, pulsos de avance y pulsos de tracción, estos hacen referencia a la velocidad y al número de pulsos que se desfasa cada pata para que exista un desplazamiento lineal.

Se realizó una tabla para determinar el valor de velocidad y número de pasos para cada pata. Si se pone el mismo valor de velocidad y mismo número de pasos, el robot tiende por desviarse hacia su derecha.

La Tabla 11 muestra las combinaciones probadas antes de determinar que valores le corresponde a cada pata para eliminar el error de desplazamiento.

Tabla 11.

Combinaciones probadas para eliminar error de trayectoria

Prueba	Velocidad avance	Velocidad tracción	Pulsos Tracción	Pulsos_tracción patas						Observaciones
				P6	P5	P4	P3	P2	P1	
1	50%	30%	45	3	3	3	3	3	3	se va hacia la derecha bastane
2	50%	30%	42	3	3	3	3	3	3	se va hacia la izquierda poco
3	50%	30%	42	4	4	4	4	4	4	se va hacia la izquierda poco
4	50%	30%	42	5	5	5	5	5	5	se va cayendo hacia delante
5	50%	30%	44	5	5	5	5	5	5	se va cayendo hacia delante
6	50%	30%	45	5	5	5	5	5	5	solo da un paso a causa de que el comienzo está mal
7	50%	30%	45	4	4	4	4	4	4	se va muy poco hacia la izquierda
8	50%	30%	44	4	4	4	4	4	4	pero el primer paso lo gira a la izquierda
9	50%	30%	43	4	4	4	4	4	4	se muy poco hacia la izquierda
10	55%	30%	44	4	4	4	4	4	4	con el primer paso lo gira a la izquierda
11	60%	30%	44	4	4	4	4	4	4	se va hacia la izquierda poco
12	60%	30%	43	4	4	4	4	4	4	se va hacia la izquierda poco
13	60%	35%	43	4	4	4	4	4	4	la 4 no esta arrastrando
14	65%	25%	43	4	4	4	4	4	4	la 5 tiene #005
15	61%	30%	42	4	4	4	4	4	4	salta mucho
16	61%	30%	44	4	4	4	4	4	4	no salta
17	65%	22%	43	4	4	5	5	5	5	la pata 4 tiene 66% de avance
18	80%	22%	43	5	6	5	5	5	5	la pata 4 tiene 81% de avance
19	68%	19%	44	6	5	5	5	5	5	excelente !! La pata 4 debe tener siempre 1% +

Nota. Elaborado por: Heinerth Romero Macas

Estos parámetros no pueden ser visualizados por el usuario final, ya que el panel de control final, únicamente cuenta con los botones básicos para que al operador se le haga fácil el manejo del robot.

Una vez eliminado el error, el robot fue capaz de avanzar 3 metros teniendo como desviación tan solo 0.01m hacia su derecha, cabe mencionar también que el desplazamiento lineal puede variar dependiendo la irregularidad del terreno, las pruebas que se realizaron fueron sobre baldosas de cerámica. A continuación se explicará las pruebas realizadas sobre superficies irregulares.

4.6.2 Terreno irregular

Se consideran terrenos irregulares a las superficies en donde existan desniveles, obstáculos o factores que dificulten el desplazamiento normal del robot.

Para realizar estas pruebas se llevó al robot a un parque, donde la superficie del terreno es césped y hay pendientes que ponen a prueba tanto el hardware mecánico como la programación.

Pruebas de campo, terreno irregular

Figura 124.
Elaborado por: Heinerth Romero Macas

Las primeras pruebas realizadas sobre césped en un terreno plano irregular cumplen las expectativas, puede caminar sin contratiempos. Sin embargo al intentar subir una

pendiente el robot tiene dificultades para seguir avanzando, debido a que el peso del robot se concentra en la parte posterior del robot, las patas traseras no tienen la suficiente fuerza para seguir avanzado y el robot se detiene solo.

4.6.3 Corrección para pendientes

El robot se desplaza sin inconvenientes por superficies planas, pero a la hora de subir o bajar pendientes se detiene solo o pierde su estabilidad, para esto se tuvo que implementar un sistema que permita determinar la inclinación del terreno.

El dispositivo que se incorporó en el robot fue un acelerómetro análogo de tres ejes, únicamente se utiliza uno de sus ejes para determinar la inclinación del robot, con esto se puede aumentar o disminuir el porcentaje de PWM y pulsos de retraso de cada pata, cuando detecte que la estructura del robot está inclinada.

Pruebas de campo, terreno irregular (con inclinación)

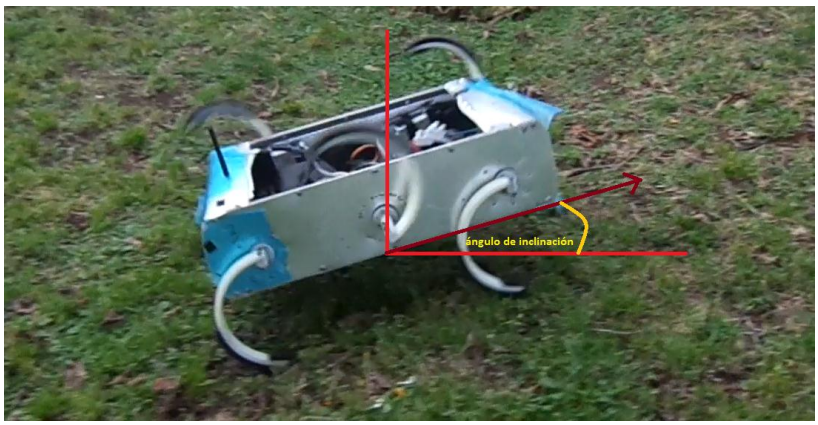


Figura 125.
Elaborado por: Heinerth Romero Macas

Al conocer la inclinación del terreno, lo que se hizo fue incrementar la velocidad y torque de las patas traseras cuando tenga que subir una pendiente, y si se encuentra bajando, las patas delanteras aumentarán su velocidad para evitar así que el peso del robot se concentre al frente.

Otra corrección que se hizo fue incrementar un freno por programación: lo que se hizo fue enviar un porcentaje muy bajo de PWM a las patas que estén sobre el suelo, que no sea capaz de mover la pata, pero si se aplica una fuerza en sentido contrario, estas ejercen una fuerza que se opone al movimiento de dicha fuerza. Haciendo que estas tres patas mantengan en pie al robot.

Luego de hacer las correcciones necesarias para que el robot se desplace en línea recta y determine la inclinación del terreno, se realizó pruebas sobre distintos tipos de superficies para determinar el comportamiento del robot sobre cada una de ellas, las superficies en las cuales se realizaron las pruebas son: baldosa, césped, concreto, madera.

4.6.4 Pruebas sobre varias superficies

Las superficies escogidas para realizar las pruebas fueron:

- Concreto
- Césped
- Madera
- Baldosa

Se escogió estas superficies porque son las más comunes en los hogares, para estas pruebas se evaluó el tiempo que tarda el robot en recorrer una determinada distancia, esto reflejará sobre qué superficie el robot es más eficiente, para mostrar los resultados de las pruebas se utilizó el software Matlab.

La distancia máxima para estas pruebas fue de diez metros, y se tomó 3 muestras de tiempo para cada tipo de superficie, y se obtuvieron los siguientes resultados:

4.6.4.1 Concreto.

Los tres tiempos que se obtuvo al realizar las pruebas sobre este tipo de superficie fueron:

Tabla 12.

Tiempos tomados en la prueba de campo sobre concreto

	Tipo de terreno	**Tiempo [s]**
Prueba 1	Plano	24.74
Prueba 2	Plano	25.09
Prueba 3	Plano	24.89
	Tiempo promedio:	24.91

Nota. Se calculó el tiempo promedio de las tres pruebas

Elaborado por: Heinerth Romero Macas

Con esta información se puede calcular la ecuación que refleja el comportamiento del robot en esta superficie.

Se sabe que en el tiempo cero segundos la distancia recorrida es cero metros, y que en el tiempo 24.91 segundos la distancia recorrida es 10 metros, por lo que se tiene dos puntos:

p1 (0 , 0) (1)

p2 (24.91 , 10) (2)

se sabe también que la ecuación de una recta es:

$y = mx+b$ (3)

a continuación se procede a calcular la pendiente:

$m = (y2 - y1) / (x2 - x1)$ (4)

$m = (10 - 0) / (24.91 - 0)$

$m = 10/24.91$ //

si se remplaza (1) en (3) se obtiene:

$b = 0$

por lo tanto la ecuación es:

$y=10*x / 24.91$ (5)

con esto se procede a graficar la recta en Matlab, para luego comparar con el resto de pruebas.

Gráfica distancia vs tiempo, concreto

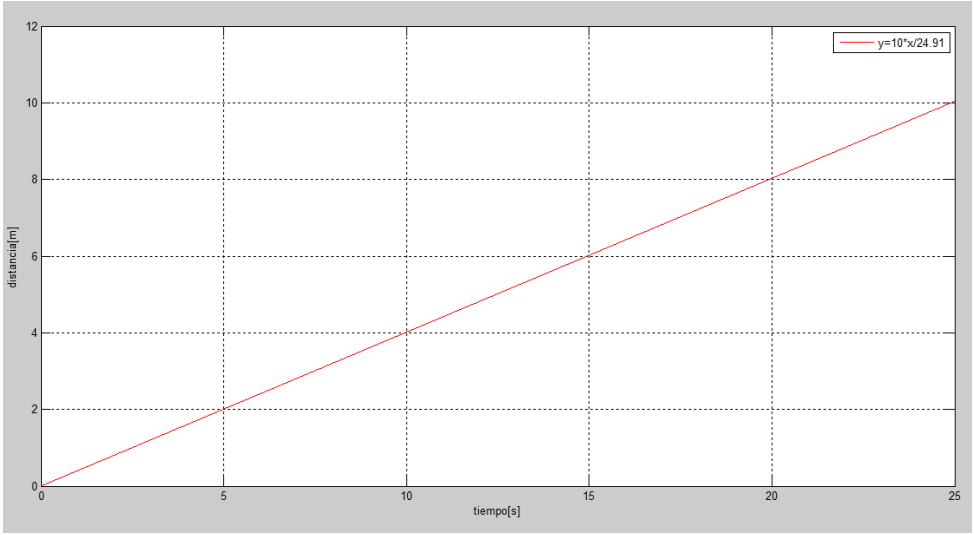


Figura 126.

Elaborada por: Heinerth Romero Macas

4.6.4.2 Baldosa

Tabla 13.

Tiempos tomados en la prueba de campo sobre baldosa

	Tipo de terreno	**Tiempo [s]**
Prueba 1	Plano	30.17
Prueba 2	Plano	29.95
Prueba 3	Plano	30.32
	Tiempo promedio:	30.15

Nota. Se calculó el tiempo promedio de las tres pruebas

Elaborado por: Heinerth Romero Macas

Se sabe que en el tiempo cero segundos la distancia recorrida es cero metros, y que en el tiempo 30.15 segundos la distancia recorrida es 10 metros, por lo que se tiene dos puntos:

p1(0 , 0) (6)

p2(30.15 , 10) (7)

al remplazar (6) y (7) en (4) y se obtiene:

m = 10/30.15 //

y al replazar (6) en (3) se obtiene que:

b = 0

por lo tanto la ecuación es:

y=10*x / 30.15

con esto se procede a graficar la recta en Matlab, para luego comparar con el resto de pruebas.

Gráfica distancia vs tiempo, baldosa

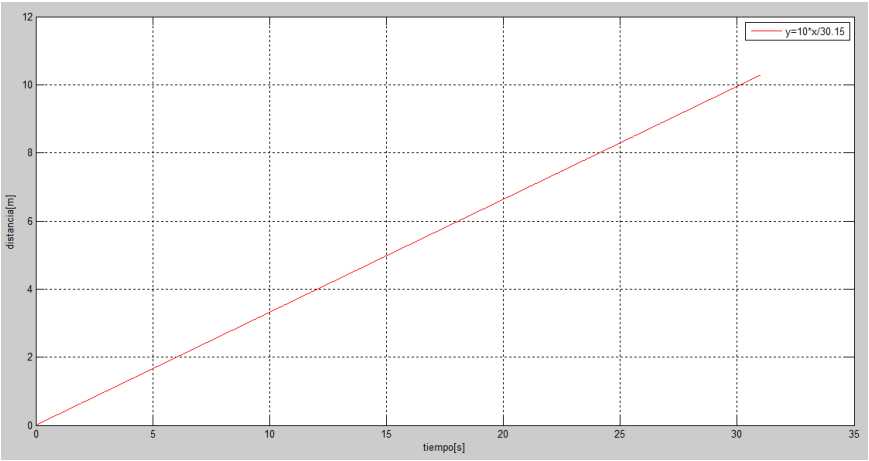


Figura 127.

Elaborada por: Heinerth Romero Macas

4.6.4.3 Madera

Tabla 14.

Tiempos tomados en la prueba de campo sobre madera

	Tipo de terreno	**Tiempo [s]**
Prueba 1	Plano	28.94
Prueba 2	Plano	28.97
Prueba 3	Plano	29.18
	Tiempo promedio:	29.03

Nota. Se calculó el tiempo promedio de las tres pruebas

Elaborado por: Heinerth Romero Macas

Se sabe que en el tiempo cero segundos la distancia recorrida es cero metros, y que en el tiempo 29.03 segundos la distancia recorrida es 10 metros, por lo que se tiene dos puntos:

p1(0 , 0) (8)

p2(29.03 , 10) (9)

al remplazar (8) y (9) en (4) y se obtiene:

m = 10/29.03 //

y al remplazar (8) en (3) se obtiene que:

b = 0

por lo tanto la ecuación es:

y=10*x / 29.03

con esto se procede a graficar la recta en Matlab, para luego comparar con el resto de pruebas.

Gráfica distancia vs tiempo, madera

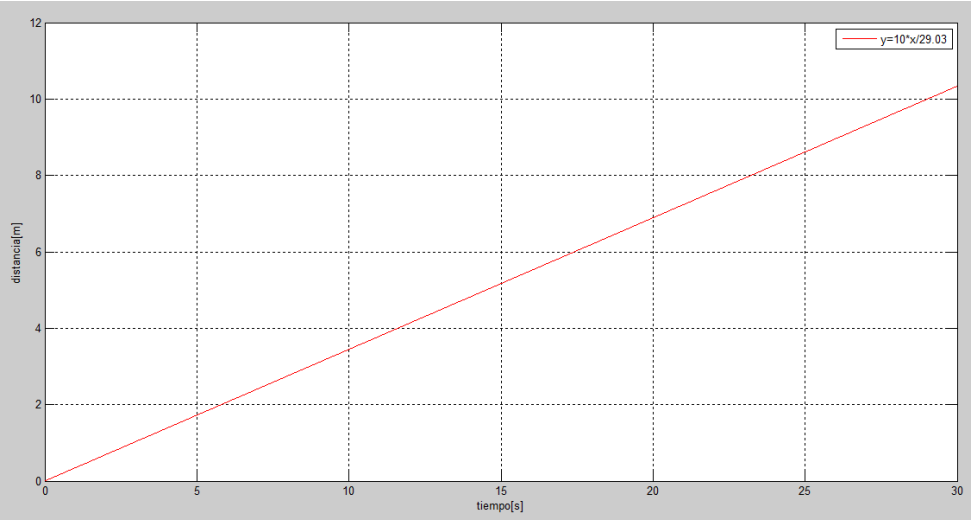


Figura 128.

Elaborada por: Heinerth Romero Macas

4.6.4.4 Césped

Tabla 15.

Tiempos tomados en la prueba de campo sobre césped

	Tipo de terreno	**Tiempo [s]**
Prueba 1	Plano	26.84
Prueba 2	Plano	27.48
Prueba 3	Plano	27.23
	Tiempo promedio:	27.18

Nota. Se calculó el tiempo promedio de las tres pruebas

Elaborado por: Heinerth Romero Macas

Se sabe que en el tiempo cero segundos la distancia recorrida es cero metros, y que en el tiempo 27.18 segundos la distancia recorrida es 10 metros, por lo que se tiene dos puntos:

p1(0 , 0) (10)

p2(27.18 , 10) (11)

al remplazar (10) y (11) en (4) y se obtiene:

m = 10/27.18 //

y al remplazar (8) en (3) se obtiene que:

b = 0

por lo tanto la ecuación es:

y=10*x / 27.18

con esto se procede a graficar la recta en Matlab, para luego comparar con el resto de pruebas.

Gráfica distancia vs tiempo, césped

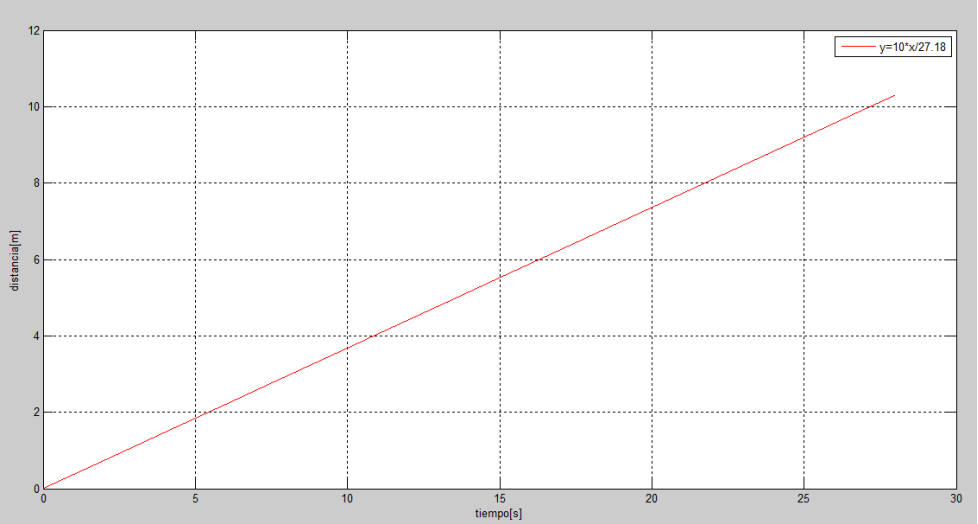


Figura 129.

Elaborada por: Heinerth Romero Macas

4.6.5 Pruebas con ambiente inflamable

Estas pruebas fueron realizadas con un ambiente inflamable controlado, se encendió pedazos de madera y cartón para generar llamas que simule un incendio.

Pruebas en ambiente inflamable

Figura 130.

Elaborado por: Paúl Cueva Castillo

Se hizo caminar al robot sobre las llamas para comprobar que es capaz de soportar altas temperaturas y mantener una temperatura estable en su interior.

Caminando sobre llamas 1

Figura 131.

Elaborado por: Paúl Cueva Castillo

Caminando sobre llamas 2

Figura 132.

Elaborado por: Paúl Cueva Castillo

Las temperaturas que marcaron los sensores externos e internos al momento de atravesar las llamas fueron 143°C y 28°C respectivamente, algo que se pudo observar fue que las patas al estar en contacto directo con las llamas soportan la mayor temperatura, lo que hizo que el caucho se ablandara, sin embargo no se deformaron ni se derritieron, cabe mencionar que en las pruebas se hizo caminar al robot por sobre las llamas por un tiempo de 20 minutos.

Aspecto de las patas luego de las pruebas en ambiente inflamable

Figura 133.

Elaborado por: Paúl Cueva Castillo

4.7 Resultados obtenidos

En este capítulo se ha explicado paso a paso el proceso de pruebas realizado para cada parte del sistema del robot. Esto permite realizar una comparación con lo propuesto al inicio de este trabajo de grado. La cual consistía en diseñar y construir un robot hexápodo tipo Rhex para ambientes inflamables controlado vía inalámbrica, y como extra se ha incorporado un sistema de reconocimiento de personas, mediante una cámara web inalámbrica y procesamiento de imágenes, para facilitar su labor de rescate.

Comparación de velocidades en todos los tipos de terreno

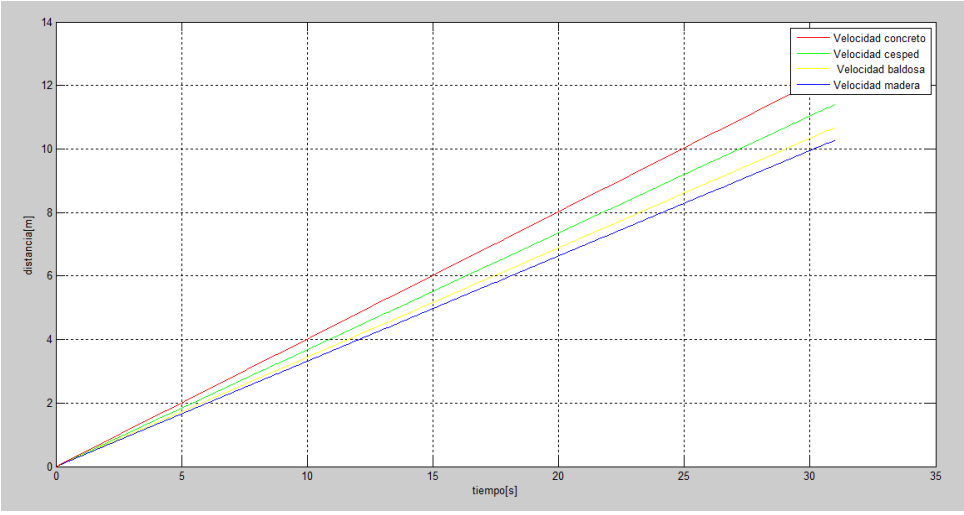


Figura 134.

Elaborada por: Heinerth Romero Macas

Los resultados obtenidos en las pruebas sobre los distintos tipos se aprecian en la figura 134, donde se hace una comparación del comportamiento del robot en los cuatro tipos de terreno más comunes. La velocidad mínima a la que se desplazará el robot será de 0.33m/s sobre madera y como máxima 0.40m/s sobre concreto, dejando un promedio de 0.36 m/s.

En las pruebas en ambientes inflamables se demostró que el robot es capaz de soportar hasta 20 minutos expuesto a llamas de altura de hasta 1 metro, sin sufrir daños en el hardware mecánico y hardware electrónico, también se observó que el caucho de las patas luego de las pruebas volvieron a tener su aspecto original, ya que durante las pruebas se observó que se ablandaron.

Gracias a la información obtenida en las pruebas de campo, se puede afirmar que el robot móvil explorador desarrollado cumple con los parámetros de seguridad, maniobrabilidad, y funcionamiento, al mismo tiempo se deja abierta vías de investigación para futuros cambios que permitan mejorar el presente trabajo.

CONCLUSIONES

- Para garantizar que el robot se desplace con facilidad sobre terrenos irregulares y no sufra daños en los componentes internos tales como: las tarjetas de control, cables de conexión, sensores y baterías, se diseñó y construyó un hardware mecánico capaz de cumplir estos dos requerimientos de movilidad, por estas razones se optó por usar tubo cuadrado de hierro dulce para la estructura interna, el cual es un material resistente muy común y fácil de encontrar en el mercado. Para el recubrimiento de los seis lados del chasis se usó láminas de aluminio, ya que es un metal muy liviano y que sirve como aislante térmico. La estructura total fue evaluada durante la etapa de pruebas, donde se sometió al robot a pruebas en distintos tipos de terreno y ambiente inflamable, comprobando que tanto el hierro dulce como el aluminio no sufrieron ningún daño, razón por la cual se determinó que son idóneos para la fabricación de la estructura del robot y pueden ser utilizados en proyectos con similares requerimientos.

- Se ha logrado implementar la comunicación inalámbrica mediante el uso de módulos comerciales Xbee Pro, ya que cuentan con un protocolo robusto y de largo alcance, durante las pruebas de campo se alcanzó distancias de hasta 600 metros esto indica que este tipo de módulos son aptos para la construcción de prototipos móviles que requieran ser controlados a largas distancias y sin pérdida de información.

- Las antenas de los módulos Xbee deben estar en línea de vista para que logren alcanzar la distancia de comunicación máxima que especifica en las hojas de datos técnicos del fabricante, por lo que se tuvo que extender el cable de la antena para ubicarlas en la parte exterior, si se dejaba la antena del módulo Xbee en la parte interna del robot, la señal que le llegaba al robot era muy débil y se perdía comunicación a 200 metros.

- La programación del microcontrolador maestro y de los esclavos, desarrolladas en lenguaje C, resultó de gran ayuda, porque optimizó el tiempo

de desarrollo de los algoritmos, ya que estos son los principales encargados de controlar el robot, y sin la correcta comunicación y transferencia de datos entre los microcontroladores que conforman el sistema, el robot no se podría ni siquiera hacer mover, peor aún usarlo como herramienta para la búsqueda y rescate de víctimas en incendios.

- Para la sujeción de las patas a los ejes de los motores se utilizó pernos avellanados con tuercas de seguridad, debido a que los pernos comunes usados en la etapa de pruebas se aflojaban luego de un tiempo de uso, lo que hacía que el robot tenga una desviación al momento de estar ejecutando las rutinas de movimientos, por esta razón se descartó el uso de pernos comunes.

- Para enviar comandos y visualizar las lecturas de los sensores acoplados al robot en la interfaz gráfica de usuario, se empleó un puerto de comunicación serial (puerto COM en el computador) para el envío y recepción de datos, debido a la fácil implementación y manejo de información con Matlab, y porque el módulo Xbee permite la conexión USB con el computador mediante un conversor serial/USB.

- Para el reconocimiento de personas se implementó un algoritmo que detecta segmentos de color piel de las personas y muestra como resultado un rectángulo de seguimiento sobre el área de piel más grande encontrada, de esta manera en una búsqueda y rescate se tiene la ventaja de reconocer a una persona al visualizar únicamente una parte de su piel. Se realizó el seguimiento al área de piel más grande para evitar la confusión del operador ya que es muy probable que se detecten varias secciones durante la búsqueda.

- Para la elaboración del algoritmo de detección de piel se trabajó dentro del campo de procesamiento de imágenes digitales, empleando técnicas de filtrado morfológico en el espacio de color HSV. Se utilizó el filtrado morfológico porque permite la obtención selectiva de estructuras y objetos de la imagen, y se empleó el modelo HSV porque permite trabajar con rangos de

color independientemente del brillo que otorga a la imagen las variaciones de iluminación del ambiente.

- Durante la etapa de pruebas de campo en cuanto al desplazamiento del prototipo sobre terrenos irregulares (concreto, madera, césped), se consiguió una desviación con tendencia al lado derecho en el recorrido rectilíneo del robot de un centímetro entre el eje de movimiento y la trayectoria obtenida por cada 3 metros de desplazamiento, que en comparación al error de 100 centímetros obtenido durante la primera prueba, se logró una corrección el error de un 99%. La corrección se realizó por software modificando la velocidad de las patas para obtener la compensación y corrección deseada.

- Para proteger a los componentes electrónicos dentro del robot se recubrió las paredes internas con dos capas de espuma aislante térmica para mantener a dichos componentes en condiciones de temperatura adecuadas para su funcionamiento, de tal modo que al realizar la pruebas de campo en ambientes inflamables la temperatura máxima al interior del robot fue 28°C mientras la temperatura al exterior del prototipo era de 143°C, y considerando que los circuitos son capaces de trabajar a una temperatura máxima de 100°C se determina que el sistema de aislamiento térmico cumple su función satisfactoriamente.

- En cada uno de los motores de las patas, se instaló un disco generador de pulsos y dos sensores ópticos reflectivos con el fin de formar un sistema codificador rotatorio o encoder para el control de posición, lo cual permitió convertir un motor de corriente continua común en un motor servo de posicionamiento angular de alta resolución al ser posible tener hasta 8 divisiones por milímetro en el disco de encoder, esto a un bajo costo en relación a los existentes en el mercado. Diseños basados en este modelo se podrían emplear en el desarrollo de aplicaciones de robótica y control electrónico en general.

- En la codificación del software maestro esclavo fue necesario aprovechar al máximo los recursos de velocidad y memoria de los procesadores debido a que la respuesta óptima de la red es fundamental para mantener el sincronismo de las patas al ejecutar las secuencias de movimiento.

- Se implementó un modo de reinicio por software, de tal manera que desde la propia interfaz el operador sea capaz de reiniciar los registros de los microcontroladores a sus estados iniciales en caso de que el sistema sufra algún fallo durante su funcionamiento.

- Al evaluar la inclinación del terreno la lectura del acelerómetro se ve sumamente afectada por ruido debido a la vibración normal del robot al moverse, por tal razón dicha lectura no se realiza de manera constante sino que primero es necesario detener la locomoción del robot para efectuar la lectura.

- La funcionalidad del robot fue evaluada y calificada positivamente por personas expertas en el tema de robótica, ya que fue inscrito en tres Concursos de Robótica dos Nacionales y uno Internacional, donde consiguió dos primeros lugares y segundo lugar en la categoría Impacto Tecnológico, demostrando ser un proyecto innovador con un alto impacto en la sociedad.

RECOMENDACIONES

- Antes de empezar cualquier proyecto que involucre la fabricación de piezas mecánicas, es necesario que se investigue y se pida asesoría a personas que dominen el tema, porque al ser estudiantes de ingeniería electrónica se encontró muchos inconvenientes para el diseño, selección y construcción de las piezas del robot.

- Es importante leer las hojas de datos de todos los dispositivos electrónicos y tarjetas comerciales usadas, ya que una mala conexión o exceder los niveles de voltaje y/o corriente, puede provocar que se quemen los dispositivo y se perjudique a otros componentes.

- Para corregir el problema de lecturas erróneas del acelerómetro debido al ruido producido por vibración, se recomienda para una nueva versión del prototipo implementar filtros sofisticados tales como filtro de Gauss, filtro complementario o filtro Kalman, y además incluir un giroscopio para corregir aún más la lectura. Con la implementación de la lectura constante del acelerómetro se puede diseñar algoritmos dinámicos que se adapten a la inclinación del terreno.

- Para mejorar el prototipo en el futuro se recomienda utilizar en la toma de video una cámara térmica, de tal modo que la presencia de neblina, humo o fuego no afecte a la detección de personas.

LISTA DE REFERENCIAS

- Atmel Corporation. (2013). *Atmel.* Obtenido de http://www.atmel.com/images/atmel-8272-8-bit-avr-microcontroller-atmega164a_pa-324a_pa-644a_pa-1284_p_datasheet.pdf
- AxisCommunicationsAB. (2013). *Axis*. Obtenido de http://www.axis.com/es/products/video/about_networkvideo/image_sensors.htm#cmos
- Burgath-Guatemala-S.A. (2008). *Burgath*. Obtenido de http://burgath.com/sabias-que/pwm-modulacion-por-ancho-de-pulso/
- Corporation, A. (07 de 2010). *Hoja de datos.* Obtenido de http://shop.emscdn.com/catalog/components/semi/mcu/atmega164p/atmega164p.pdf
- Elechouse. (2014). *Elechouse*. Obtenido de http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=100_146&products_id=2179
- Espinosa, F. S. (2012). Los Microcontroladores de Atmel. Huajuapan de León, Oax.: Optime Impresos.
- Francisco, O. (2002). Tesis Doctoral. *Procesamiento morfológico de imágenes en color. Aplicación a la reconstrucción geodésica*. Alicante, España: Universidad de Alicante.
- IdeasyTecnología. (2014). *IDETEC*. Obtenido de http://www.ideastechnology.com/
- MikroElektronika. (21 de Marzo de 2013). *mikroe*. Obtenido de http://www.mikroe.com/mikroc/avr/
- Noelia Sanjurju Dovigo, L. P. (2012). *S.A.B.I.A.* Obtenido de Sistemas Adaptativos y Bioinspirados en Inteligencia Artificial: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Tutoriales/color/HSV.htm
- Oyarce, A. (Julio de 2010). *Omlimex.* Obtenido de www.olimex.cl/pdf/Wireless/ZigBee/XBee-Guia_Usuario.pdf
- Pensilvania, U. d. (4 de 11 de 2010). *Kodlab.* Obtenido de http://kodlab.seas.upenn.edu/uploads/Main/xrhextechreport.pdf

- Soto, J. (31 de Octubre de 2013). *Open RC.* Obtenido de http://openrcecuador.com/conceptos-basicos-en-las-baterias-lipo-tensioncapacidad-descarga-y-conexion-de-elementos/

- Wikipedia. (8 de Marzo de 2014). *wikipedia*. Obtenido de http://es.wikipedia.org/wiki/Convertidor_DC_a_DC

- Wikispaces. (2015). *Wikispaces.* Recuperado el 9 de Marzo de 2015, de http://sx-de-tx.wikispaces.com/ZIGBEE

ANEXOS

Anexo 1

Programa de la interfaz gráfica

- **Programa Principal**

```
function varargout = InterfaceRHEX_P3(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@InterfaceRHEX_P3_OpeningFcn, ...
                   'gui_OutputFcn',
@InterfaceRHEX_P3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function InterfaceRHEX_P3_OpeningFcn(hObject, eventdata,
handles, varargin)

%Mostrar imagen play.jpg en botón Play
[a,map]=imread('play.jpg');
[r,c,d]=size(a);
x=ceil(r/30);
y=ceil(c/60);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Play,'CData',g);


%Mostrar imagen pause.jpg en botón Pause
[a,map]=imread('stop.jpg');
[r,c,d]=size(a);
x=ceil(r/30);
y=ceil(c/60);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Pause,'CData',g);

%Mostrar imagen adelante.jpg en botón Adelante
[a,map]=imread('adelante.jpg');
[r,c,d]=size(a);
x=ceil(r/80);
y=ceil(c/40);
g=a(1:x:end,1:y:end,:);
```

```
g(g==255)=5.5*255;
set(handles.Adelante,'CData',g);

%Mostrar imagen inf.jpg en pushbutton Información
[a,map]=imread('inf.jpg');
[r,c,d]=size(a);
x=ceil(r/80);
y=ceil(c/40);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.inf_proyecto,'CData',g);

%Mostrar imagen derecha.jpg en pushbutton Derecha
[a,map]=imread('derecha.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/80);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Derecha,'CData',g);

%Mostrar imagen izquierda.jpg en pushbutton Izquierda
[a,map]=imread('izquierda.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/80);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Izquierda,'CData',g);

%Mostrar imagen parar.jpg en pushbutton Parar
[a,map]=imread('parar.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/40);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Parar,'CData',g);

%Mostrar imagen logo_rhex.jpg en axes1 cuando la cámara esté
apagada
axes(handles.axes1)
Rhex = imread('logo_rhex.jpg');
axis off;
imshow(Rhex);

handles.output = hObject;
guidata(hObject, handles);
function varargout = InterfaceRHEX_P3_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;

%********** LEER DATOS DE LOS SENSORES DE TEMPEARATURA
**********
function pushtemp1_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Pedido de temperatura S1')
fopen(PS);
```

```
fprintf(PS,'%s','Lm')%envío un dato entero
variable = fscanf(PS,'format',2) %fscanf(obj,'format',size)
set(handles.settemp1,'String',variable);
fclose(PS); delete(PS); clear PS;

function pushtemp2_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Pedido de temperatura S2')
fopen(PS);
fprintf(PS,'%s','Im')%envío un entero
variable = fscanf(PS,'format',2)
set(handles.settemp2,'String',variable);
fclose(PS); delete(PS); clear PS;

% ***************** BOTONES MOVIMIENTOS GENERALES:
*****************
% ponerse de pie
function ponerse_de_pie_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Pos. Inicial (bzm)')
fopen(PS);
fprintf(PS,'%s','bv025m')%envío de trama
pause(0.03)
fprintf(PS,'%s','lg023m')%envío de trama
pause(1)
fprintf(PS,'%s','bzm')%envío de trama
pause(1)
fprintf(PS,'%s','bv075m')%envío de trama
pause(0.03)
fprintf(PS,'%s','4v076m')%envío de trama
fclose(PS); delete(PS); clear PS;

% acostarse.
function acostarse_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Acostarse (MU040,NU017,OU017)')
fopen(PS);
fprintf(PS,'%s','Jm')%envío de trama
fclose(PS); delete(PS); clear PS;
% parar
function Parar_Callback(hObject, eventdata, handles)
clc;
global PS
global t
global PS
COM_SERIAL_INIT
disp('Parar (Xm)')
fopen(PS);
fprintf(PS,'%s','Xm')%envío de trama
fclose(PS); delete(PS); clear PS;

% adelante.
function Adelante_Callback(hObject, eventdata, handles)
clc;
global PS
global t
t=1;
```

```
COM_SERIAL_INIT
disp('Adelante (Wm)')
fopen(PS);
fprintf(PS,'%s','Wm')%envío de trama
fclose(PS); delete(PS); clear PS;

% derecha
function Derecha_Callback(hObject, eventdata, handles)
clc;
global PS
global t
t=2;
COM_SERIAL_INIT
disp('Derecha (Dm)')
fopen(PS);
fprintf(PS,'%s','Dm')%envío de trama
fclose(PS); delete(PS); clear PS;

% izquierda.
function Izquierda_Callback(hObject, eventdata, handles)
clc;
global PS
global t
t=2;
COM_SERIAL_INIT
disp('Izquierda (Am)')
fopen(PS);
fprintf(PS,'%s','Am')%envío de trama
fclose(PS); delete(PS); clear PS;


%***************** INICIAR/DETENER CÁMARA
************************

% iniciar_camara.
function iniciar_camara_Callback(hObject, eventdata, handles)

function detener_camara_Callback(hObject, eventdata, handles)

%********************** SETEAR VELOCIDAD
**********************
% inf_proyecto.
function inf_proyecto_Callback(hObject, eventdata, handles)
clc;
helpdlg({'Universidad Politécnica Salesiana Quito',' ',
'Autores:', '             Heinerth Romero Macas','
Jahiro P. Cueva C.',' ','Tutor:','             Ing. Roberto
Perez',' ', '             2014'},'RHEX');

% dato a enviar
function dato_a_enviar_Callback(hObject, eventdata, handles)
Dato=get(hObject,'String'); %Almacenar dato ingresado
handles.dato_a_enviar=Dato; %Almacenar en puntero
guidata(hObject,handles);%Guardar datos de la aplicación

if isempty(Dato) % evaluo si tengo o no un dato para enviar
   set(handles.boton_enviar,'enable','off'); % desactivo botón
enviar si no hay dato
else
```

```
   set(handles.boton_enviar,'enable','on'); % activo botón
enviar si tengo dato
end

function dato_a_enviar_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%********************** Botón Tramas
*****************************
% --- Executes on button press in boton_enviar.
function boton_enviar_Callback(hObject, eventdata, handles)
Trama=handles.dato_a_enviar;
clc; global PS
COM_SERIAL_INIT
disp('Trama (datos ingresados)')
disp(Trama)
fopen(PS);
fprintf(PS,'%s',[Trama])%envío una trama
fprintf(PS,'%s','m')%envío una trama
fclose(PS); delete(PS); clear PS;

function dato_a_enviar_KeyPressFcn(hObject, eventdata,
handles)
set(handles.boton_enviar,'enable','on'); % se activa botón
enviar al escribir algo en el edit text

%******************** Imagen con las Tramas
***********************
function pushbutton32_Callback(hObject, eventdata, handles)
clc;
Rhex = imread('tramas3.jpg');
figure(1);
imshow(Rhex);

%******************** Botones Cámara
*****************************

% Play.
function Play_Callback(hObject, eventdata, handles)
global x;
x=2;
% Aquirir la imagen de la cámara
vid = videoinput('winvideo', 2, 'MJPG_320x240');
set(vid,'TriggerRepeat',Inf);
vid.FrameGrabInterval = 3; %3
%vid.FramesAcquired = inf;
set(vid,'Timeout',900); % SETEO TIEMPO DE ESPERA EN CASO DE
DESCONEXIÓN DE LA CÁMARA
% ESPERA 900 SEG. ANTES DE QUE DE ERROR.
start(vid)
h_range = [0 0.11]; % Hue - > Tonalidad
s_range = [0.2 .7];  % Saturation -> Saturación
while x>1
img=getdata(vid,1);
axes(handles.axes1);
imshow(img)
seguir_cara(img, h_range, s_range);
```

```
drawnow
set(handles.Play,'visible','off');
end
set(handles.Play,'visible','on');
stop (vid)
imaqreset
axes(handles.axes1)
Rhex = imread('logo_rhex.jpg');
axis off;
imshow(Rhex);

% Pause.
function Pause_Callback(hObject, eventdata, handles)
clc
global x;
x=0;
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
delete(hObject);

% --- Executes during object creation, after setting all
properties.
function axes1_CreateFcn(hObject, eventdata, handles)

% pushbutton37.
function pushbutton37_Callback(hObject, eventdata, handles)

% Baja.
function Baja_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Velocidad baja (V050)')
fopen(PS);
fprintf(PS,'%s','bv050m')%envío de trama
pause(0.03)
fprintf(PS,'%s','4v051m')%envío de trama
fclose(PS); delete(PS); clear PS;

function Media_Callback(hObject, eventdata, handles)

clc; global PS
COM_SERIAL_INIT
disp('Velocidad media (bv75)')
fopen(PS);
fprintf(PS,'%s','bv075m')%envío de trama
pause(0.03)
fprintf(PS,'%s','4v076m')%envío de trama
fclose(PS); delete(PS); clear PS;

% Alta.
function Alta_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Velocidad alta (bv090)')
fopen(PS);
fprintf(PS,'%s','bv090m')%envío de trama
pause(0.03)
fprintf(PS,'%s','4v091m')%envío de trama
fclose(PS); delete(PS); clear PS;
```

```
% --- Executes on button press in Inclinacion.
function Inclinacion_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Pedido de grados de inclinación')
fopen(PS);
fprintf(PS,'%s','Um')%envío un entero
variable = fscanf(PS,'format',3);
set(handles.grados,'String',variable);
fclose(PS); delete(PS); clear PS;

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
clc; global PS
COM_SERIAL_INIT
disp('Reset (Rm)')
fopen(PS);
fprintf(PS,'%s','Rm')%envío de trama
fclose(PS); delete(PS); clear PS;
```

- **Función para reconocimiento de personas**

```
function seguir_cara(imagen, rango_h, rango_s)
%Leer imagen y ocnvertir a HSV
im = imagen;
im1 = double(im);
hsv_im1 = rgb2hsv(im1);

% Colocar H & S en nuevas variables
H = hsv_im1(:,:,1);
S = hsv_im1(:,:,2);
h = H;
s = S;

% Seleccionar el color de la piel: obtener los valores que
estén dentro del rango del color de la piel
piel = ((s>rango_s(1)) & (s<rango_s(2)) &(h>rango_h(1)) &
(h<rango_h(2)));
% Filtrar imagen
piel=bwmorph(piel,'close');
piel=bwmorph(piel,'open');
% Etiquetar objetos de la imagen BN
L=bwlabel(piel);
% Obtener áreas y rectángulo de seguimiento
out_a=regionprops(L);
% Contar el número de objetos
N=size(out_a,1);
%N=size(out_a,2);
if N < 1 || isempty(out_a) % Retorna si no hay ningún objeto
en la imagen
    return
end
% Seleccionar el área más grande
areas=[out_a.Area];
[m_x pam]=max(areas);
hold on %**********************************
% Plotear rectángulo de seguimiento
```

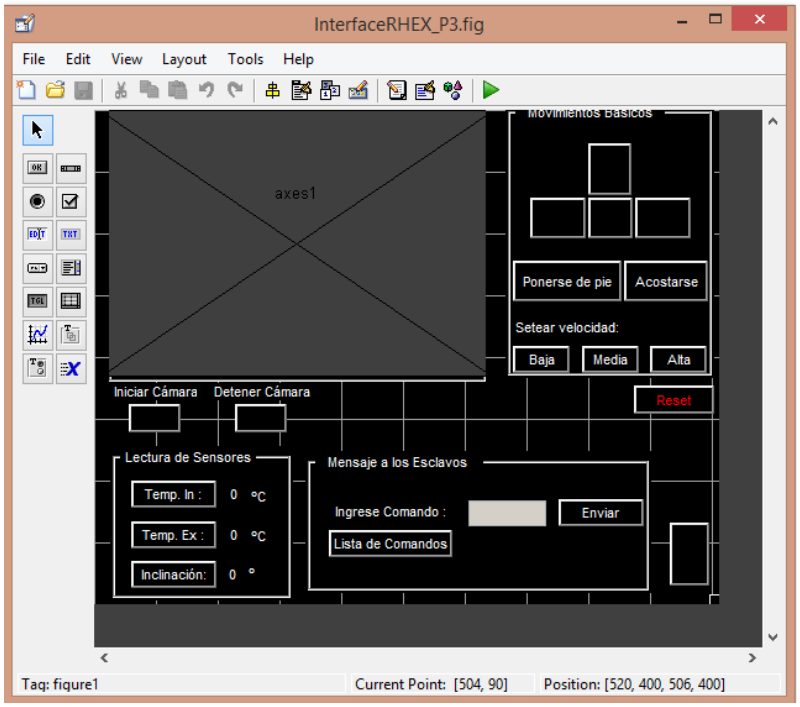
```
rectangle('Position',out_a(pam).BoundingBox,'EdgeColor','r','L
ineWidth',2)
hold off
```

- **Función para leer el puerto serial**

```
function  COM_SERIAL_INIT
disp('CONFIGURACION COM SERIAL')
global PS
PS=serial('COM5');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600
Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits,
debe estar entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza
el envío
set(PS,'OutputBufferSize',100); % "n" es el número de bytes a
enviar
set(PS,'InputBufferSize' ,100); % "n" es el número de bytes a
recibir
set(PS,'Timeout',5); % 5 segundos de tiempo de espera
disp('--COM SERIAL ACTIVA EN COM5--')

end
```

- **Guide de Matlab**



Anexo 2

Información de la cámara inalámbrica

Descripción:

- Cámara inalámbrica de hasta 100 metros de alcance.
- Diseño de escritorio con gancho para colgar en la pared.
- Usada como monitor para cuidar bebés u oficinas, casas, etc.
- Función de voz de dos vías puede escuchar y puede hablar.
- Grabación automática de video.

Especificaciones técnicas:

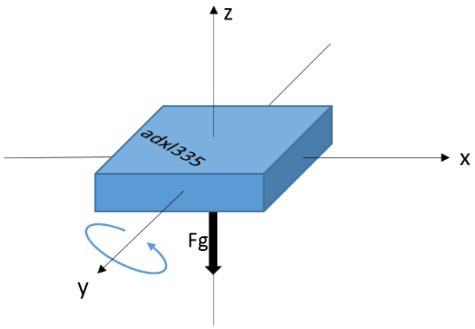
- **Marca:** Apex
- **Modelo:** AP-WC101
- **Material:** Plástico
- **Color:** Blanco con gris
- **Ancho:** 9cm.
- **Alto:** 11,5cm.
- **Conversación:** micrófono y altavoz integrado
- **Lente:** 5 glass de 330 pixels
- **Batería:** Recargable

Anexo3

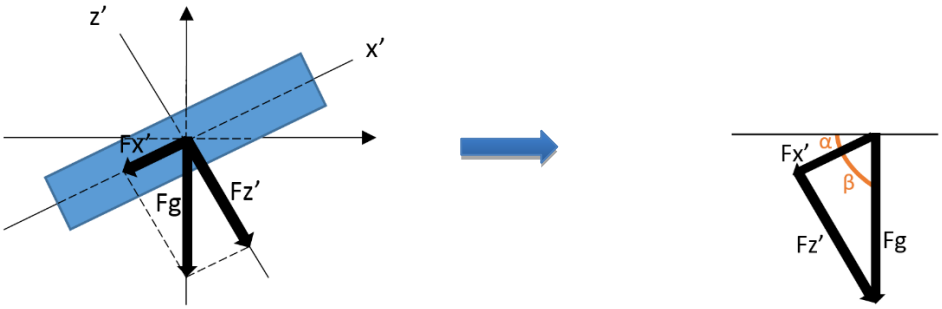
Cálculo de inclinación del terreno y temperatura

- **Cálculo de la inclinación usando el acelerómetro análogo de tres ejes adxl335**

Sobre una superficie plana con el robot en reposo sobre el acelerómetro solo actúa la fuerza de gravedad (Fg) como se muestra en la siguiente figura.



Si se inclina el acelerómetro de modo que gire alrededor del eje Y varían las componentes en Z y en X; y es posible simplificar el diagrama a dos dimensiones de la siguiente manera:



Ahora se tiene tres componentes de aceleración, con los cuales es posible hallar la inclinación α del plano sobre el que se encuentra el acelerómetro (y por lo tanto del robot) aplicando reglas trigonométricas como se muestra a continuación.

$$\tan\beta = \frac{Fz'}{Fx'}$$

$$\beta = \tan^{-1}\frac{Fz'}{Fx'}$$

$$\propto = 90° - \beta$$

$$\propto = 90 - \left(\tan^{-1}\frac{Fz'}{Fx'}\right) \qquad (a)$$

Se observa que sólo es necesario conectar dos de los ejes a los pines analógicos del MCU para calcular el ángulo alrededor del eje Y.

Para la implementación en el software se debe transformar los valores del conversor A/D a valores de aceleración. La conversión va de 0 a 1024 donde se obtiene la mitad del rango cuanto el sensor mide 0G (caída libre). Por lo tanto este valor se restará del valor leído.

Según la hoja de datos del adlx335 la sensibilidad es de 360mV por cada *G* de aceleración, donde la G equivale al valor de la gravedad en m/s2, y el sensor mide linealmente en el rango de -3G a 3G. Si se considera la aceleración como 10m/s2, cuando sobre uno de los ejes del sensor se tiene la máxima aceleración de 3g o 30m/s2, lectura del conversor A/D es de 1.08V. Por lo tanto para convertir la lectura de cualquiera de los ejes del valor del conversor a un valor de aceleración se aplica la siguiente fórmula:

$$aceleracion\left[\frac{m}{s^2}\right] = (lecturaADC - 512) * \frac{1.08}{30} \qquad (b)$$

- **Cálculo de la temperatura usando el sensor LM35**

El LM35 es un sensor lineal capaz de medir de -55°C a 150°C. Cuando el sensor está a 150°C la lectura del conversor A/D de 10 bits es de 308, con lo que es posible calcular la temperatura en °C aplicando una regla de tres simple:

$$temperatura[°C] = lecturADC * 150/308 \qquad (c)$$