BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS
DE LA COMPUTACIÓN

Diseño y construcción de un prototipo de robot móvil para el uso en la aplicación de algoritmos multiagentes.

TESIS PROFESIONAL
para obtener el título de
INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

Fernando Rodríguez Juárez

ASESOR:

M.C. Mariano Larios Gómez

COASESOR

M.C. Apolonio Ata Pérez

Puebla, Puebla. Noviembre de 2020

Dedicatoria:

Llegar hasta este momento ha sido largo, no cabe duda de que he tomado varias decisiones; de las cuales algunas han sido acertadas y otras no tanto. Estas junto con el apoyo incondicional de mi familia, amigos, y personas especiales, me han permitido avanzar y en cada una de esas decisiones, mis padres siempre han estado presentes, a pesar de no estar siempre de acuerdo, me han apoyado de manera incondicional, y me han brindado las posibilidades óptimas para terminar mis estudios y seguir adelante en mi formación profesional.

Es por ello que con mi profundo agradecimiento a Dios por estar siempre presente cuando cumplo cualquiera de mis objetivos, le dedico esta Tesis a mis Padres y a las personas más especiales que han estado presentes en cada momento de mi vida, acompañándome por el complejo y emocionante camino de la vida.

Agradecimientos:

Agradezco profundamente a todas las personas especiales que me han apoyado e impulsado a lo largo de mi vida y a lo largo de mi estancia en esta gran institución, la cual me brindo muchas experiencias agradables y conocimientos de gran importancia para mi desarrollo por lo cual, siempre estaré agradecido y recordare lo maravilloso que fue estudiar en la Benemérita Universidad Autónoma de Puebla, a mi familia que siempre ha estado ahí para apoyarme en todo momento cuidando mis pasos con gran amor, así como a mis mejores amigos que me han mostrado las verdaderas prioridades de la vida además de brindarme grandes momentos de diversión.

"Muchas Gracias"

ÍNDICE GENERAL

Capítulo I. Aspectos básicos

1.1 Planteamiento del problema

En esta tesis de investigación se propone el diseño y desarrollo de un prototipo móvil llamado “Todo terreno”, en tal móvil proponemos un hardware mínimo necesario para implementar un algoritmo bioinspirado, bajo las características de un lenguaje de programación adecuado para el desarrollo de aplicaciones de inteligencia artificial y el manejo de dispositivos electrónicos para la robótica, proporcionando el API para tener acceso de forma transparente a las capacidades del móvil, los lenguajes ideales son C++ o Java.

Este diseño propone que la información necesaria pueda ser adaptada a algún otro evento deseable. El problema consiste en la múltiple interconexión de módulos en diferentes niveles de programación. El proceso de la construcción y la programación se divide en módulos como: comunicación, sensores, distancia, alimentación con baterías de litio, armado, elaboración de piezas y control. La modularización propuesta en este trabajo de investigación permite resolver el problema en etapas y dar mejoras específicas.

1.2 Motivación para el desarrollo

Cuando se inicia un proyecto en donde se pretende obtener resultados en un determinado ambiente lo más real posible a un ambiente de trabajo (escenario), es necesario contar con un sistema que permita ser adaptado o modificado para implementar algoritmos bioinspirados, así

como adecuar o mejorar el hardware para el funcionamiento esperado. Sin embargo, no todos los sistemas proporcionan la información necesaria e incluso no son de software libre, por lo consiguiente no es fácil de realizar una implementación acorde a las necesidades del proyecto.

En la Facultad de Ciencias de la Computación es necesario desarrollar un prototipo que brinde dichas necesidades ya que se cuenta con los conocimientos necesarios para diseñar y construir un prototipo, y así, brindar dicho desarrollo a la comunidad. Como consecuencia pretendemos que este desarrollo demuestra una aportación a la libertad del hardware y software en la comunidad de la facultad.

1.3 Objetivos, metas e hipótesis.

El propósito es la elaboración de un prototipo robot móvil “todo terreno” de propósito general que permita la implementación de un algoritmo específico, cubriendo las necesidades básicas tanto de software como de hardware, permitiendo al usuario final enfocarse a la solución del problema y la implementación del algoritmo a través de una interface en un lenguaje de alto nivel. Se habla entonces de un sistema embebido que contempla varios subsistemas de manera que se interconectan para trabajar como un sistema único. Para lograr dicho prototipo es necesario hacer una división modular, que muestre de manera ordenada el diseño, construcción e interconexión de todo el sistema. Se toman como base los siguientes módulos: comunicación, sensores, alimentación con baterías de litio, armado, elaboración de piezas en una impresora de 3D y control.

El desarrollo del prototipo que fue diseñado y propuesto en esta investigación tiene como objetivo el ser utilizado en un área de investigación y didácticas que permita ser utilizado en un espacio de trabajo controlado, así como, implementación de varios algoritmos que compartan los mismos objetivos entre ellos.

Se utilizó tecnología de diferentes fabricantes como Microchip y Raspberry, donde se contemplan dos microcontroladores de 8 bits y una tarjeta Raspberry Pi v.3 (mini PC), de 64 bits con cuatro núcleos ARM. Para adaptar las diferentes tecnologías se utilizó diferentes protocolos de comunicación y un diseño maestro-esclavo. Para proporcionar información acerca de su entorno se usaron diferentes tipos de sensores como: sensor inercial de tres ejes, opto-interruptor, Puente H, Motores DC y sensores de proximidad.

Se usaron módulos de software para la comunicación y control de los diferentes sensores, facilitando la obtención de los datos. Estos módulos se programaron en forma de funciones para solo ser llamados en una rutina principal.

Los protocolos de comunicación que soportan ambas tecnologías son comunicaciones seriales sincrónica como: I2C, así como comunicación asíncrona RS232. Los lenguajes de programación que son utilizados de manera nativa son el lenguaje c, para los microcontroladores, y el lenguaje Python para la mini PC.

El antecedente de este proyecto surge a partir del proyecto del laboratorio de enjambres y multiagentes en robótica donde se pretende realizar un estudio sobre el comportamiento en

grupos, usando como entidad a un robot simple que tengan la capacidad de una comunicación local entre robots y sensores para obtener información de su entorno. Se requiere de una cantidad considerable para el uso de este tipo de aplicaciones, por lo que se requiere de un prototipo móvil para la implementación de un algoritmo de este tipo, que cumpla con ciertas características para garantizar la operatividad del mismo.

Lo que se busca es una alta capacidad de cómputo y capacidad suficiente de almacenamiento, pero también considerar el tamaño y el consumo de energía. Sin perder de vista la capacidad de respuesta a un evento y la forma de comunicación para su manipulación. Lo más cercano sería una computadora personal, sin embargo, con la tecnología actual se puede tener un sistema mínimo que satisfaga los requerimientos buscados.

Una de las causas que limitan la compra de móviles para un proyecto, es que no se pueden modificar más allá de lo permitido por el fabricante y por lo consiguiente te limitan a lo que se puede realizar con dicho móvil.

La tendencia a los desarrollos de software y hardware es que sean de OpenSource y open hardware, donde cualquier persona puede modificar para mejorar o adaptar el móvil a su proyecto. Con este nuevo panorama de desarrollo se plantea tener una contribución al desarrollo tecnológico y la libertad del conocimiento.

Por lo tanto, se propone desarrollar un móvil con una arquitectura hibrida, utilizando microcontroladores y una mini PC para poder lograr un móvil que cumpla las expectativas antes mencionadas.

Es necesario contar con un equipo de cómputo con sistema operativo Windows 7, así como una distribución de Linux (por ejemplo, Ubuntu 14.04), para la instalación del software de control remoto (putty), el compilador PIC C (CCS) e instalar el sistema de Rasbian en una microSD con ayuda de Win32DiskImager. Se utilizan dos microcontroladores de la empresa MicroChip donde los modelos son: 18F4550 y 16F628A. Una tarjeta Raspberry Pi v3, sensores de aproximación, sensor inercial (girómetro y acelerómetro de 3 ejes cada uno), motores DC, Puente H y la carrocería del móvil.

Es necesario un multímetro y un osciloscopio. Como herramienta se utilizan una estación para soldar, así como estaño, grasa, pinzas de corte y de agarre. Para la interconexión de los sistemas se utilizan jumper y para la alimentación cinco baterías de litio.

Capítulo II. Estado del arte

Los algoritmos bioinspirados y genéticos fueron una propuesta como alternativa para solucionar problemas que son difíciles de resolver en la programación lineal. Esto para probar problemas y cómo se comporta con diferentes entradas de datos. Uno de los algoritmos genéticos implementados fue el algoritmo de evolución diferencial (AED). Se ha probado con la función de Michalewicz y Rosenbrock, estas son muy famosas en el ámbito de algoritmos bioinspirados, genéticos y heurísticos. De estas maneras se obtienen mínimos de cada función. Podemos expresar a estos vectores como un conjunto de límites inferiores y superiores. El AED en sus etapas más importantes:

- Mutación Diferencial $u_i^g = x_{r_1}^g + F\left(x_{\lfloor r_2 \rfloor}^g - x_{\lfloor r_3 \rfloor}^g\right)$
- Selección de agentes $x_i^{(g+1)}$
- Nueva generación $x_1^{g+1}, x_2^{g+1}, \cdots, x_{¿}^{g+1}$
- Probabilidad de recombinación o cruza Cr

Otro algoritmo que soporta el HSW es el de Harmony Search (HS), fue propuesto por Z.W. Geem [1], donde explica que el método de búsqueda armónica está inspirado en los principios subyacentes de la improvisación de armonía (HS) por parte de los músicos. El HS tiene las características distintivas de la simplicidad algorítmica y la eficiencia de búsqueda. Durante los últimos años, se ha utilizado con éxito en áreas como la optimización de funciones, el diseño de estructuras mecánicas, la optimización de redes y la optimización de sistemas de clasificación de datos. Yang en [2] aplicó la teoría de HS de forma algorítmica. En este trabajo se menciona al algoritmo Harmony Search, por sus buenos resultados que se obtienen en el contexto de los algoritmos metaheurísticos. Mencionamos lo fundamental de

HS y cómo funciona. Luego identifica las características de la metaheurística y analiza por qué HS es un buen algoritmo metaheurístico. Luego revisa brevemente otras metaheurísticas populares como la optimización de enjambres de partículas para encontrar sus similitudes y diferencias con HS.

Otro algoritmo que cubre la HSW es el de Geem y la propuesta de Yang, esto con el motivo de un buen entendimiento del funcionamiento, detalles y características que conlleva la implementación de este algoritmo HS. Así también se explica el concepto de la matriz armónica, su representación y uso.

Originalmente, las aplicaciones donde HS se evaluó por primera vez como un enfoque metaheurístico efectivo principalmente en el diseño de redes de distribución de agua, optimización de referencia, diseño estructural y problemas de enrutamiento de vehículos. En 2004, se publicó un diagrama de flujo de representación de HS y desde entonces se han realizado varios estudios para el desarrollo de nuevas variantes de HS y algoritmos híbridos con otros algoritmos meta-heurísticos. Desde entonces, la actividad en torno a este algoritmo ha aumentado considerablemente, abarcando su capacidad de aplicación a cada cartera heterogénea de escenarios de aplicaciones.

La matriz o memoria en la búsqueda armonía MHS (memory in the harminy search) ó HM (Harminy Memory) se inicializa comenzando por cada fila de la *MH* contiene los valores de las variables de diseño que son soluciones factibles seleccionadas al azar del grupo de diseño para una variable en particular. Por lo tanto, esta matriz tienen *n* columnas, donde *N* es el número total de variables de diseño y filas (matriz cuadrada). La HM se selecciona en el

primer paso del algoritmo. La HM es similar al número total de individuos en la matriz de población del algoritmo genético.

En [2] comenta que el elemento de la HM $x_{i,j}$ es el valor de la i-ésima variable de diseño en la j-ésima solución factible seleccionada al azar. Estos candidatos se ordenan de modo que el valor de la función objetivo correspondiente al primer vector de solución sea el mínimo. En otras palabras, las soluciones factibles en la HM se ordenan en forma descendente de acuerdo con su valor de función objetivo. Vale la pena mencionar que solo los elementos factibles son aquellos que satisfacen las restricciones establecidas por el algoritmo y se insertan en la matriz de memoria de armonía y los candidatos inviables se descartan del proceso.

Los algoritmos bioinspirados y meta-heurísticos basados en el algoritmo de Mirjalili [3], obtenemos el algoritmo y su estudio de este. En este artículo se propone un nuevo algoritmo de optimización metaheurístico inspirado en la naturaleza, llamado Algoritmo de Optimización de Ballenas (WOA), que imita el comportamiento social de las ballenas jorobadas. El algoritmo está inspirado en la estrategia de caza con red de burbujas. WOA se prueba con 29 problemas de optimización matemática y 6 problemas de diseño estructural. Los resultados de la optimización demuestran que el algoritmo WOA es muy competitivo en comparación con los algoritmos metaheurísticos de última generación, así como con los métodos convencionales.

Para resolver ciertos problemas es necesario conocer dos grandes disciplinas, como lo son: la informática y la electrónica. La importancia de dichas áreas es tan relevante, que

actualmente son utilizadas como base para el estudio de problemas de gran relevancia, de hecho, dichas disciplinas son complementarias.

En [9] se menciona que la computadora es la convergencia más popular de ambas disciplinas, sin embargo, menciona a los sistemas embebidos como un caso particular y, una aproximación del concepto está basado en un objetivo principal: resolver problemas específicos. En general es un dispositivo compacto (hardware y software especializado).

El prototipo fue diseñado con el objetivo de ser utilizado en un área específica y en un espacio de trabajo controlado, pero, que permita la implementación de varios algoritmos que compartan los mismos objetivos.

En [10] menciona que los parámetros: precios, tamaño y capacidad de procesamiento, han sido la base para el desarrollo de sistemas embebidos usando hardware de propósito general. La tecnología SOC (System On Chip) contiene varios módulos de hardware, teniendo como resultado un único módulo de propósito general. Gracias a los beneficios que se tienen con la tecnología SOC, los sistemas embebidos han tenido un avance muy importante ya que se puede tener un mayor nivel de cómputo en un menor tamaño y como consecuencia con un menor consumo de energía.

Dos factores importantes en el desarrollo de los sistemas embebidos han sido la producción masiva de circuitos integrados y como consecuencia el abaratamiento; estos dos factores han hecho posible el uso en diferentes ámbitos de la vida cotidiana. La tecnología utilizada es

consecuencia de un avance en el concepto de SOC, fabricación en masa y abaratamiento. Los componentes utilizados en este trabajo son un claro ejemplo del avance tecnológico que permite su adquisición y el libre uso para realizar "cualquier" prototipo donde sea necesario un alto procesamiento que utiliza poco espacio a un costo accesible.

En los robots es común el uso de sistemas embebidos para realizar tareas específicas, por ejemplo, en [11] se diseña un prototipo móvil con la finalidad de transportar objetos de manera automatizada trazando el camino de forma manual. Se basa en un robot móvil que utiliza una configuración tipo triciclo con tracción delantera (dos llantas y un motor en cada una) y una "rueda loca" en la parte trasera del robot móvil. El control se encuentra basado en un microcontrolador PIC16F873 y utiliza diferentes sensores como el CNY70 que es un sensor infrarrojo. En [12] se elabora un prototipo robot móvil autónomo enfocado a la exploración, donde a diferencia del anterior, cuenta con seis ruedas, de las cuales cuatro cuentan con motor cada una y se emplean más sensores como: brújula, GPS, ultrasonido y comunicación inalámbrica con Xbee Pro. Se utiliza un microcontrolador Atmel128 y define módulos importantes como subsistemas que en conjunto permiten al móvil tener la capacidad de conocer su entorno y obtener información de las condiciones del medio. En [13] se diseña un móvil para resolver un laberinto usando el hardware mínimo para realizar dicha tarea. Se utiliza un microcontrolador PIC16F877A, memoria FLASH, servomotores para la dirección y la locomoción de una llanta al frente y dos llantas posteriores de giro libre, varios sensores CNY70 y adiciona un PDA como interfaz de usuario.

En [14] se realiza la construcción de un robot móvil utilizando el concepto de "internet de las cosas" con el objetivo de una navegación autónoma, dado un espacio de trabajo. Utiliza software y dispositivos S*martThings,* Motores Motor Dynamixel MX-12W, sensor láser

Hokuyo URG-04LX-UG01, Raspberry pi 3 model B, Arduino uno R3 y Arduino ThingShield, cámara IP D-Link. Utiliza dos llantas con motor para la tracción delantera y dos llantas pasivas en la parte posterior. En este trabajo se utiliza un diseño que reúne varias características de los trabajos anteriores, se utilizan dos ruedas con motor cada una para la tracción trasera y el uso de una "rueda local" al frente, un arreglo de tres sensores infrarrojos para detectar obstáculos, el uso de LCD como interfaz de usuario, encoders, una arquitectura con dos microcontroladores y una Raspberry pi 3 model B, teniendo un conjunto de subsistemas donde existe el concepto de esclavo-maestro entre la Raspberry y los microcontroladores.

El sistema de navegación define en gran medida las técnicas y la complejidad del sistema, en [15] se menciona un estudio de diferentes robots móviles utilizando diferentes técnicas de control de movimiento que permite la trayectoria de un punto A a un punto B. También se menciona la importancia de sensores como método para obtener información del entorno y actuadores que permitan reaccionar ante un evento. El control de movimiento de un robot es importante para calcular la posición en el espacio en algún tiempo: *t*. En [16] propone solucionar el problema de ubicación de tres maneras, utilizando diferentes técnicas como: odometría, odometría utilizando técnicas de benchmark y odometría usando múltiples sensores. Cada una de estas posibles soluciones tiene sus características tanto de costo como de complejidad al implementar. Algunos de los errores que se tiene que considerar son: las llantas no son iguales, llantas no alineadas, resolución y tiempos de muestreo del codificador, piso irregular, el ruido de los dispositivos de medición, así como de las partes mecánicas al momento de ejecutar el movimiento indicado. La técnica del cálculo de ubicación basado en odometría con múltiples sensores, ayuda a minimizar los errores en tiempos largos, normalmente se utiliza un filtro de *Kalman*. Otra técnica para la ubicación es el uso de una

cámara como alternativa a la navegabilidad en tiempo “real”, en [17] se desarrolla un robot móvil que es dirigido con la ayuda de una retroalimentación visual, el cual es transmitido desde el robot móvil hasta una terminal, donde se realiza el control de navegabilidad. En [18] plantea una manera fácil y barata para la implementación de un algoritmo que sea capaz de estimar la posición utilizando sensores inerciales, en concreto se utiliza un microcontrolador y un acelerómetro de tres ejes que utiliza la aceleración captada por el sensor como base del algoritmo. En este trabajo se utiliza encoders con opto interruptores para calcular la distancia que recorre el robot móvil en línea recta y se considera un filtro para el ruido que se genera al estar en movimiento. Se utiliza el acelerómetro de tres ejes para calcular los grados sexagesimales que tiene que realizar el móvil, ya que primero realiza el recorrido en línea recta, se detiene y después realizar el giro a la izquierda o a la derecha (según la instrucción proporcionada), se considera los grados de error sobrantes o faltantes al terminar de realizar el giro. Se utilizan filtros para los datos del acelerómetro y un control PWM en conjunto con un control PID para estimar el giro del móvil.

En [19] se diseña y se construye un robot móvil para el transporte de carga radioactiva, se utiliza un móvil tipo triciclo con capacidad de aproximadamente 500 kg; de control semi autónomo y comunicación inalámbrico para el manejo. Se utiliza un microcontrolador MC68HC11 con lenguaje ensamblador. Se propone una serie de funciones principales que tiene el móvil para su operación y plantea un autómata para definir los estados según cada función en el que se encuentre. En este trabajo se utiliza el concepto de funciones definidas para el robot móvil como interfaces de programación, es decir, es transparente para el usuario proporcionando un API para utilizar en lenguajes de alto nivel.

Capítulo III. Marco Teórico

3.1 Introducción

La robótica móvil es un tema multidisciplinario al necesitar conocimiento de varias áreas para realizar desde el diseño hasta la construcción e implementación en algún campo de interés. La complejidad puede ser básica o compleja dependiendo de la aplicación del móvil y del entorno. Algunas disciplinas requeridas son eléctrica, electrónica, mecánica y computacional para resolver temas como motores, sensores, actuadores, microcontroladores, comunicación local y remota, interpretación de los datos, navegación, posicionamiento y control remoto. A pesar de que es un tema reciente ha tenido un crecimiento importante en los últimos años, recientemente el abaratamiento del hardware ha permitido que las universidades puedan desarrollar y construir robots móviles.

En [20] menciona que existen una gran cantidad de arquitectura de robots móviles dado que es un tema reciente y no se ha fijado una arquitectura estándar que pueda definir a un robot móvil, por lo que existen una gran variedad de trabajos con distintas propuestas pero que existen tendencias o partes en común que son básicas para cualquier desarrollo, así como las tecnologías y técnicas utilizadas.

En la arquitectura propuesta en este trabajo requiere del dominio de ciertos temas en específico. Se propone utilizar dos microcontroladores y una RaspBerry pi 3, por lo que el uso de protocolos de comunicaciones entre cada una de ellas es fundamental. Uno de los

protocolos fundamentales para el intercambio de información es el protocolo i2c que permite un esquema maestro-esclavo, múltiples dispositivos conectados al mismo bus (de dos líneas) como esclavos y le permite al dispositivo maestro empezar una comunicación utilizando la técnica de interrupción para empezar el intercambio de información. También se utiliza una comunicación serial (módulo USART), ya que es soportado en la mayoría de los microcontroladores para comunicación con el exterior y porque no se requiere acondicionamiento adicional para su funcionamiento. Al igual que el protocolo i2c, la comunicación serial requiere de interrupciones para su funcionamiento e iniciar un intercambio de información entre sistemas.

Se hace uso de la técnica de *polling* para obtener información del estado de los sensores y, a través de la información, realizar acciones como "avanzar". El robot móvil cuenta con dos motores DC (tracción trasera) y utiliza motorreductores en conjunto de técnicas de control como *PWM* y el control *PID* para realizar acciones como ir "adelante" y hacer uso de encoders para "parar" dado un evento. El uso del acelerómetro y un girómetro de tres grados de libertad permite obtener la información necesaria para realizar acciones como "girar".

Dado el uso de la técnica de *polling* y el constante cálculo que implica el uso del acelerómetro y girómetro, se utiliza un microcontrolador para censar usando la técnica de *polling* y comunicación serial con otro microcontrolador, que realiza los cálculos y realiza las acciones según la información obtenida. La función de la Raspberry Pi 3 es realizar las tareas de alto nivel y comunicación con otros dispositivos. Se puede implementar algoritmos usando lenguajes de alto nivel y que requieran mayor nivel de cómputo, por lo que es fundamental que se proporcione un API para el control del hardware. Para la comunicación con el microcontrolador que realiza las acciones se utiliza el i2c.

3.2 Protocolos de comunicación

3.2.1 Protocolo i2c

El protocolo I2C, permite una comunicación síncrona entre dispositivos, proporcionando un bus half-duplex, es decir, que la comunicación es bidireccional pero no simultáneo. Este protocolo fue desarrollado por Philips Semiconductors (actualmente NXP Semiconductors) para la comunicación de varios dispositivos utilizando únicamente un bus de dos líneas.

Las características de éste protocolo son:

- Sólo son necesarias dos líneas para la comunicación, conocidas como SDA (datos) y SCL (reloj).
- La cantidad de dispositivos que pueden conectarse al bus está limitada por la cantidad de 7 a 10 bits de direccionamiento y por una carga máxima de 400 pf.
- Se utiliza la técnica de colector abierto para utilizar la lógica AND, por lo que será necesaria resistencias PullUp.
- Permite una configuración multi maestro.
- Permite diferentes velocidades de comunicación (100 Kbps, 400 Kbps, 1Mbps y 3.4 Mbps)
- La comunicación es serial, es decir, bit a bit.

Hay varias versiones en lo que a la velocidad respecta [21], por ejemplo:

- Standard Mode: con una velocidad de hasta 100 kbit/s
- Fast-mode: con una velocidad de hasta 400 kbit/s
- Fast-mode Plus: con una velocidad de hasta 1 Mbit/s
- High-speed mode con una velocidad de hasta 3.4 Mbit/s.

Cada línea tiene un propósito, sin embargo, trabajan en conjunto para establecer la comunicación. La línea SDA es la que se encarga de transmitir los datos y la línea SCL es la que lleva los tiempos, es decir, es el reloj. Es oportuno indicar que cada dispositivo tiene definido una dirección (se indica en la hoja de datos) que es única, entonces si se tiene una red de dispositivos, cada uno de ellos será identificado por la dirección. En los microcontroladores existe una rutina para configurar una dirección específica.

En la figura 1 se muestra la estructura básica del protocolo I2C.

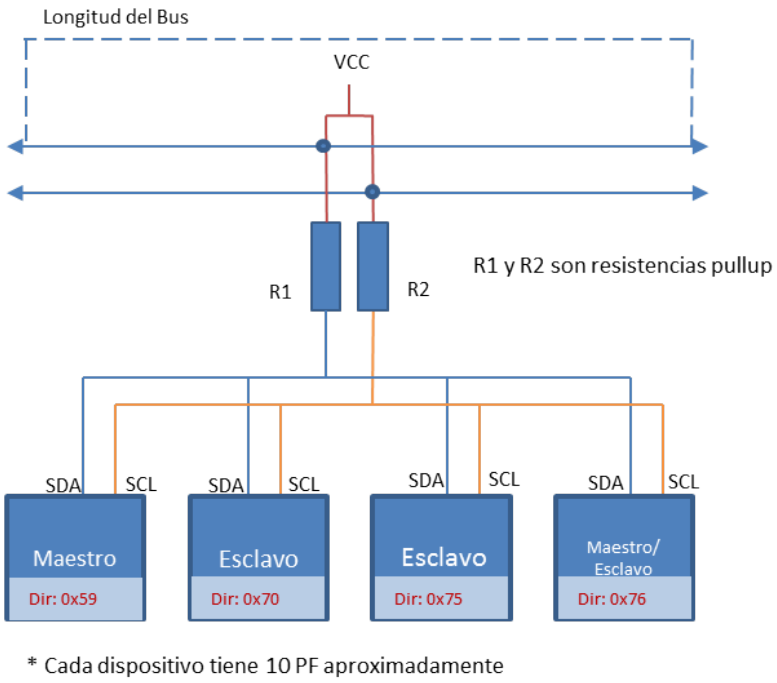

Figura 1: Esquema general del protocolo I2C

El maestro inicia la comunicación, por lo que es el encargado de generar la señal del reloj, para sincronizar la transmisión de los datos. Por lo tanto depende de la aplicación o el objetivo del dispositivo, sí sólo recibe información o recibe y transmite.

Para iniciar la comunicación el Maestro genera un pulso de inicio, conocido como START, seguido de 8 bits que indican la dirección del dispositivo a comunicarse (7 bits) y un bit para indicar si es escritura o lectura, entonces el dispositivo que coincida con la dirección responderá en el noveno pulso de reloj con un nivel alto (ACK) indicando de recibido. En caso de que no exista el pulso de reconocimiento, entonces se aborta la comunicación.

Si el Maestro indica la escritura, entonces enviará la información byte por byte y por cada byte el esclavo generará un pulso de reconocimiento, cuando no se genere dicho pulso entonces se indica que no puede recibir más datos.

En caso de que el Maestro indique la lectura, entonces dejará libre la línea SDA y el esclavo envía la información byte por byte hasta que el Maestro no genere el pulso de reconocimiento.

Para finalizar la comunicación el Maestro genera un pulso conocido como STOP, sin embargo se puede generar un segundo pulso START para reiniciar la comunicación.

La señal de reloj siempre la genera el Maestro, incluso cuando el maestro recibe información del esclavo o cuando el esclavo envía el pulso de reconocimiento. Se transmite el bit de mayor peso MSB (Most Significant Bit).

La línea SCL le da la capacidad al esclavo indicar que se encuentra ocupado (calculando, a la espera de un dato, etc.) llevando a nivel bajo la línea SCL, entonces el Maestro queda a la espera hasta que el esclavo libere la línea SCL para reanudar la comunicación. En la figura 2 se muestra dicho proceso.

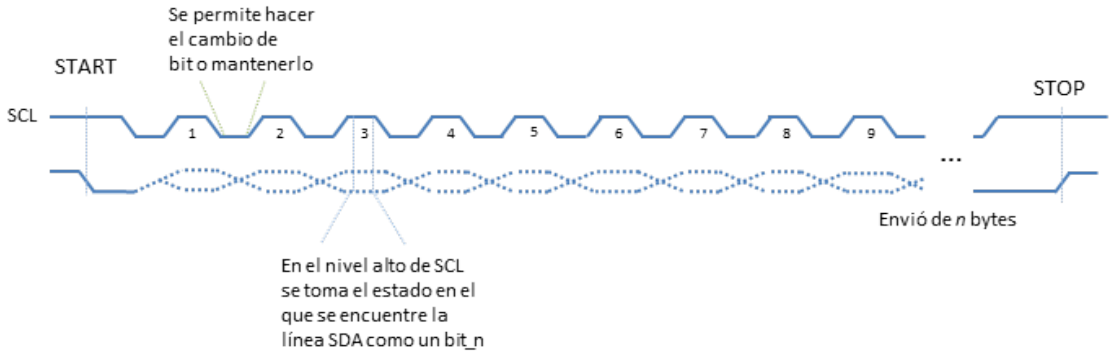

Figura 2: Protocolo de comunicación I2C

Si hay más de un Maestro, los dispositivos deben de soportar las implementaciones de **sincronización** y **arbitraje**, ya que puede ocurrir que el bus ya esté ocupado o ambos Maestros traten de comunicarse al mismo tiempo. En el primer caso es importante detectar que el bus este desocupado verificando que las líneas estén a un nivel alto (VCC). En caso de que inicien al mismo tiempo la comunicación, la línea SCL de ambos maestros se sincronizan, es decir, cuando un maestro tenga la línea SCL en un nivel bajo y el otro maestro tenga la

línea SCL en un nivel alto, el resultado final es un nivel bajo. Hasta que los dos maestros tengan la línea SCL en un nivel alto, el resultado final es un nivel alto, por lo tanto se aplica un AND en las líneas, teniendo como resultado un nivel alto cuando los dos maestros tenga la línea en alto y un nivel bajo en cualquier otro caso.

La intención es sincronizar la línea de reloj, independientemente de los diferentes tiempos de cada Maestro.

Para la línea SDA de cada maestro sucede el arbitraje que tiene como finalidad escoger a un sólo maestro que será el único que ocupe el bus, esto sucede ya que cada maestro verifica en el nivel alto de SCL el bit que envió (en el nivel bajo de la línea SCL), en caso de no ser el bit que envió pierde el arbitraje y deja el bus.

Existe la posibilidad de que dos o más Maestros puedan coincidir en todo el proceso, eso indica que empezaron al mismo tiempo y se dirigen al mismo dispositivo, es dicho caso no existe ningún problema.

En caso de que sólo exista un sólo Maestro no es necesario la **sincronización** ni el **arbitraje**.

Algunas consideraciones eléctricas en las especificaciones [21] se indica que para velocidades de 400 Khz es necesario no sobrepasar los 400 pF, por lo que se debe de

considerar la longitud del bus de comunicación (Capacitancia total) y el valor de las resistencias pull-up.

Para usar la resistencia pullup adecuada, es importante tener en cuenta dos requerimientos. El primer requerimiento es el tiempo máximo permitido para el cambio de estado de la señal (pasar de '0' a '1'), es decir, menor o igual a 300 nanosegundos para la velocidad de 400 Khz. El segundo requerimiento es la corriente mínima necesaria de 3 mA.

Para cumplir con estos dos requerimientos, en [22] se indican las ecuaciones para calcular la resistencia máxima y mínima que se puede usar.

Ya que no es tan fácil calcular la capacitancia total del bus, se puede hacer una estimación de 10 pF por dispositivo [22].

En [23] se indica que un valor de 4.7 KOhm es lo estándar para casi cualquier aplicación. Muchos sensores que soportan el protocolo I2C, generalmente primero se le indica la dirección a donde va a escribir los siguientes valores, para después reiniciar la comunicación y enviar los datos.

Un ejemplo de comunicación para realizar la configuración inicial del dispositivo sería:

1. Iniciar la comunicación indicando la dirección del dispositivo
2. Enviar la dirección de memoria a la cual el esclavo tiene que apuntar para posteriormente enviar datos
3. Finalizar la comunicación.
4. Iniciar la comunicación indicando la dirección del dispositivo.
5. Enviar los datos que se van a escribir en la dirección de memoria indicada anteriormente.
6. Finalizar la comunicación.

Un ejemplo de lectura y Escritura de datos sería:

7. Iniciar la comunicación indicando la dirección del dispositivo
8. Enviar la dirección de memoria a la cual el esclavo tiene que apuntar para posteriormente escribir o leer datos del esclavo.
9. Finalizar la comunicación.
10. Iniciar la comunicación indicando la dirección del dispositivo
11. Recibir o enviar los datos de la dirección de memoria indicada anteriormente.
12. Finalizar la comunicación.

3.2.2 USART (Universal Asynchronous Receiver Transmitter)

Es un módulo que viene en la mayoría de microcontroladores para transmitir y recibir bits en serie permitiendo utilizar un estándar como por ejemplo el RS232 (utilizado para ser compatible con el ordenador) para ser compatible con otro dispositivo. Sin embargo, se puede utilizar sin emplear ningún estándar adicional cuando el otro dispositivo es compatible con el USART y los niveles de voltaje. Esto se aplica cuando existe una comunicación entre microcontroladores (que ambos soporten el USART y sus niveles de voltajes sean los mismo) aunque puede cambiar.

Comúnmente el periférico USART se complementa con un circuito externo para soportar un protocolo de comunicación como es el caso del RS232 para la comunicación con una Computadora. Permite la comunicación síncrona/asíncrono permitiendo el half-dúplex/full-duplex respectivamente. En la comunicación síncrona se utiliza una línea para el reloj y otra línea para los datos y en la comunicación asíncrona se utiliza una línea para transmitir y otra línea para recibir datos, por lo tanto, el reloj es interno en cada dispositivo y deben de tener la misma frecuencia y fase.

Se permite el envío de tramas compuestas por 8 o 9 bits, y el noveno bit tiene diferentes funciones y una de ellas es la de utilizarlo como bit de paridad (se verifica por software), así como también se puede utilizar para comunicarse con un dispositivo cuando existan varios en el bus.

En general es muy común utilizar la comunicación asíncrona, pero dependiendo de la longitud del bus se puede utilizar la comunicación síncrona (distancia corta), en caso contrario se utiliza la comunicación asíncrona (distancia larga).

En el modo síncrono se envían bytes sin límite y en el caso del modo asíncrono por cada byte que envié tiene que tener un bit de inicio y un bit de fin, en cualquier caso, cada dato es representado por 8 bits es decir un byte. En el primer modo se puede configurar como Maestro o Esclavo (envían o reciben información de forma no simultánea). En la configuración del Maestro, éste genera la señal de reloj e inicia y finaliza la comunicación, en la segunda configuración el Esclavo recibe la señal de reloj y dependerá del Maestro para enviar o recibir información. En el modo asíncrono el dispositivo tiene la capacidad de enviar y recibir información al mismo tiempo por lo que el reloj lo tiene cada dispositivo internamente. En la figura 3 se muestra el envío y recepción de los datos en el modo asíncrono.

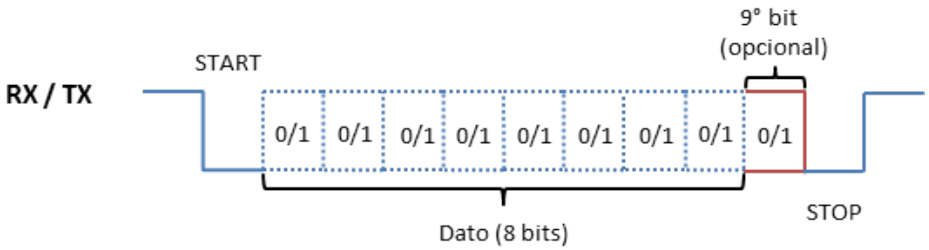

Figura 3: Transmisión y recepción de bytes en modo asíncrono

Un parámetro configurable es la velocidad de transmisión que es expresada en Baudios, es decir, bits por segundos. La velocidad estándar es 9600 bits por segundo, aunque se pueden configurar otras velocidades. Los Baudios no siempre coinciden con los bits por segundo, de hecho, no significan lo mismo, se puede transmitir 9600 estados de la señal (baudios) pero transportar 19200 bits por segundo. Otro concepto que se tiene que tener en cuenta es la cantidad de información que se transmite por segundo, ya que de los 9600 bits existen 1200 bytes, sin embargo, se utilizan dos bits (tres bits en caso de utilizar el noveno bit) para indicar el inicio y el fin de un byte, por lo tanto, existen 960 bytes por segundo de datos.

El noveno bit se puede utilizar como bit de paridad para corregir errores o saber que el byte llegó mal y por lo tanto solicitarlo nuevamente. Esta comprobación se realiza a nivel de software.

3.3 Técnicas de control

3.3.1 Modulación por ancho de pulsos

El PWM (***Pulse Width Modulation***) es una técnica que controla el tiempo en el que la señal permanece en alto o en bajo. Una aplicación común es el control de la velocidad de un motor de corriente directa, ya que, al controlar el tiempo de un estado alto de la señal, será el tiempo en el que el motor funcione, por lo tanto se manejan los parámetros de frecuencia y ciclo de trabajo. El ciclo de trabajo indica la cantidad de tiempo que la señal estará en un nivel alto de un periodo y se expresa en porcentaje. En la figura 4 se puede observar el funcionamiento.

En el ejemplo del motor de corriente continua, cuando se encuentra en funcionamiento y se le desconecta de la fuente, se deberán de considerar la inercia, y en caso de invertir la polaridad cambiará la dirección del giro.

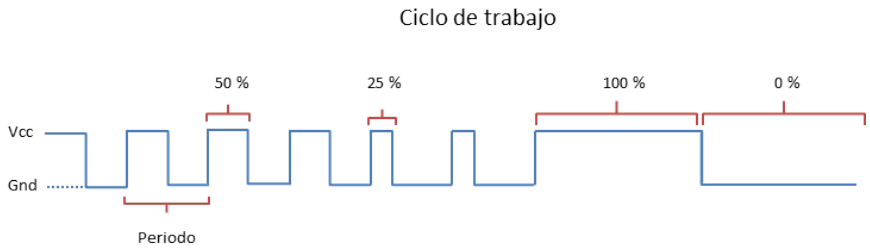

Figura 4: Funcionamiento del PWM

3.3.2 Control Proporcional Integral Derivativo

El control PID es una técnica ampliamente utilizada para el control de sistemas. Se basa en el principio de la realimentación y cuenta con la capacidad de operar con ruido y reaccionar anticipadamente tomando en cuenta posiblemente valores anteriores, actuales y estimando los valores siguientes. El objetivo del control PID es obtener el mejor control posible de un sistema, aproximándose a un control ideal pese al ruido.

3.3.3 Control On-Off

Permite tener un control básico, puede ser implementado únicamente con hardware, de hecho al inicio, los controladores así fueron implementados. A pesar del poco control del

valor deseado, aún al día de hoy tiene un importante uso y permite entender una transición tecnológica.

El control On-Off únicamente tiene dos posibles salidas sin importar que "tanto" error exista entre el valor obtenido y el valor deseado, es decir, hay o no hay error y eso se traduce en una acción, Encendido o Apagado, 1 o 0, etc. La magnitud del error es variable y depende principalmente de la velocidad con la que se alcance el valor deseado, las variaciones de la variable medida dentro de un intervalo de tiempo (por ejemplo si el incremento de temperatura no es uniforme), el tiempo necesario para tomar una acción opuesta, la precisión y sensibilidad del sensor, entre otros aspectos.

En [28] se define la siguiente ecuación:

$$e = Ysp - Y \qquad \textit{(ePID 1)},$$

Donde e es el error obtenido de la diferencia del valor deseado y el valor obtenido. El valor de la variable medida (Y) siempre tendrá una variación en el tiempo respecto a un valor deseado (Ysp).

Notar que sí $e = 0$ la variable medida será igual al valor deseado, por lo que no se tiene definido una acción, sin embargo, sí se sólo se reacciona cuando el error es diferente de cero, nunca tendremos problemas. Al reaccionar cuando el error sea diferente de cero, tendremos muchas conmutaciones entre on y off, ya que por mínimo que sea el error se tomará una acción.

Para solventar una excesiva conmutación se tiene el concepto de histéresis. Ahora el valor de referencia es un intervalo con un valor máximo y mínimo, expresando que:

- $Ysp - e1 < Ysp < Ysp + e2.$

Tenemos ahora que *e1* y *e2* pueden ser modificados de acuerdo a las necesidades del sistema y agregando dicho intervalo, podemos definir una tercera acción. Quedarse con el valor actual (On u Off) y no hacer nada mientras la variable medida permanezca dentro del intervalo. Fuera del intervalo se comporta como se ha definido anteriormente. Se consigue reducir la cantidad de conmutaciones y se tiene una mayor variación que depende de los límites *e1* y *e2*.

Notar que en la figura 5 se muestra un evento que se tiene que considerar: **la frecuencia.**

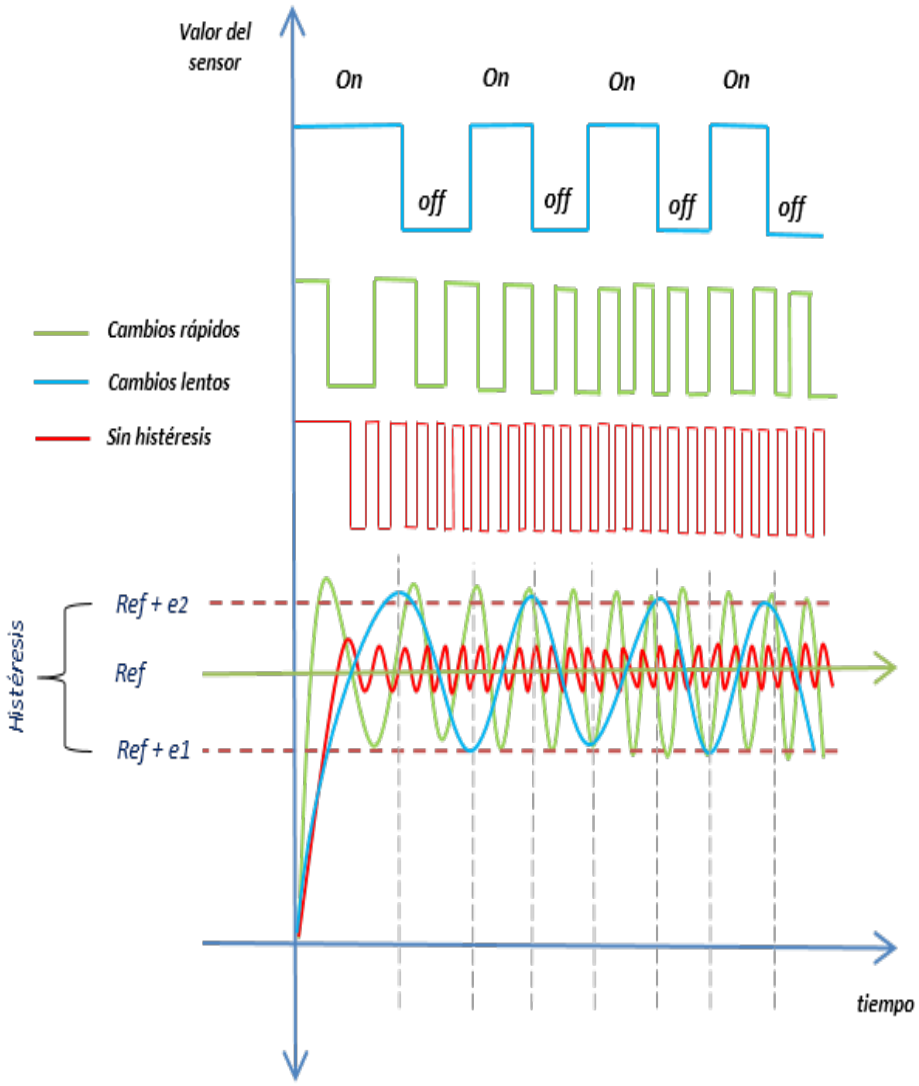

Figura 5: Diagrama de diferentes formas de control On/Off

3.3.4 Control Proporcional

Al ampliar la resolución de las posibles acciones a tomar respecto al error calculado, se plantea el concepto proporcional, lo que conlleva a tomar acciones proporcionalmente según la magnitud del error.

En [28] el control proporcional se puede definir como:

$$u = kp * e + m. \qquad \textit{(ePID 2)},$$

dónde: **kp** es la ganancia proporcional, **e** es el error que hay entre la variable medida y el valor deseado, **m** es la desviación para cuando e = 0 (reset manual).

La ganancia proporcional es una constante que define en muchas formas el comportamiento del controlador y tiene un valor límite para un control estable. En [28] la define como la pendiente de la recta característica que se forma por la diferencia del valor máximo y mínimo de la salida (definida entre 0% y 100%) y el valor máximo y mínimo del error permitido. De esta manera se puede ajustar el margen de error permitido, así como la acción mínima y máxima deseada, no necesariamente se requiere una acción de 0% a 100%, puede que se requiera mantener una acción entre el 25% a un 75% con un error total del 10%.

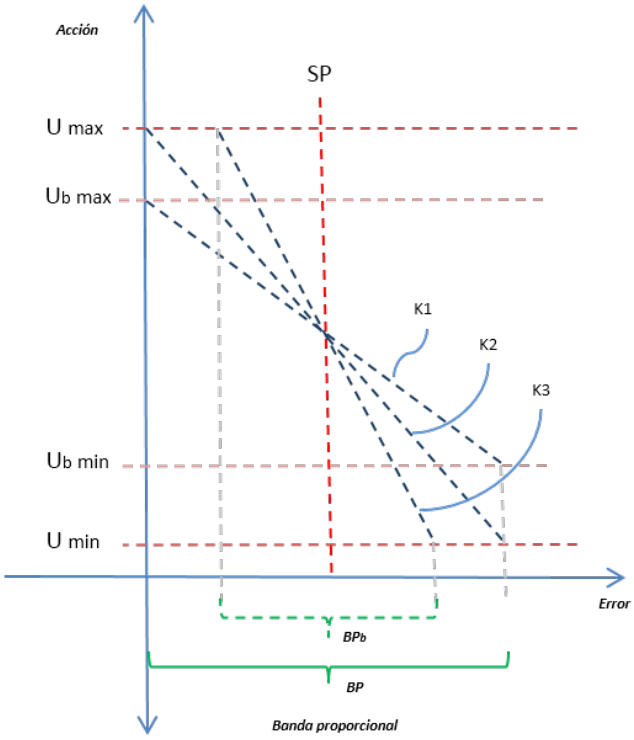

Figura 6: Diferentes pendientes de rectas de interés según los límites requeridos para el sistema

En muchas literaturas se maneja la banda proporcional que está definida de la siguiente manera:

$$Bp = 100 / Kp \qquad (ePID\ 3),$$

Que permite representar de una forma equivalente a la ganancia, sin embargo, da un nuevo significado y se refiere a el porcentaje como la magnitud de cambio de la variable de proceso requerido para tener una acción del 100% en la salida.

El uso del control proporcional da un mejor control respecto al on/off, sin embargo, puede no ser suficiente ya que se requiere un ajuste para llegar al valor deseado (reset manual), esto se debe al valor límite de la constante proporcional para ser un control estable. Por lo tanto, siempre existirá un error permanente conocido como offset.

Cuando el sistema se encuentra en constantes cambios que perturban el comportamiento, el control proporcional tiene que tener un *reajuste automático* ya que es la única forma de poder alcanzar el valor deseado.

3.3.5 Control Proporcional con acción Integral

Para quitar el offset que genera el control proporcional se utiliza el reajuste automático que plantea el uso de una acción integral, en [28] se define de la siguiente manera:

$$u_t = u_{t-1} + K_i * e. \qquad \text{(ePID 4)},$$

donde: k_i = es la ganancia integral, se define como $K_i = (K_p/T_i)$, donde Ti es el tiempo integral, e = es el error que hay entre la variable medida y el valor deseado.

Se utiliza la ecuación ePID4 para un cálculo computacional, ya que es un cálculo discreto. En [28] la ecuación completa se define como:

$$u_t = K_p * e + u_{t-1} + K_i * e + m. \qquad \text{(ePID 5)},$$

Tener en cuenta que se requiere de dos ciclos de ejecución para calcular la ecuación. Al inicio se contempla la ejecución anterior como cero.

3.3.6 Control Proporcional con acción Derivativo

La variabilidad del error en el tiempo por cambios bruscos de la variable a controlar, la acción proporcional e integral no resuelve convenientemente, por lo que es necesario el uso de una acción derivativa para responder rápidamente a dichos cambios. En [28] se define de la siguiente manera:

$$u = K_d * [(PV_t - PV_{t-1}) - (PV_{t-1} - PV_{t-2})]. \qquad \text{(ePID 6)},$$

donde: k_d = es la ganancia integral, se define como $K_d = K_p * T_d$, donde Td es el tiempo derivativo.

La ecuación completa se define como:

$$u_t = K_p * e + K_d * [(PV_t - PV_{t-1}) - (PV_{t-1} - PV_{t-2})] + m. \quad \text{(ePID 7)},$$

Tener en cuenta que se requiere de tres ciclos de ejecución para calcular la ecuación.

3.4 Técnicas de sondeo

3.4.1 Polling

Ésta técnica es una de las primeras de manera natural en utilizarse para el conocer los cambios de estado en el hardware. Como si se tratase de una pregunta, cada ciclo de ejecución del programa se tendría que realizar la pregunta para cada estado que se requiera conocer. La técnica a emplear resuelve el problema de conocer el estado para después actuar con dicho valor, sin embargo, presenta algunas desventajas.

Hay dos desventajas principales que hacen esta técnica no adecuada a ser usada para ciertos problemas. La primera es que cada ciclo del programa se realizan las mismas preguntas para conocer los estados, esto puede causar instrucciones que se ejecutan innecesariamente. La segunda desventaja es que se puede perder información en ese ciclo de ejecución del programa. Entre las preguntas para conocer el estado, existe un tiempo *x*, que para muchos problemas a resolver es suficiente para que el estado pueda cambiar de estado.

3.4.2 Interrupciones

Las interrupciones permiten detener la ejecución de un programa para “brincar” a una rutina especial que ha sido programada previamente, cuando se termina de ejecutar la rutina de dicha interrupción, regresa a la siguiente instrucción en la que se quedó el programa principal antes de ser interrumpido, para continuar con la ejecución.

Un microcontrolador puede tener diferentes interrupciones, por lo tanto, es natural pensar sobre la jerarquía de dichas interrupciones e identificar quién generó la interrupción. Cuando se genera una interrupción se evalúa la prioridad (esto realmente funciona cuando se ejecutan dos interrupciones a la vez) y por lo tanto se pregunta quién generó la interrupción a través de unas banderas del vector de interrupciones. Cuando se termina de ejecutar la rutina de dicha interrupción, entonces se reinicia la bandera poniéndola a cero.

Cuando se genera una interrupción y se está ejecutando una rutina de interrupción de la misma jerarquía, entonces no se interrumpe, sin embargo, una interrupción de alta prioridad puede interrumpir a una de menor prioridad pero no al revés.

Normalmente el microcontrolador permite diferentes interrupciones y dependiendo de la cantidad de jerarquías que tenga, es la cantidad de vectores de interrupción (normalmente tiene una dirección fija por el fabricante). Siempre que se genere una interrupción y dependiendo de la prioridad irá a un vector de interrupción para saber la dirección de memoria donde se encuentra la rutina que servirá a dicha interrupción. Estando en la rutina se pregunta quién generó la interrupción.

3.5 Tarjetas de Desarrollo

3.5.1 Microcontrolador PIC16F628A

Es un microcontrolador de la familia PIC16 de 8 bits con características muy específicas, ideal para implementar una solución que requiere ser programado de forma aislada y que requiera recursos muy específicos. Soporta una frecuencia máxima de 20 Mhz. Cuenta con una pila de 32 niveles y soporta 20 fuentes diferentes de interrupción. Cuenta con 224 Bytes de RAM, una memoria EEPROM de 128 Bytes y 2 KB para el programa. Soporta 3 timers, 1 módulo de PWM y USART y 2 comparadores. Por sus dimensiones puede ser integrado fácilmente.

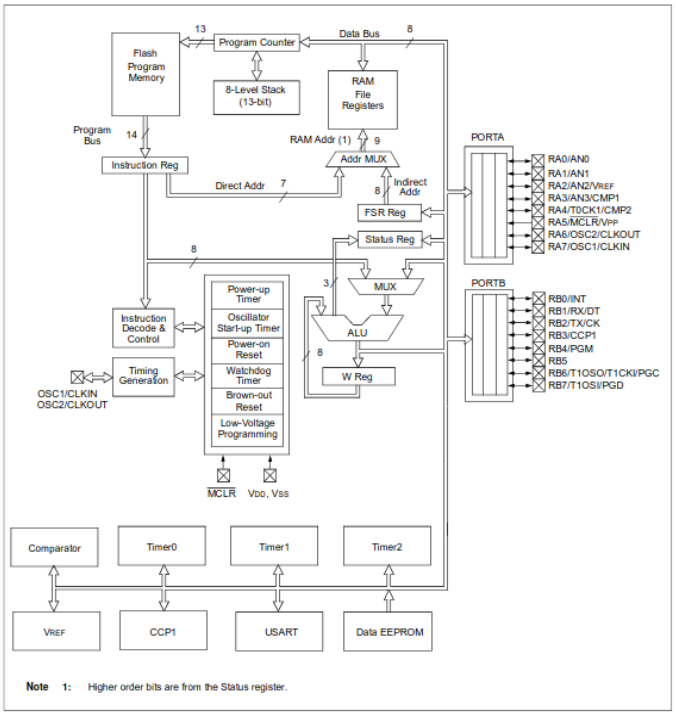

Figura 7: Diagrama de bloque de la familia PIC16, la imagen fue tomada de [30]

3.5.2 Microcontrolador PIC18F4550

Es un microcontrolador de la familia PIC18 de 8 bits con múltiples prestaciones, ideal para proyectos en la etapa del prototipo. Soporta una frecuencia máxima de 48 Mhz que incluye un High Precision PLL (Phase Locked Loop). Cuenta con una pila de 32 niveles y

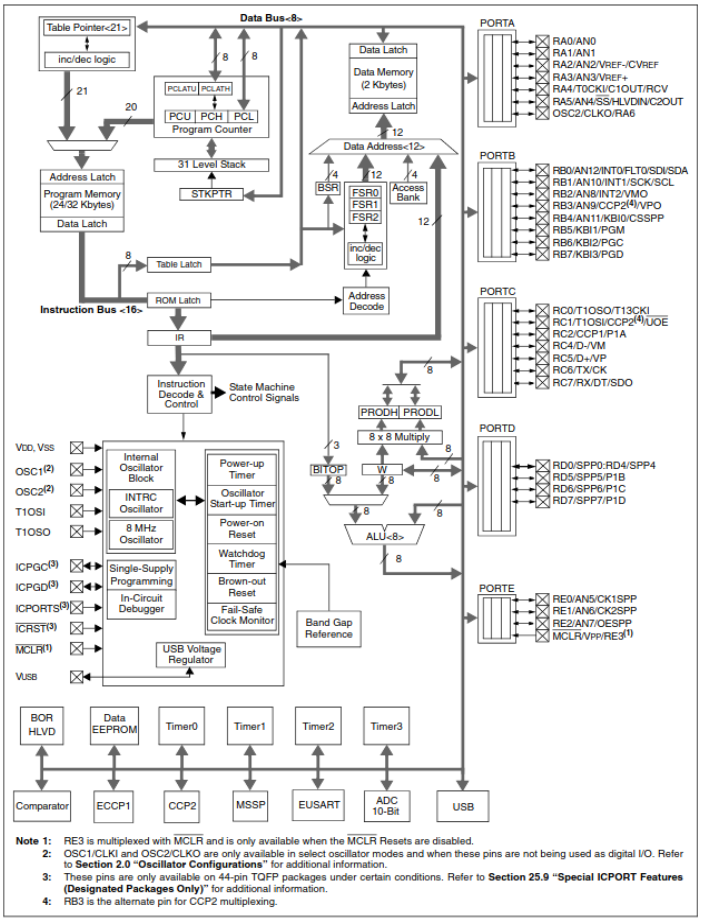

Figura 8: Diagrama de bloque de la familia PIC16, la imagen fue tomada de [31]

soporta 20 fuentes diferentes de interrupción. Cuenta con 2 KB de RAM, una memoria EEPROM de 256 Bytes, 16 KB de memoria para las instrucciones y 32 KB de memoria para

datos del programa. Soporta 4 timers, 1 módulo de PWM, 1 módulo MSSP (para ser usado como I2C o SPI) y soporte para Enhanced USART. Adicional soporta el USB, SPP y tiene un módulo Análogo a digital de 10 bits.

3.5.3 Raspberry Pi V3

La Raspberry Pi v3 cuenta con un procesador ARM de 64 bits, tiene un módulo wifi 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Salida de video Full HDMI, soporte H.264, MPEG-4 decode, OpenGL ES 1.1, 4 puertos USB 2.0, un puerto de Gigabit Ethernet y un puerto para una microSD. Cuenta con 1 GB de memoria LPDDR2 SDRAM, Soporta un sistema operativo llamado Raspbian que es una distribución de GNU/Linux. Permite programar en lenguajes de alto nivel, como Python, JavaScript, C++ y por supuesto el lenguaje C, entre otros.

El chip que está incluida en la Raspberry es un Broadcom BCM2837B0 [32] que es un system-on-chip (SoC) e incluye un Cortex-A53 (ARMv8) de 64 bits a 1.4 Ghz.

Este SoC proporciona un *GPIO header* de 40-Pin, lo que permite puertos UART, SPI, DMA, puertos de entrada/salida de propósito digital (GPIO), sistema de interrupciones, PCM, PWM, sistema de Timers, USB, lo que lo hace realmente un sistema de multi propósito.

3.6 Sensores y actuadores

3.6.1 Puente H l298 y el motor DC.

Este dispositivo tiene dos puentes H, lo cual permite manejar dos motores (incluso otro tipo de dispositivos de carga inductiva) y opera en un rango de 4 a 35 v con un máximo de 2 A. por cada puente H. Soporta la señal lógica TTL y tiene dos entradas independientes para habilitar o deshabilitar el suministro de corriente al motor DC [33].

El uso más común es para controlar el sentido del giro de un motor DC, ya que al estar configurado el puente H como interruptores.

Para controlar los motores se hace uso de los habilitadores independientes para cada motor y con ayuda de un control PID se logra tener un control estable en los motores DC.

3.6.2 Sensor Infrarrojo Sharp GP2Y0A21YK0F

El sensor analógico infrarrojo Sharp GP2D12 tiene la capacidad de medir entre 10 y 80 cm. [34], sin embargo, existen factores externos (por ejemplo, la reflexión con objetos cercanos, espejos, etc.) que reducen el rango de medición o generan mediciones incorrectas, por lo tanto, se tienen que evitar o incluir algún tipo de tipo de filtro (puede ser a nivel de software o hardware, incluso hacer una combinación).

El fabricante proporciona una gráfica donde muestra una comparación entre la distancia y el voltaje que entrega, en el rango de los 0 a los 3 volts y también recomienda conectar un capacitor de 10 microfaradios entre VCC y GND. El sensor tiene un tiempo de respuesta de 39 ms para entregar una nueva lectura y se recomienda tener una colocación correcta.

Ya que el sensor es analógico, el nivel de voltaje que entrega tiene que ser procesado y realizar la conversión a una distancia, en [35] indica el procedimiento para obtener un polinomio de grado 3 utilizando muestras del sensor y aplicando mínimos cuadrados para obtener dicha ecuación.

3.6.3 Sensor inercial MPU6050

El sensor MPU 6050 es un sensor de medición inercial de 6 grados de libertad, es decir, está compuesto por un girómetro y un acelerómetro, ambos de 3 ejes, de tecnología MEMS (Microelectromechanical System). También incorpora un sensor de temperatura y un DMP (Digital Motion Processor) que es el encargado de procesar la información de los 6 ejes utilizando algoritmos complejos propietarios del fabricante. Permite agregar un Magnetómetro de 3 ejes para realizar procesos más complejos y proporciona un FIFO de 1024 bytes para obtener datos de dicho tamaño [36].

En la tabla 1 y 2, se muestran las características y rango de medición, así como, los filtros soportados por el dispositivo.

Tabla 1. Características del acelerómetro y girómetro.

Parámetro	Girómetro	Acelerómetro
Voltaje de Operación	3.3 v	3.3 v
Consumo de corriente	5 mA	500 microA
Filtros Programables	Filtro paso bajo	Filtro paso bajo
Salida digitales	Tres ejes X, Y y Z	Tres ejes X, Y y Z
Rango de escala programable	+/-250, +/-500, +/-100 y +/-2000 º/Seg (dps)	+/- 2g, +/- 4g, +/- 8g y +/- 16g
Protocolos de comunicación	I2C (400 KHz) y SPI (1 MHz)	I2C (400 KHz) y SPI (1 MHz)
Resolución de cada eje	16 bits por cada eje.	16 bits por cada eje.

Tabla 2. Filtros programables para acelerómetro y girómetro.

DLPF_CFG	Acelerómetro (Fs = 1kHz)		Girómetro		
	Banda de frecuencia (Hz)	Retardo (ms)	Banda de frecuencia (Hz)	Retardo (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	Reservado		Reservado		8

La frecuencia de muestreo del acelerómetro es de 1 KHz para cualquier configuración del filtro. En [37] se muestra un diagrama de bloques de los componentes del MPU6050 que se

muestra en la figura 9, donde se muestra que los ejes del acelerómetro y del girómetro pasan por un ADC para después pasar a los registros de donde se tomara lectura con el protocolo I2C o SPI. Los datos se pueden obtener desde los registros en un valor en “raw”, es decir, sin algún previo procesamiento o tomar los datos ya procesados con algún algoritmo privado del fabricante utilizando el DMP y accediendo a los datos a través de los registros del FIFO de 1024 bytes.

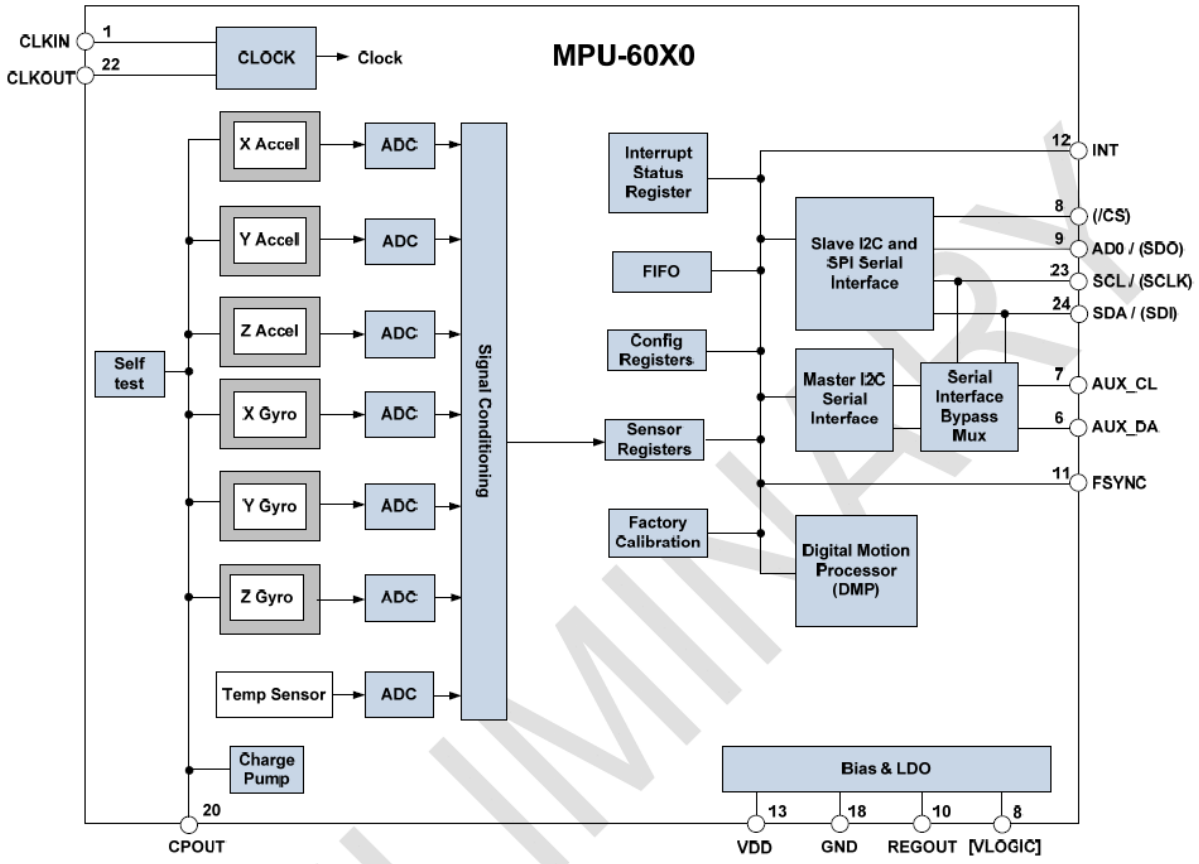

Figura 9: Esquema interno del MPU-6050, la imagen fue tomada de [37]

Los registros más relevantes para obtener los datos en crudo son los siguientes:

- GYRO_CONFIG, Configuración del girómetro
- ACCEL_CONFIG, Configuración del acelerómetro
- CONFIG, configuración del filtro.
- PWR_MGMT_1, Habilitacion/Deshabilitacion del modo sleep
- ACCEL_XOUT, parte alta de los 16 bits del eje x del acelerómetro
- ACCEL_XOUT, parte baja de los 16 bits del eje x del acelerómetro
- ACCEL_YOUT, parte alta de los 16 bits del eje y del acelerómetro
- ACCEL_YOUT, Parte baja de los 16 bits del eje y del acelerómetro
- ACCEL_ZOUT, Parte alta de los 16 bits del eje z del acelerómetro
- ACCEL_ZOUT, Parte baja de los 16 bits del eje z del acelerómetro

El acelerómetro mide la aceleración al que se someta el dispositivo, sin embargo, también mide la aceleración al que todos estamos sometidos que es la aceleración gravitacional de aproximadamente $9.8 m/s^2$ por lo que permite saber la inclinación de los demás ejes como se muestra la figura 10.

$$1g = \sqrt{Rx^2 + Ry^2 + Rz^2}$$

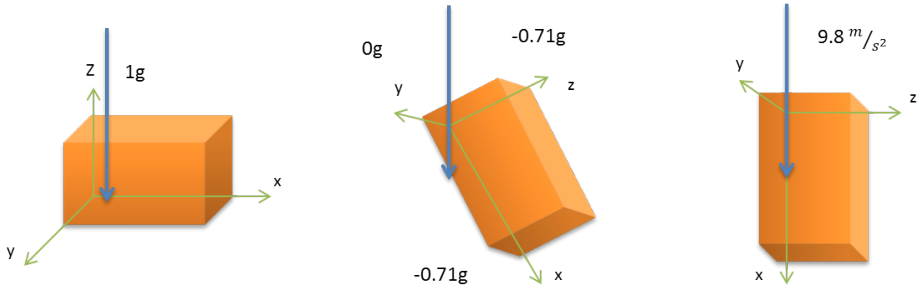

Figura 10: La fuerza gravitacional en un plano de 3 ejes.

3.6.4 Optointerruptor

El módulo encapsulado consiste en un diodo emisor de infrarrojo y un fototransistor que es el encargado de recibir dicha señal. El funcionamiento es a 5 V. de alimentación y tiene un consumo de 75 mW [40]. Es recomendable que se acompañe de un comparador de voltaje (LM393) para tener en la salida 0v o en este caso 5V (Vcc). Cuando se interrumpe la emisión del infrarrojo al fototransistor se genera una señal en la salida, sin embargo, se genera ruido con lo que ocasiona pulsos que son rebotes. El microcontrolador toma lectura de dichos pulsos y se ayuda del LM393 para obtener pulsos estables (0-Vcc). Una técnica que se utiliza es que el microcontrolador espere 20 ms después del primer pulso para obtener una nueva lectura.

Capítulo IV Arquitectura propuesta del móvil.

4.1 Interconexión de los subsistemas.

El problema se dividió en tres tareas principales que son: control de la distancia a recorrer, control de giro y dirección y comunicación al exterior.

La comunicación juega un papel vital en el proyecto y para que exista una correcta coordinación, se ha definido a la Raspberry PI v3 como el rol de “Maestro”, es decir, coordina los otros subsistemas. Para llevar a cabo el intercambio de instrucciones se implementó una comunicación directa con el microcontrolador PIC18F4550 utilizando el protocolo I2C a 100 Khz. Dado que el nivel de voltaje de ambos sistemas es incompatible, ya que el microcontrolador funciona a 5v y la Raspberry Pi v3 opera a niveles de 3.3, se utiliza un circuito adicional que permite realizar la conversión del voltaje a su respectiva compatibilidad. Se utiliza el componente 2N7000 (mosfet) que va a permitir la comunicación bidireccional y será el encargado de cambiar los niveles de voltaje con ayuda de los pull-up, ver la figura 11. Es necesario que se acoplen las tierras del microcontrolador y la RaspBerry Pi.

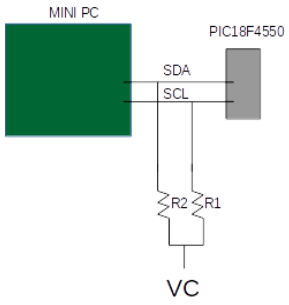

Figure 11: Circuito para el protocolo I2C entre la Raspberry y el microcontrolador.

El microcontrolador PIC18F4550 coordina al microcontrolador PIC16F627A, es decir, es "esclavo" de la Raspberry y "maestro" del PIC16F627A. La comunicación entre los dos microcontroladores PIC es utilizando el protocolo serial RS232 a 900 baudios. Es el encargado de controlar directamente el hardware, por ejemplo, para realizar: el paro total del móvil, la velocidad, la distancia a recorrer y la dirección. En consecuencia, obtiene la información de todos los sensores, procesa y toma decisiones de acuerdo a las instrucciones recibidas. Siempre se encuentra disponible para recibir comandos de la Raspberry o para enviar información solicitada.

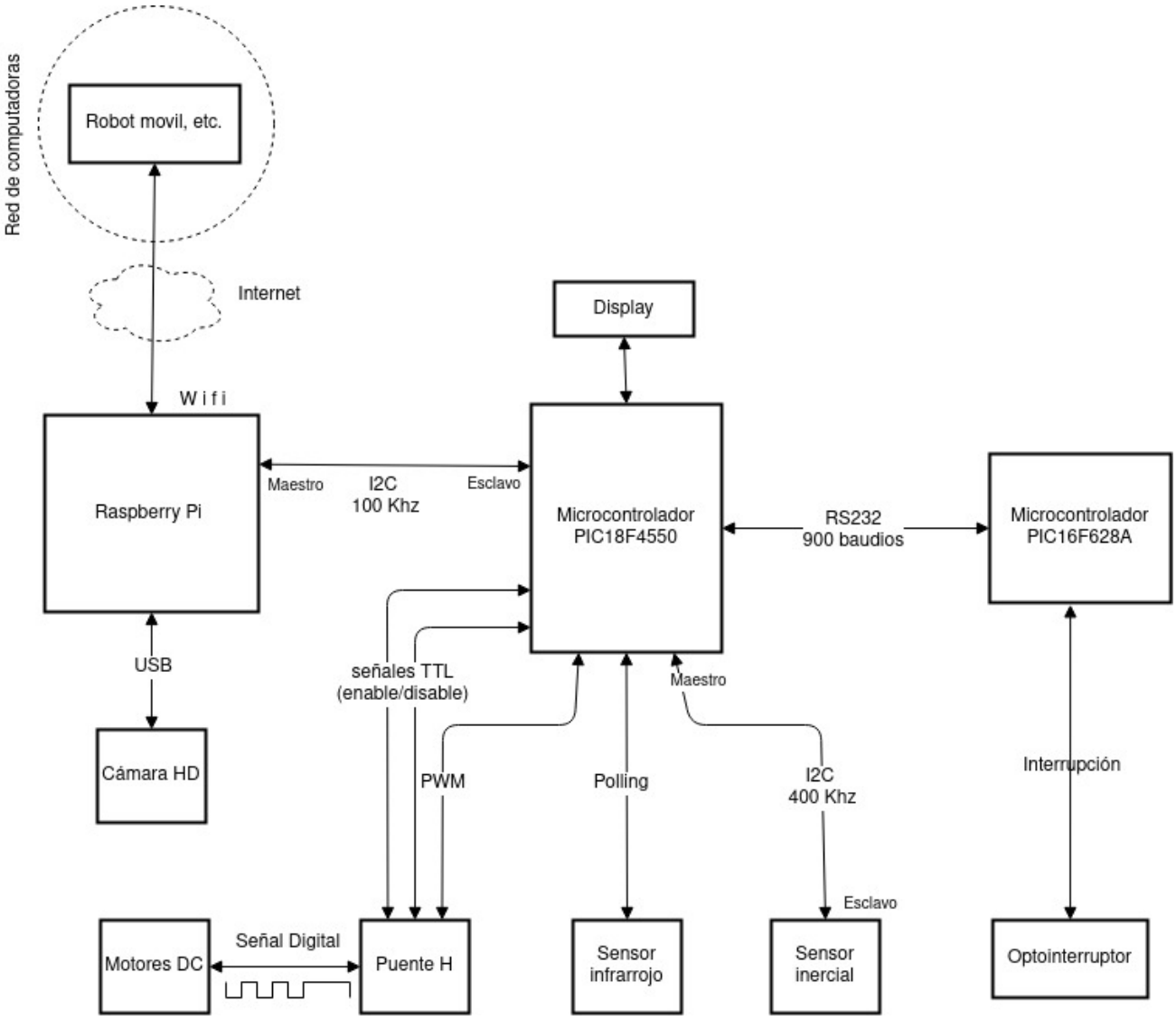

Figura 12: Arquitectura del carrito "todo terreno"

4.2 Diseño del carrito

El objetivo de la Raspberry PI v3, es la coordinación, centralización de datos recolectados y proporcionar interoperabilidad a través de una interface (API). Cuenta con una cámara de 2 Megapíxeles conectada a través de un puerto USB tipo A para uso futuros.

El PIC18F4550 es el microcontrolador que coordina la mayoría de los sensores para obtener y manipular las acciones más básicas del sistema, por ejemplo, la de avanzar, parar, etc. Es el encargado de proporcionar datos cuantitativos listos para ser utilizados en las operaciones de alto nivel, por lo que se entiende que los datos que se leen de los sensores tienen que pasar por un proceso de "conversión". El PIC16F627A es el microcontrolador que tiene a su cargo la medición de la distancia recorrida proporcionando los centímetros recorridos después de una señal de inicio.

La locomoción es mediante ruedas ya que permite una fácil construcción, capacidad de carga y mejor estabilidad. La configuración es de tres ruedas, tipo triciclo, con tracción trasera. La llanta delantera es giratoria de 360 grados.

Se utilizan dos llantas en la parte posterior y cada llanta funciona con un motor DC con motorreductor. Los dos motores son activados a través del mismo puente H que permite manipular por separado o en conjunto.

Dos llantas independientes con motor en la parte posterior y una llanta unidireccional sin motor al frente.

Figura 13: Configuración de las llantas para el carrito "todo terreno"

Como tarjeta de desarrollo para el PIC18F4550 se utiliza una tarjeta de desarrollo de diseño del Mtro. Apolonio Ata Pérez para el microcontrolador antes mencionado. En la figura 1 se muestra dicho desarrollo.

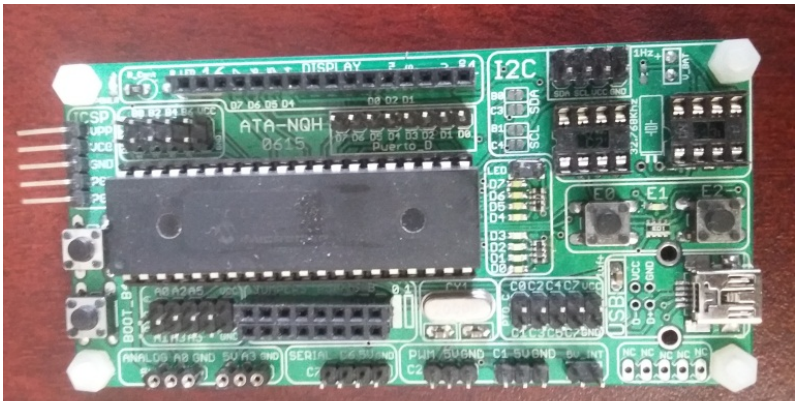

Figure 14: Tarjeta diseñada para el microcontrolador PIC18F4550.

Para suministrar una alimentación por medio de baterías, se utilizaron tres pilas de litio con una carga de 10 amp en total y con la capacidad de suministrar 2 amp.

4.3 Estimación de la distancia recorrida y detección de obstáculos.

La distancia recorrida es un proceso de alta prioridad, ya que debe indicar si parar o dejar que el carrito “todo terreno” continúe con la trayectoria. Para calcular la distancia recorrida se realiza a través de un encoder y un optointerruptor. Dado que se hace uso de un encoder con resolución de 20 pulsos por vuelta y se utilizan las interrupciones del microcontrolador, se requiere que sea la tarea más importante y prioritaria, por lo tanto, se decide que el microcontrolador PIC16F628A sea el encargado de obtener, calcular y entregar la información en todo momento.

En la trayectoria del carrito “todo terreno” existe, en todo momento, ruido mecánico, es decir, vibraciones entre el optointerruptor y el encoder que puede generar pulsos falsos/positivos que modifican la precisión de la distancia recorrida. A consecuencia del ruido se utiliza un filtro pasa-bajos en cada interrupción con la finalidad de descargar la mayor cantidad de pulsos generados por el ruido.

Como se ha mencionado, la única tarea del PIC16F628A es procesar los pulsos para obtener información útil, sin embargo, el proceso de iniciar o parar el conteo es proporcionado por el PIC18F4550.

Para reconocer un obstáculo en la trayectoria se utilizan tres sensores infrarrojo Sharp GP2Y0A21YK0F que permite reconocer obstáculos a una distancia de 80 cm, sin embargo, la distancia mínima es de 10 cm y dado que no es un sensor digital se tiene que estimar a través de una conversión.

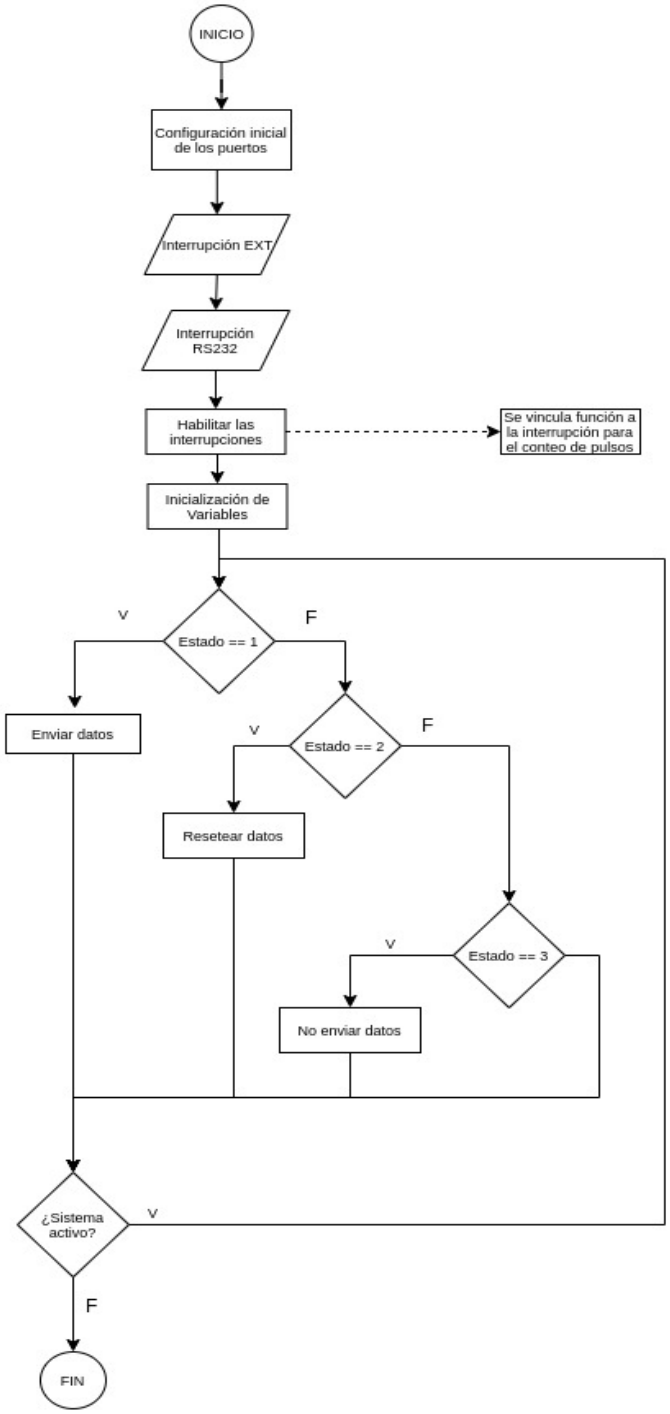

Figura 15: Diagrama de flujo para la estimación de la distancia recorrida.

El proceso de reconocer un objeto en cualquier estado, por ejemplo: en la trayectoria, es considerado importante pero no de la más alta prioridad, ya que gracias a los 80 cm de alcance y una comprobación de distancia cada ciclo de ejecución del programa puede anticipar una colisión y para el **carrito "**todo terreno". Un punto importante es que dicho sensor sólo reconocerá el objeto si se encuentra en su línea de vista.

4.4 Control del giro y desplazamiento.

Para controlar la velocidad se utiliza un puente H que está diseñado para aceptar señales de niveles lógicos estándar TTL y manejar cargas inductivas como por ejemplo: motor DC. Soporta el manejo de dos motores individualmente. Opera hasta un voltaje de 40 V con un máximo de 4 A. Se utilizan dos señales TTL para la dirección (girar hacia adelante o girar hacia atrás) y una señal que permite habilitar o deshabilitar; en total se requieren tres señales para cada motor DC.

Los motores son **DC TGP01D-A130** modelo 14175-120 que opera en un rango de 3v a 6v y cuenta con un torque de 1.5 kg cm. Cuenta con una relación de reducción de 1/120.

Se utiliza los mismos motores DC, llantas y son controlados por el mismo puente H usando el microcontrolador (PIC18F4550), sin embargo, por factores de fabricación, mecánicas, de suelo y algunos otros factores la velocidad que pueda alcanzar cada motor en un determinado tiempo, varia. Teniendo un sistema inestable, se requiere el uso de la técnica de PWM para controlar la velocidad de cada motor con el objetivo de avanzar con una velocidad constante. Es necesario un punto de referencia para que cada motor pueda decidir si disminuir o

aumentar la velocidad. Dicho punto de referencia es proporcionado por el sensor inercial MPU6050.

El MPU 6050 que forma parte del integrado GY 521. Dicho integrado cuenta con un procesador de movimiento interno que proporciona datos procesados y listos para su uso, sin embargo, para su uso se requiere pagar. Una forma de implementar sin utilizar el procesador de movimientos (DMP) que tiene integrado, es obtener los datos en crudo y a través de operaciones geométricas y un filtro complementario se pueda obtener un dato aceptable, utilizando el protocolo I2C para obtener los datos que proporciona el integrado.

Nota: con este proceso los grados de inclinación en Z, no son óptimos, pero son estables.

Dado que el integrado es un dispositivo MEMS, es altamente recomendable realizar una calibración antes de su uso. Realizando múltiples pruebas, se observa que es necesario darle un tiempo de 3 a 5 seg. es más que suficiente para realizar las lecturas.

El objetivo de la calibración es obtener valores en crudo (quizá unos 100 valores) para tener un promedio haciéndolo para cada uno de los ejes (*x,y* y *z* tanto para el giróscopo como para el acelerómetro), esto se realiza porque el sensor no proporciona los mismos valores para una sola posición. Entonces se realiza un promedio con los datos obtenidos durante el tiempo de calibración, para obtener el OFFSET y posteriormente quitar el offset de un valor obtenido, para tener un valor real. Para llevar a cabo el proceso de calibración, se requiere que el sensor

tiene que estar sin movimiento y posicionado de manera que los ejes estén en el origen (excepto en el eje z del acelerómetro que deberá de marcar +1G) para obtener un promedio respecto a dicho origen. Nunca indicará el CERO, siempre existirá un offset.

Antes de iniciar una trayectoria, se establece un proceso de inicio, con lo que se establece un punto de referencia a través de la información que proporciona el sensor MPU6050 y aplicando un filtro se logra obtener los grados de la rotación del eje Z (guiñada) por lo tanto se puede determinar qué motor debe modificar su velocidad.

Dado que las modificaciones entre los dos motores pueden variar todo el tiempo, entonces, se fija una velocidad constante en un motor, por lo tanto, el otro motor es el que se modifica en el tiempo según los grados proporcionados. Por ejemplo, la velocidad aumenta cuando el valor es negativo y aumenta cuando es mayor a cero, dependerá del motor que modifique su velocidad.

El PIC18F4550 realiza el cálculo del eje Z (yaw) del carrito, obteniendo los datos del acelerómetro, así como la recepción de las instrucciones de la Raspberry PI v3 y el control del PWM de los motores. El PIC16F628A realiza el cálculo de la distancia que se desplaza el móvil procesando los datos del optointerruptor.

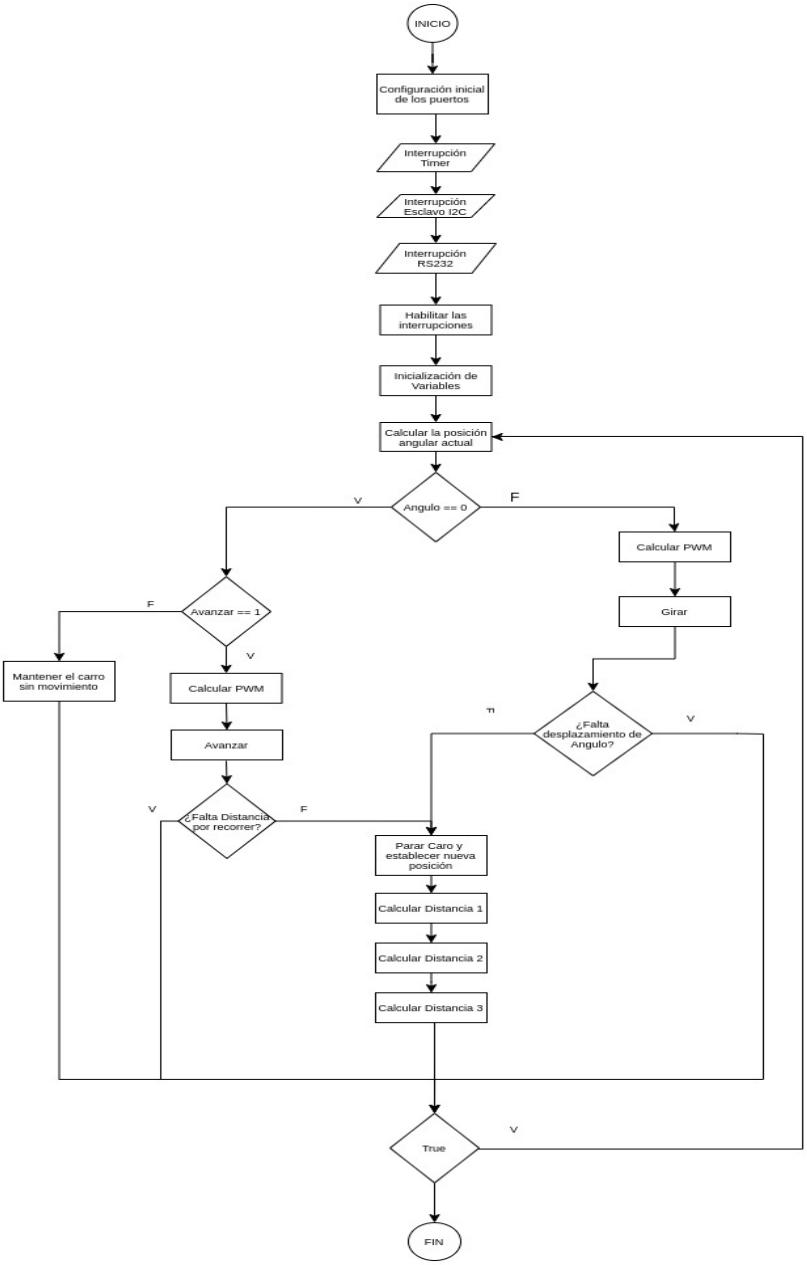

Figura 16: Diagrama de flujo para el control de la dirección, giro y detección de obstáculos.

4.5 Esquemático para los microcontroladores.

El esquemático se realizó con ayuda del Software Proteus 8.0. El objetivo de tener una tarjeta de desarrollo con los dos microcontroladores unificados en una sola placa para facilitar el desarrollo y disminuir errores por fallas de suministro de energía o falsos en las entradas y salida de datos. Dicho esquema está basado en el desarrollo de la tarjeta PIC18F4550 del M.C. Apolonio Ata Pérez utilizado en el proyecto.

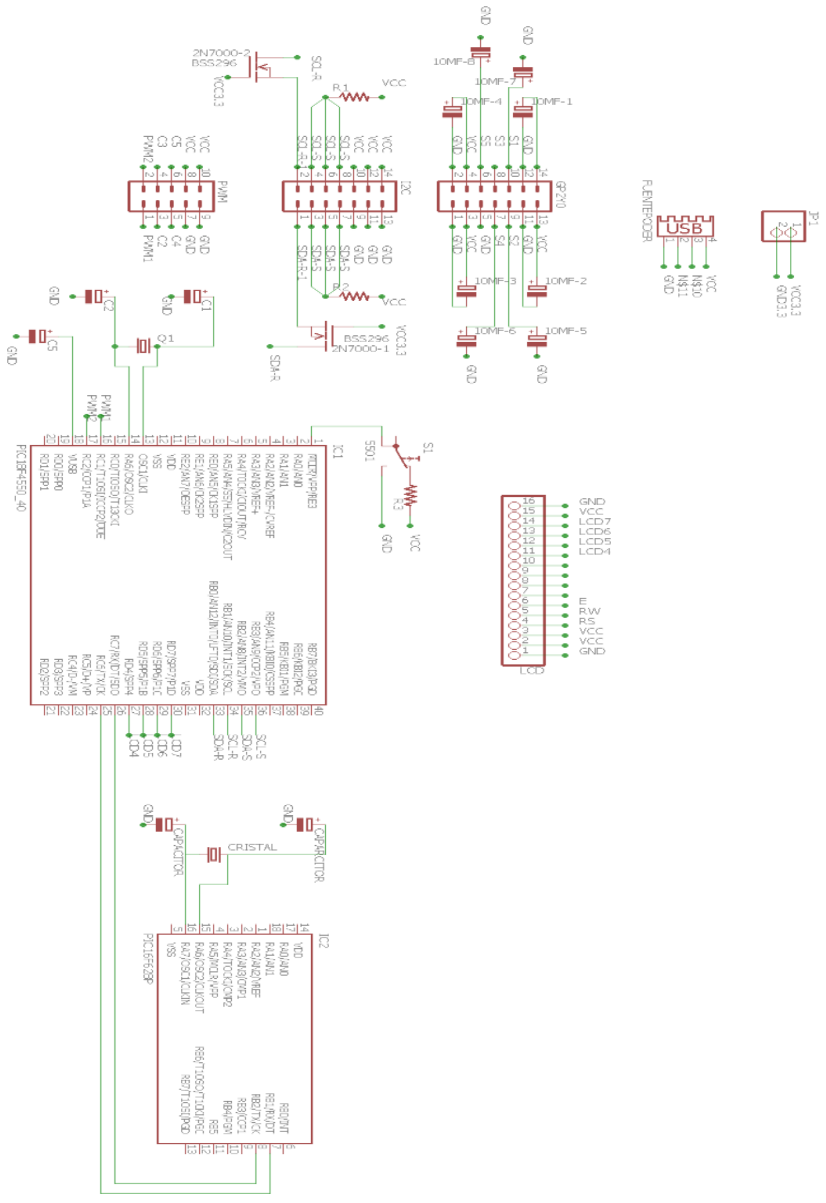

Figura 17: Esquema para el PIC18F4550 y PIC16F627A

Capítulo V Puesta a punto del sistema.

5.1 Entorno de desarrollo.

Para configurar la Raspberry Pi v3 se utiliza el sistema operativo *Raspbian* que esta basado en Debian. Los pasos necesarios para la configuración se mencionan a continuación.

1.- Descarga de la imagen de Raspbian en la página oficial de Raspberry Pi.

2.- Se utiliza una microSD, a la cual se le ha montado Raspbian previamente.

3.- Se habilita la comunicación I2C, usando el comando *sudo raspi-config*

4.- Se puede modificar la frecuencia del protocolo en el archivo */etc/modprobe.d/i2c.conf.* Por ejemplo, options i2c-bcm2708 baudrate=400000.

5.- Comprobar que el módulo i2c-bcm278 se encuentre en */etc/modprobe.d/raspi- blacklist.conf*

6.- Instalar los módulos i2c-tools y python-smbus

Para programar los microcontroladores se utilizó el software PIC Compiler 5.008 instalado en un Windows 7 de 32 bits.

5.2 Descripción del API del sistema.

Estas funciones están escritas en el lenguaje de programación python en la versión 2.7.x y se ayuda en el lenguaje de programación C para la conversión de datos y operaciones a nivel de bits.

Funciones prototipo de python 2.7.x

def set_distancia(metros, centimetros):
Establece en metros y centímetos la distancia que va a recorrer
Rangos permitios: metro = [0-2] y centímetros = [0,15,30,45,60,75,90]

def instruccion():
Devuelve 1 si el carrito se encuentra ocupado, 0 en caso contrario.

def get_infrarrojo_en_frente():
Devuelve la distancia en cm. sí ha un objeto enfrente.

def get_infrarrojo_derecha():
Devuelve la distancia en cm. sí ha un objeto a la derecha.

def get_infrarrojo_izquierda():
Devuelve la distancia en cm. sí ha un objeto a la izquierda.

def get_distancia():
Devuelve la distancia en cm. que ha recorrido.

def get_anguloz():
Proporciona el ángulo Z.

def reiniciar():
Reinicio el sistema del carrito.

def detener():
Detiene el proceso de ejecución.

def girar(param):
Indica los grados y sentido del giro a realizar.
Los valores permitidos son: [0 +0, 1 +45, 2 +90,
3 +135, 4 +180, 5 +225, 6 +270, 7 +315, 8 +360,
9 -45, 10 -90, 11 -135, 12 -180, 13 -255, 14 -270,
15 -315, 16 -360]

Función prototipo en C.

float Bytes_Convert_Float(Byte msb, Byte a, Byte b, Byte lsb):

Realiza la conversion de un número de 32 bits entero a un flotante.

5.3 Funciones para la comunicación entre el PIC18F4550 y MPU6050

Se implementan funciones de escritura y lectura por bytes utilizando el protocolo I2C, de tal manera, que sea de forma transparente la comunicación. Ya que el compilador PIC C proporciona las funciones mínimas para establecer dicha comunicación, se implementaron las siguientes funciones:

- *byte read_i2c (byte direccion,byte registro);*
- *void read_i2c_bytes (byte direccion, byte registro,int8 length, int8* data);*
- *int8 read_i2c_bit (byte direccion, byte registro, byte bitNum);*
- *byte read_i2c_bits (byte direccion,byte registro,byte BitInicio, byte Longitud);*
- *void write_i2c (byte direccion,byte registro,byte dato);*
- *void write_i2c_bit (byte direccion, byte registro, byte bitNum, byte data);*
- *void write_i2c_bytes (byte direccion,byte registro,int16 inicio,int8 tamano, int8 tipo);*
- *void write_i2c_bits (byte direccion, byte registro, int8 bitStart, int8 length, int8 data);*
- *void write_i2c_word (byte direccion, byte registro , int16 offset);*

Dichas funciones prototipo se utilizaron para acceder al sensor digital y obtener los valores que proporciona para su posterior procesamiento.

Cuando se obtienen los datos que proporciona el sensor, se utiliza una ecuación geométrica para obtener los grados de inclinación respecto a un punto inicial. Utilizando un filtro complementario para calcular y obtener datos correctos, o que tengan un margen de error "mínimo".

Las funciones prototipo para acceder al sensor son los siguientes:

- *void MPU6050_Inicializar();*
- *void MPU6050_Calibrar(float* aux_gy_x, float* aux_gy_y, float* aux_gy_z);*
- *void MPU6050_GetAccelX(signed int16 * acc_x);*
- *void MPU6050_GetAccelY(signed int16 * acc_y);*
- *void MPU6050_GetAccelZ(signed int16 * acc_z);*
- *void MPU6050_GetGyroX(signed int16 *gy_x);*
- *void MPU6050_GetGyroY(signed int16 *gy_y);*
- *void MPU6050_GetGyroZ(signed int16 *gy_z);*
- *void MPU6050_GetGyroXYZ(signed int16 *gy_x,signed int16 *gy_y,signed int16 *gy_z);*

Nota: Se basa en la implementación que se encuentra en el repositorio i2cdevlib/Arduino/ del usuario jrowberg.

5.4 Distribución e Instalación.

El programa principal, es el de la Raspberry Pi v3, dado que los microcontroladores son programados previamente. Como es un módulo .py, será necesario el archivo setup.py para poder generar la distribución y el código fuente, así como el código fuente de la función escrita en C.

DevCar /
|- carrito.py
|- setup.py
|- bytestofloat.c
|- bytestofloat.h

Comando para crear una biblioteca dinámica:

- *gcc -fPIC -c bytestofloat.c -o bytestofloat.o*
- *gcc -shared -Wl,-soname,libbytestofloat.so -o libbytestofloat.so.1.0.0 bytestofloat.o -lc*
- *ln -s libbytestofloat.so.1.0.0 libbytestofloat.so*

Comandos para generar la distribución del programa .py.

- *sudo python setup.py build # contruye*
- *sudo python setup.py sdist # distribuye, en este caso genera la biblioteca dinámica *.so*

Para realizar la instalación se requiere de los siguientes pasos:

- descomprimir "carrito-1.9.tar.gz" y ejecutar.
- *sudo python setup.py install* # copia el codigo carrito.py, carrito.pyc y bytestofloat.so a la carpeta /urs/local/lib/python2.7/dist-packages/

Para verificar la instalación revisar que existan los archivos en los directorios:

- /usr/local/lib/python2.7/dist-packages/carrito.py
- /usr/local/lib/python2.7/dist-packages/carrito.pyc
- /usr/local/lib/python2.7/dist-packages/bytestofloat.so
- /usr/local/lib/python2.7/dist-packages/carrito.1.0.egg.info

Capítulo VI Pruebas y resultados

6.1 Pruebas previas a las rutinas de test

Las pruebas previas realizadas al carrito “todo terreno” se clasifican de la siguiente manera:

1. Componentes físicos
2. Programación.

Los sensores infrarrojos sharp GP2D12 fueron acondicionados con capacitores para eliminar ruido en la señal y mantener el voltaje estable. Los capacitores que se utilizaron fueron de 10 microfaradios conectados como se muestra en la figura 17.

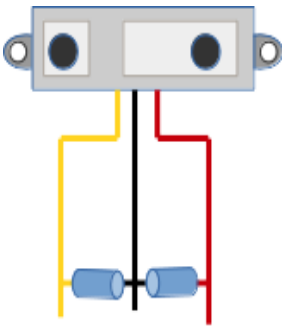

Figure 18: Circuito del Sharp GP2D12 con capacitores

Los tres sensores Sharp fueron acondicionados con estos capacitores teniendo beneficios en las pruebas. En la figura 19 se muestra a los capacitores de un solo sensor que han sido soldados.

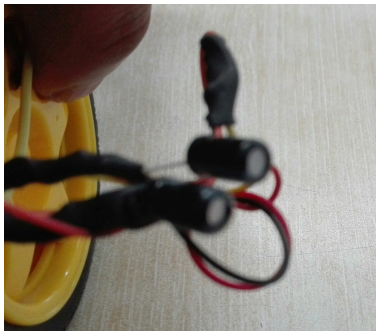

Figure 19: Capacitores soldados a un sensor

Se creó una rutina para almacenar datos en crudo en una memoria microSD de 2 GB en formato fat32 en el microcontrolador para su posterior procesamiento. El objetivo, procesar dicha información para proponer una mejor configuración de sus valores iniciales. La configuración del acelerómetro y girómetro está basado en las diversas pruebas realizadas. En la figura 20 se puede observar que los datos en “crudos” que proporciona el acelerómetro de manera inicial es inestable, por lo que fue necesario aplicar un filtro pasa bajos para mejorar los resultados, en la figura 21 se puede observar una mejora.

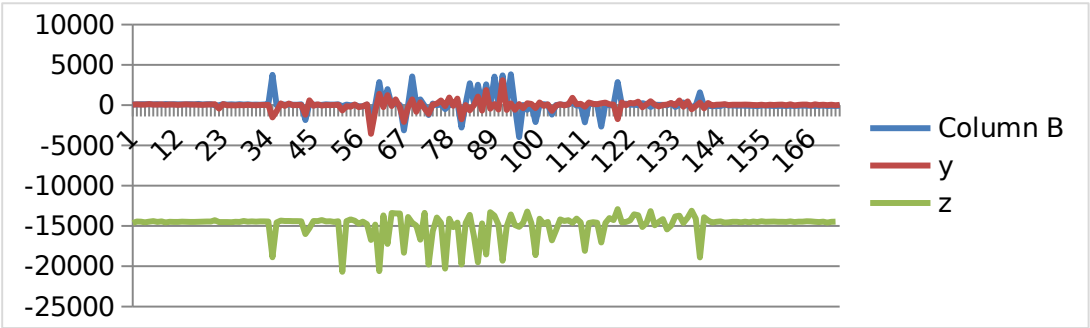

Figura 20: Gráfica que representa datos en "crudo" de los tres ejes del acelerómetro. La línea verde representa el eje Z, la línea azul el Y y la línea roja el eje X.

En la gráfica 21 se observa que en el eje Y se tiene una inclinación y esto se debe a que el carrito en donde se encuentra ubicado el sensor, se encuentra con una inclinación por la altura de la rueda “loca”.

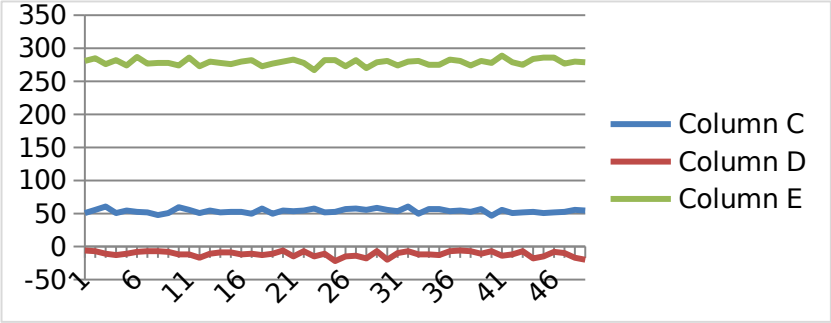

Figura 21: Gráfica que representa datos con un filtro pasa bajos de los tres ejes del acelerómetro. La línea verde representa el eje Z, la línea azul el Y y la línea roja el eje X.

Se realizaron pruebas sin éxito, al intentar, agregar una distancia calculada a partir de los datos proporcionados por el acelerómetro (ver figura 22), sin embargo, para realizar dicho cálculo se requiere de otro microcontrolador o bien cambiar a otro de mayor potencia de

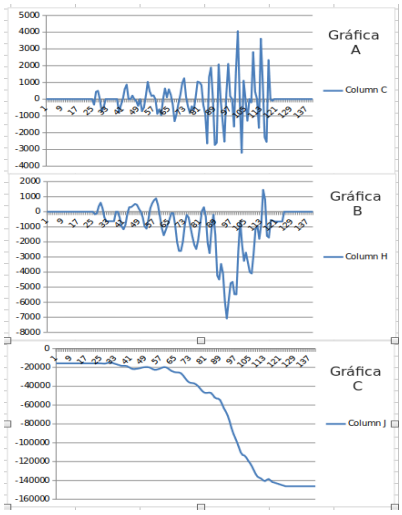

Figura 22: La gráfica A representa los datos del acelerómetro del eje X. La gráfica B representa la velocidad y la gráfica C la distancia.

cálculo, así como de adquirir otro sensor MPU6050. Dado que la arquitectura se hace más compleja, se descarta.

Para las pruebas de la configuración del PMW fueron a través de diferentes parámetros en la configuración, ya que los motores requieren de un pulso mínimo para el funcionamiento correcto y con una frecuencia adecuada, de tal manera que no pierda el torque para un desplazamiento efectivo.

PR2	Período del PWM en microsegundos	Hz	Khz
145	1557.333333333	642	0.642
146	1568	637	0.637
147	1578.66666666	633	0.633
148	1589.33333333	629	0.629
149	1600	625	0.625
150	1610.66666666	620	0.62
151	1621.33333333	616	0.616
152	1632	612	0.612
153	1642.66666666	608	0.608
154	1653.33333333	604	0.604
155	1664	600	0.6
156	1674.66666666	597	0.597
157	1685.33333333	593	0.593
158	1696	589	0.589
159	1706.66666666	585	0.585
160	1717.33333333	582	0.582
161	1728	578	0.578
162	1738.66666666	575	0.575
163	1749.33333333	571	0.571
164	1760	568	0.568
165	1770.66666666	564	0.564

Figura 23: Parámetros con lo que se realizaron las pruebas del PMW

La prueba de software es mediante la rutina de *reiniciar()* con la que se realiza una verificación de todo el sistema del carrito. En caso de perder alguna comunicación entre los dispositivos se notifica mediante un display de 7 segmentos que sirve como terminal para mostrar los mensajes de depuración, sin embargo, no se monitorea o verifica la presencia de los sensores.

En la figura 24 se muestra el prototipo del carrito “todo terreno” con la reutilización de material y en una versión de test.

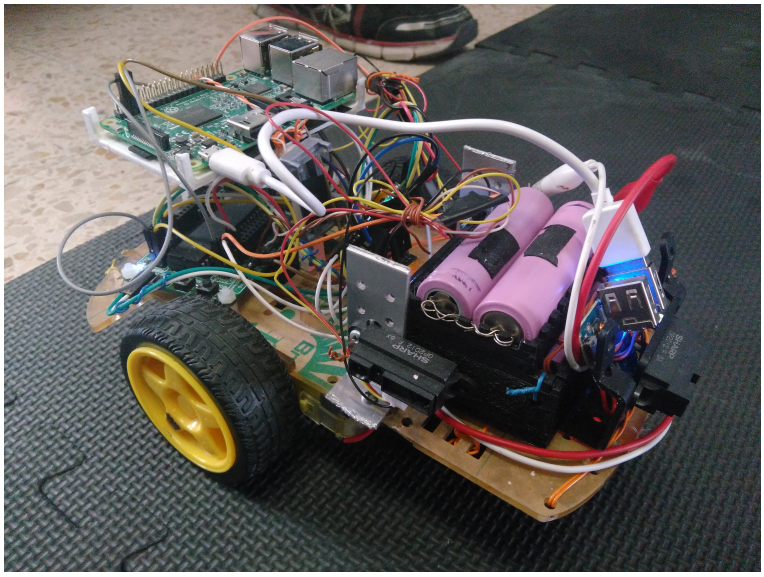

Figura 24: Carrito "todo terreno" en test de componentes y prueba de la comunicación

6.2 Rutinas de test

Se desarrollaron tres rutinas con lenguaje de Python utilizando el API que se desarrolló. La primera prueba consiste en realizar un desplazamiento en línea recta de 1 metro con pausas de un segundo y continuar la trayectoria. La segunda prueba consiste en dibujar un cuadrado. La tercera prueba consiste en generar una rutina con desplazamiento recto con giros a la izquierda o derecha en un ángulo de 45 o 90 grados y el objetivo es que el carrito pueda detectar obstáculos y poder tomar una acción para no colisionar.

Para la primera prueba que se realiza, se encontró un error acumulado, después de 2 metros se nota una desviación constante que incrementa de acuerdo a la distancia recorrida. De las 5 veces que se realizó esta misma prueba se calculó un promedio de 2.5 centímetro por metro desplazado. Aplicando un reset cada metro, se reduce el error, sin embargo, el tiempo para llegar a los 5 metros se duplica.

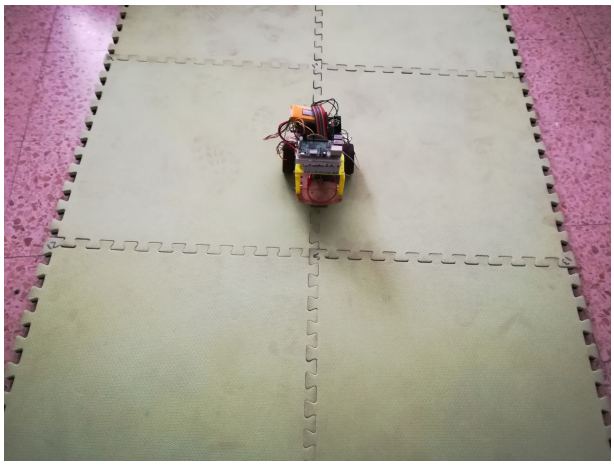

Figure 25: Escenario para el desplazamiento en línea recta

Para la segunda prueba se aplicó 3 veces, se notó una tendencia a dibujar un círculo. En los giros que realiza en cada esquina se puede apreciar que la fuerza con la que realiza el giro hace que no se detenga exactamente a los 90 grados, con lo que el error se va acumulando en cada giro, con cada vuelta se aproxima a su máximo error y después se restablece. Se realizó una modificación en el control para restar los grados que se giran de más, sin embargo, no se obtuvieron resultados favorables, se repite el mismo patrón.

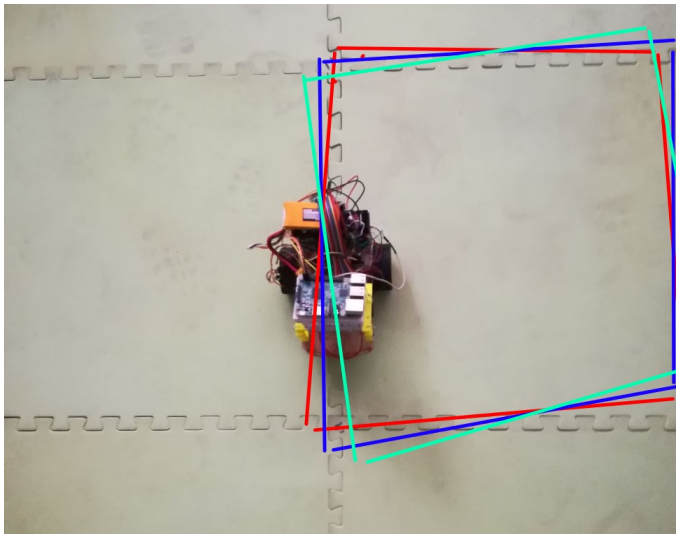

Figure 26: Prueba del carrito para dibujar un cuadrado.

La última prueba realizada, es una rutina considerando obstáculos que tiene que detectar, ya que el carrito cuenta con tres sensores para la detección, entonces se realiza una rutina indefinida en el tiempo (while True) y con instrucciones de distancia y giros aleatorios. Si se detecta un obstáculo en frente entonces, girar a la derecha o izquierda. Si se ha detectado un obstáculo en la izquierda, entonces realizar un giro a la derecha, es decir, al lado opuesto.

Se puede observar que los sensores no soportan la distancia suficiente para detectar un objeto, ni mucho menos, cuando el sensor se encuentra en movimiento. De los diferentes obstáculos utilizados, los de forma irregular no los detectó, por lo que siempre colisionó. Con objeto regulares, por ejemplo, cajas, paredes, etc. si realizó correctamente la detección, aunque por el rango, en algunas ocasiones quedaba justo al lado del objeto.

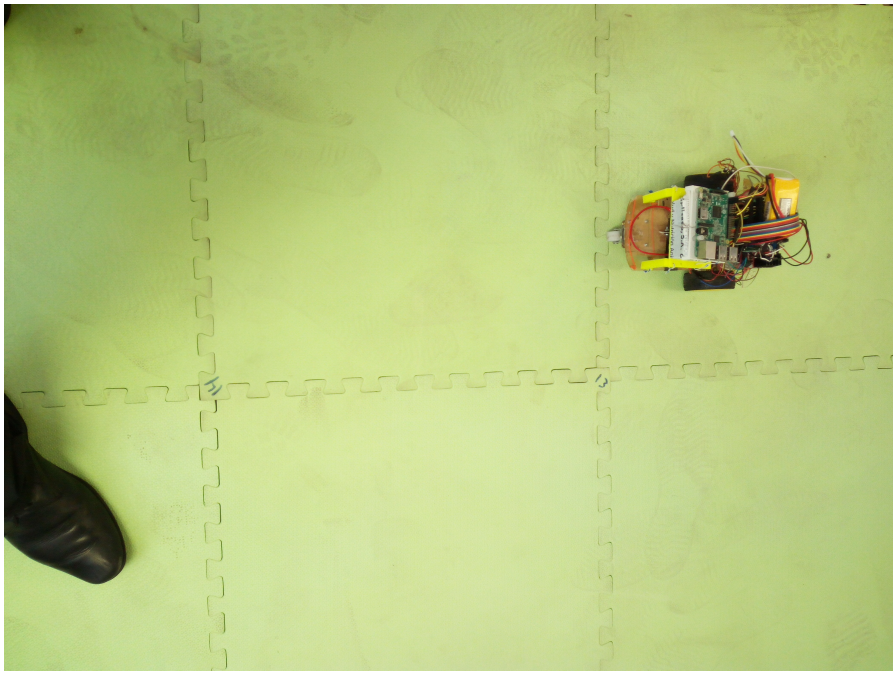

Figure 27: Obstáculo irregular para probar el sensor frontal

6.3 Conclusiones y trabajo futuro

La capacidad de cómputo en un dispositivo de pequeñas dimensiones y con bajo suministro de energía, permite revolucionar las propuestas de solución a problemas actuales. El desarrollo de este prototipo ha sido posible gracias a la capacidad técnica que me ha brindado la universidad y que tiene la finalidad aportar a la comunidad universitaria, en particular a los desarrolladores de algoritmos bioinpirados, a tener una base para poder contribuir, utilizar, mejorar, aprender y proponer diversos prototipos adecuados para diferentes problemas.

Los retos del proyecto se encontraron en los diferentes filtros para conseguir una estabilidad y corrección por los errores producidos por el "terreno", ruidos mecánicos, etc. por lo que, se puede mejorar y proporcionar una gama de filtros que se puedan usar según los intereses del problema.

El intercambio de información entre los diferentes subsistemas ha abierto un nuevo campo para mejorar. La programación asíncrona, así como mecanismos de sincronización entre los diferentes subsistemas son una mejora a futuro. El reto de las partes mecánicas no ha sido objeto de estudio, sin embargo, quedan cómo una gran mejora que pueda aportar una mayor estabilidad y evitar interferencias entre los sensores.

Inspirado en los ideales del *open source* la meta ha sido el estudio del código fuente y la reutilización del mismo, he decidido dejar el código fuente en https://github.com/ferthus.

Bibliografía

[1] Z.W. Geem, J. H. Kim, and G. V. Loganathan, “A new heuristic optimization algorithm: harmony search,” Simulation, vol. 76, no. 2, pp. 60–68, 2001.

[2] Yang, Xin-She. "Harmony search as a metaheuristic algorithm." Music-inspired harmony search algorithm. Springer, Berlin, Heidelberg, 2009. 1-14.

[3] Mirjalili, Seyedali, and Andrew Lewis. "The whale optimization algorithm." Advances in engineering software 95 (2016): 51-67.

[4] Storn, Rainer, and Kenneth Price. Differential evolution. A simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11.4 (1997): 341-359.

[5] Cheng, Shih-Lian, and Chyi Hwang. Optimal approximation of linear systems by a differential evolution algorithm. IEEE Transactions on Systems, man, and cybernetics-part a: systems and humans 31.6 (2001): 698-707.

[6] Sreedhar, M., et al. "A Review on Advanced Optimization Algorithms in Multidisciplinary Applications." Recent Trends in Mechanical Engineering. Springer, Singapore, 2020. 745-755.

[7] Abdel-Raouf, Osama, and Mohamed Abdel-Baset Metwally. "A survey of harmony search algorithm." International Journal of Computer Applications 70.28 (2013).

[8] Assad, Assif, and Kusum Deep. "Harmony search based memetic algorithms for solving sudoku." International Journal of System Assurance Engineering and Management 9.4 (2018): 741-754.

[9] Ing. Rubén D. Sánchez Dams, Estado del Arte del Desarrollo de sistemas embebidos desde una perspectiva integrada entre el hardware y software, 2013 vol. 2 - número 22: 98-105.

[10] José Isidro Hernández Vega, El Software embebido y los retos que implica su desarrollo. Conciencia Tecnológica N.º 40, julio-diciembre 2010.

[11] José Ignacio Suárez Marcelo, 2001, tesis, Robot móvil para transporte automatizado, Tesis, Universidad de Extremadura.

[12] Carlos Borrás, Jaime Barrero, 2009, tesis. Diseño y construcción de un prototipo de robot móvil autónomo de exploración. Tesis, Universidad Industrial de Santander.

[13] Rafael Agusto Sobrevilla Figueroa, 2009, Diseño y construcción de un robot móvil para resolver laberintos, tesis. Universidad Nacional Autónoma de México.

[14] Jorge Luis Azuara Domínguez, 2018, Navegación de un robot móvil para la supervisión de una residencia monitoreada via internet. Tesis, Universidad Nacional Autónoma de México.

[15] Aguilera Hernández Martha I., Bautista Miguel A., Iruegas Joaquín, Diseño y control de robots móviles, 2007.

[16] Farouk Azizi, Nasser Houshangi, Mobile Robot Position Determination, Purdue University Calumet,USA. 69-81.

[17] Luis Miguel Betancourt, Nelsón Botón Gómez, 2009, Implementación de un sistema teleoperado con realimentación visual para evasión de obstáculos de un robot móvil, Tesis, Universidad Militar Nueva Granada.

[18] Implementing Positioning Algorithms Using Accelerometers, Kurt Seifert and Oscar Camacho, Aplication Note, AN3397, 2007.

[19] José Armando Segovia de los Ríos, Jorge Samuel Benítez Read, Robots Moviles y sistemas remotos en aplicaciones nucleares. (2009): 389-406

[20] Roland Siegwart, Illah R. Nourbakhsh. Introduction to Autonomous Mobile Robots, Massachusetts Institute of Technology, 2004

[21] NXP Semiconductors, I2C-bus specification and user manual, Rev 6. 2014, UM10204

[22] RajanArora , Texas instrument I2C BUS PULLUP RESISTOR CALCULATION, Aplication Report,, 2015.

[23] Erique Palacios, Fernando Remiro, Lucas J. López. Microcontrolador PIC16F84 Desarrollo de proyecto, AlfaOmega, 2009, 3re edición.

[24] Eduardo Garcia Brejio, Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC. Alfaomega, 2008, primera edición.

[25] Stephen Bowling, Richard L. Fischer, Microchip Technology Incorporated, An I2C Network Protocol for Environmental Monitoring, 2000, AN736.

[26] Mike Garbutt, Microchip Technology Inc, Asynchronous Communications with the PICmicro USART (AN774). 2003.

[27]. J. Acedo Sánchez, Instrumentacion y control básico de procesos, España: Díaz de Santos. 2006

[28]. J. Acedo Sánchez, Control avanzado de procesos, España: Díaz de Santos, 2003

[29]. Karl. J Aström, Tore Hägglund, Control PID avanzado, España: Prentice Hall, 2009

[30]. Microchip Technology Inc.: PIC18F2455/2550/4455/4550 Data Sheet. *Microchip*. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf. Accedido el 10 de febrero de 2016.

[31] Microchip Technology Inc.: PIC16F627A/628A/648A Data Sheet. *Microchip*. http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf. Accedido el 10 de febrero de 2016

[32] BCM2837 ARM Peripherals Data Sheet, Broadcom, https://cs140e.sergio.bz/docs/BCM2837-ARM-Peripherals.pdf. Accedido el 5 de agosto de 2017.

[33] STMicroslectronics, L298 DUAL FULL-BRIDGE DRIVER, Datasheet, 2000.

[34] Sharp Corporation.: GP2Y0A21YK0F Distance Measuring Sensor Unit Measuring distance: 10 to 80 cm Analog output type. *Parallax*. https://www.parallax.com/sites/default/files/downloads/28995-Sharp-GP2Y0A21YK0F-IR-Datasheet.pdf. Accedido el 20 de agosto de 2016.

[35]. Berrutti J.; Falkenstein L.; Favaro F.: Proyecto de fin de carrera de Ingeniería y Eléctrica Vuelo autónomo de un cuadricóptero. Tesis, 2015, Facultad de Ingeniería de la Universidad de la República.

[36]. InvenSense Inc.: MPU-6000 and MPU-6050 Register Map and Rescriptions Revision 4.0. *Invensense*. https://store.invensense.com/Datasheets/invensense/RM-MPU-6000A.pdf Accedido el 20 de octubre de 2015.

[37] InvenSense Inc.: MPU-6000 and MPU-6050 Product Specification. *Invensense*. https://store.invensense.com/datasheets/invensense/MPU_6500_Rev1.0.pdf Accedido el 20 de octubre de 2015.

[38] Kimberly Tuck, Accelerometer Systems and Applications Engineering Tempe, AZ, Tilt Sensing Using Linear Accelerometers, 2007, Rev 2, AN3461.

[39] Bruno Barone, Jhonny Rodriguez., Desarrollo de un sistema de navegación inercial reprogramable para múltiples plataformas móviles, proyecto de grado, 2014, pag. 65.

[40] Everlight.: Technical Data Sheet Opto Interrupter. *Everlight*. ITR8307/F43, http://pdf1.alldatasheet.es/datasheet-pdf/view/229695/EVERLIGHT/ITR9608-F.html, Accedido el 21 de Agosto de 2016.