UNIDAD DE GESTIÓN DE TECNOLOGÍAS

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

TEMA: "IMPLEMENTACIÓN DE UN VEHÍCULO DE DOS RUEDAS AUTO-EQUILIBRADO PARA PRÁCTICAS DE MICROCONTROLADORES."

AUTOR: SUNTAXI REIMUNDO JIMMY JEFFERSON

DIRECTOR: ING. CRISTIAN CHUCHICO

LATACUNGA

2016

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

CERTIFICACIÓN

Certifico que el trabajo de titulación. " **IMPLEMENTACIÓN DE UN VEHÍCULO DE DOS RUEDAS AUTO-EQUILIBRADO PARA PRÁCTICAS DE MICROCONTROLADORES** " realizado por el señor **SUNTAXI REIMUNDO JIMMY JEFFERSON**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **SUNTAXI REIMUNDO JIMMY JEFFERSON** para que lo sustente públicamente.

Latacunga, 12 Octubre del 2016

_____

ING. CRISTIAN CHUCHICO

DIRECTOR

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

AUTORÍA DE RESPONSABILIDAD

Yo, **SUNTAXI REIMUNDO JIMMY JEFFERSON**, con cédula de identidad N° 1721732558 declaro que este trabajo de titulación " **IMPLEMENTACIÓN DE UN VEHÍCULO DE DOS RUEDAS AUTO-EQUILIBRADO PARA PRÁCTICAS DE MICROCONTROLADORES** " ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 12 Octubre del 2016

_____

JIMMY JEFFERSON SUNTAXI REIMUNDO

C.C. 172173255-8

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

AUTORIZACIÓN

Yo, **SUNTAXI REIMUNDO JIMMY JEFFERSON**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación " **IMPLEMENTACIÓN DE UN VEHÍCULO DE DOS RUEDAS AUTO-EQUILIBRADO PARA PRÁCTICAS DE MICROCONTROLADORES** " cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 12 Octubre del 2016

_____

JIMMY JEFFERSON SUNTAXI REIMUNDO

C.C. 172173255-8

DEDICATORIA

Mi gratitud inmensa a Dios por las bendiciones recibidas a mi persona, por darme las fuerzas para seguir adelante a pesar de los obstáculos que se presentaron en el camino de la vida y salir triunfante en cada uno de ellos.

Este trabajo de graduación producto de mi perseverancia y dedicación en cada una de los períodos, les dedico a mis padres quienes a lo largo de toda mi vida han apoyado y motivado mi formación académica y como persona, mismos que creyeron y confiaron en mí.

A toda mi familia, por haberme apoyado moral quien con sus consejos me han animado para seguir adelante y como respuesta a todo ese apoyo es la obtención de un nuevo triunfo en mi vida.

Jimmy Suntaxi.

AGRADECIMIENTO

A Dios por brindarme el conocimiento y perseverancia para cumplir mis metas trazadas.

A mis padres que con su amor protección y apoyo supieron guiarme por el camino del bien y en el trascurso de mi vida pueda caminar con pie firme y con sueños claros.

Mi gratitud a la Unidad de Gestión de Tecnologías por brindarme la oportunidad de profesionalizarme, a todos los ingenieros que con firmeza y disciplina supieron formarme con valores y honestidad.

Gracias a mis compañeros por haberme dado la oportunidad de compartir todos mis sueños, experiencias, alegrías y frustraciones, los llevare toda mi vida en el alma como mejor de mis recuerdos.

Jimmy Suntaxi

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El trabajo de grado trata de la implementación de un vehículo auto-equilibrado con la utilización de varios sensores, actuadores y un algoritmo de control, el mismo que será programado en Arduino Mega 2560. La aplicación consiste en un vehículo de dos ruedas que mantiene el equilibrio a través de los datos obtenidos por un sensor giróscopo (MPU 6050). Se realiza el envío de datos entre el MPU 6050 y Arduino Mega 2560 mediante la comunicación I2C. La implementación del vehículo no tuvo grandes problemas al momento de realizar las pruebas de funcionamiento ya que se utiliza uno de los tantos métodos existentes llamado, método de sintonización de ganancia limites, el cual es uno de los más sencillos de realizarlos, se explica claramente como variar los parámetros Proporcional, Integral, Derivativo (PID) y encontrar los adecuados. En el trabajo realizado se puede apreciar claramente como comprobar el correcto funcionamiento de cada uno de los dispositivos utilizados y poder encontrar la falla técnica en caso de que esta exista. Una de las pequeñas desventajas encontradas en el vehículo es la inestabilidad a grandes perturbaciones, las cuales provocan que pierda el equilibrio y caiga.

PALABRAS CLAVES:

- **I2C**
- **Arduino**
- **MPU 6050**
- **Lenguaje de programación**
- **Microcontrolador**

ABSTRACT

This written work is about the development of a vehicle self-balanced through the use of sensors, actuators and a control algorithm. This process involves of a two-wheeled vehicle which keeps the balance through the data obtained by a gyro sensor (MPU 6050). The data transfer between the MPU 6050 to Arduino Mega. It is done through the I2C communication code. The implementation of the vehicle had no problems at testing operation due the applying of tuning increase parameters method, which is one of the simplest of making, it is clearly explained how to vary the PID parameters and find the right ones. In the current work you can clearly see how check proper functioning of each device used. One of the disadvantages encountered in the vehicle is instability at large alarms causing it falls down.

KEY WORDS:

- I2C
- Arduino
- MPU 6050
- Programming Language
- Microcontroller

_____

Lic. Wilson E. Villavicencio F. Msc

DOCENTE UGT

CAPÍTULO I

INTRODUCCIÓN

1.1 ANTECEDENTES

La UGT fomenta tanto la práctica como la investigación experimental de nuevas tecnologías aplicadas a la industria. Por lo cual es importante conocer el funcionamiento y aplicación de una IMU incluyendo la programación de un microcontrolador basado en la plataforma Arduino.

Conforme la tecnología avanza se debe tener conocimiento de los diferentes tipos de microcontroladores existentes y de sus diferentes lenguajes de programación.

El vehículo autónomo de dos ruedas está compuesto de tres partes principales; sensores, unidad de procesamiento lógico y actuadores. En el presente proyecto, el robot puede mantener una posición vertical y en equilibrio. Consta de una Unidad de Medida Inercial (IMU) que mide la orientación, estas señales se transmiten al microcontrolador donde se ejecuta la secuencia el algoritmo de control, cuyo resultado es transmitido a los actuadores para que estos mantengan al vehículo en posición deseada.

1.2 PLANTEAMIENTO DEL PROBLEMA

La falta de IMUs que permita a los alumnos realizar prácticas de programación y utilización de elementos compatibles con Arduino limita el aprendizaje actualizado y genera un desfase con la tendencia actual de tecnología.

Este vehículo servirá como apoyo para la realización de futuras prácticas que motiven a los estudiantes a la investigación de nuevas tecnologías relacionado con IMUs.

Una consecuencia futura de la falta de conocimiento de la utilización de IMUs puede verse cuando el estudiante ejerza su profesión en una industria

basada en el funcionamiento de mencionados sistemas primarios de medida, se desplegaría dificultades al encontrarse con este tipo de tecnología para determinar problemas de funcionamiento en diseño de proyectos o revisión de los mismos caso contrario un estudiante que conozca y haya realizado prácticas de este tipo de tecnología se encontrara con mayor aptitudes para destacarse.

1.3 JUSTIFICACIÓN

En la actualidad dada la alta incursión de sistemas IMUs se ha convertido en una herramienta necesaria en todas las áreas de la industria especialmente dentro de la robótica y del campo aeronáutico.

La robótica usa una gran variedad de sensores giroscópicos y acelerómetros para determinar su ubicación en el espacio en sus tres ejes y mantenerse equilibrados automáticamente.

De forma similar en la industria automotriz los brazos robóticos utilizan IMUs ya que su trabajo es de suma presión en el ensamblaje de automotores. Otra importante utilización de giroscopios se lo percibe desde la antigüedad hasta nuestros días en el campo aeronáutico ya que la actitud de la aeronave depende de datos obtenidos por sensores giroscópicos para ser visualizados en los instrumentos de cabina por ejemplo el: horizonte artificial, indicador de rumbo, indicador de virajes entre otros.

La implementación de un vehículo de dos ruedas auto-equilibrado permitirá aplicar los conocimientos adquiridos durante el estudio en la carrera de Electrónica y desarrollar nuevas destrezas de programación e interpretación de datos de una IMU.

El proyecto además ayudará a que los estudiantes puedan relacionar la teoría con la práctica ya que los conocimientos teóricos son fundamentales al momento de utilizar dispositivos electrónicos.

El proceso de enseñanza-aprendizaje debe crecer e innovarse conjuntamente con el fin de desarrollar proyectos enfocados en beneficio de la sociedad en general.

1.4 OBJETIVO GENERAL

- Implementar un vehículo de dos ruedas auto-equilibrado utilizando Arduino para prácticas de microcontroladores

1.4.1 OBJETIVOS ESPECÍFICOS

- Investigar el funcionamiento de una IMU mediante el análisis de sus características para una correcta aplicación en la implementación del vehículo.
- Implementar el algoritmo de control para el funcionamiento del vehículo a través de librerías existentes en la plataforma de programación.
- Realizar la sintonía PID para el control de equilibrio mediante diversos métodos de sintonización

1.5 ALCANCE

El vehículo autónomo de dos ruedas está compuesto de tres partes principales, conteniendo sensores, unidad de procesamiento lógico y actuador (Figura 1).

Como sensor se utilizará un giróscopo MPU6050 el cual dará la desviación en los tres ejes existentes. La unidad de procesamiento lo constituye una tarjeta Arduino Mega que es óptima por sus características y como actuadores serán dos motores con su respectivo sistema reductor que permitirán el giro de las ruedas.

En el presente proyecto el vehículo puede mantener una posición vertical y en equilibrio, si es sometido a leves y medianas perturbaciones que cambie su estado el vehículo se moverá hasta colocarse equilibradamente, en caso

que las perturbaciones sean fuertes el vehículo perderá su equilibrio y caerá sin poderse recuperar y volver a su estado inicial.

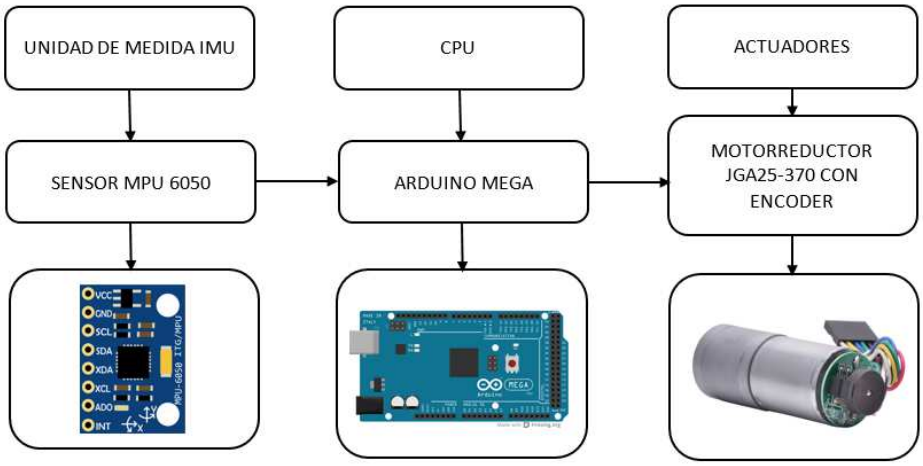

Figura 1 Diagrama de funcionamiento del vehículo

CAPÍTULO II

MARCO TEÓRICO

2.1 ¿QUÉ ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado para que realice desde un simple parpadeo de un led hasta un sofisticado control de un robot.

Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc., simplificando todo el diseño a una placa de reducido tamaño y pocos elementos. (Reyes, 2008)

2.1.1 CARACTERÍSTICAS DE UN MICROCONTROLADOR

Se compone de tres bloques fundamentales: la CPU (Central Processing Unit), la memoria, e interfaces de entrada y salida. Los bloques se conectan entre sí mediante grupos de líneas eléctricas denominados buses.

Los buses pueden ser de direcciones (si transportan direcciones de memoria o de entrada y salida), de datos (si transportan datos o instrucciones) o de control (si transportan señales de control diversas) (Figura 2). (Pérez, 2007)

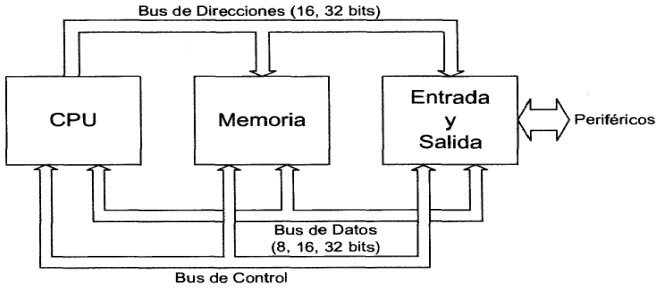

Figura 2 Esquema básico general de un microcontrolador

Fuente: (Pérez, 2007)

2.1.2 FABRICANTES DE MICROCONTROLADORES

- **Microchip:**

Microchip Technology Inc. es una empresa fabricante de microcontroladores, memorias y semiconductores analógicos, situada en Chandler, Arizona, EE. UU. Su Producto más popular son los microcontroladores PIC de 8 bits.

- **Atmel Corporation:**

Su línea de productos incluye microcontroladores (incluyendo derivados del 8051, el AT91SAM basados en ARM, y sus arquitecturas propias AVR y AVR32), dispositivos de radiofrecuencia, memorias EEPROM y Flash, ASICs, WiMAX, y muchas otras.

- **Texas Instruments:**

Texas Instruments o TI, es una empresa norteamericana que desarrolla y comercializa semiconductores y tecnología para sistemas de cómputo. Igualmente, es el mayor productor de procesadores digitales de señal y semiconductores analógicos.

TI es el tercer mayor fabricante de semiconductores del mundo tras Intel y Samsung y es el mayor suministrador de circuitos integrados para teléfonos móviles.

- **Motorola:**

Motorola Empresa dedicada a fabricar microprocesadores y microcontroladores entre otros productos, su mayor logro en la industria fue poner al Mercado un microprocesador de 8 bits, llamado 6800.

- **Intel:**

Intel empresa dedicada a la fabricación de microcontroladores y microprocesadores, aunque no trabajaba sola obtuvo un logro en abril de 1974 pone en el Mercado el microprocesador bajo el nombre 8080 con capacidad

de direccionar 64kb de memoria, con 75 instrucciones y un precio de inicio de $360 dólares. (Sánchez., 2013)

Figura 3 PIC 16f877A fabricado por Microchip

Fuente: (Sánchez., 2013)

2.1.3 ENCAPSULADOS DE MICROCONTROLADORES

- **DIP:**

Los pines se extienden a lo largo del encapsulado (en ambos lados) y tiene como todos los demás una muesca que indica el pin número 1. Este encapsulado básico fue el más utilizado hace unos años y sigue siendo el preferido a la hora de armar plaquetas por partes de los amantes de la electrónica casera debido a su tamaño lo que facilita la soldadura (Figura 4).

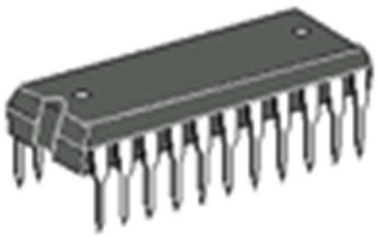

Figura 4 Encapsulado DIP

Fuente: (Herramientas Informaticas, 2012)

- **SIP:**

Los pines se extienden a lo largo de un solo lado del encapsulado y se lo monta verticalmente en la plaqueta (Figura 5). La consiguiente reducción en la zona de montaje permite una densidad de montaje mayor a la que se obtiene con el DIP.

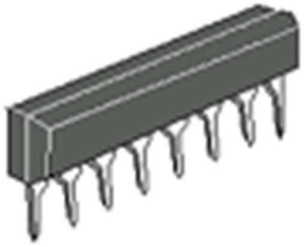

Figura 5 Encapsulado SIP

Fuente: (Herramientas Informaticas, 2012)

- **PGA:**

Los múltiples pines de conexión se sitúan en la parte inferior del encapsulado. Este tipo se utiliza para CPUs de PC y era la principal opción a la hora de considerar la eficiencia pin-capsula-espacio antes de la introducción de BGA (Figura 6).

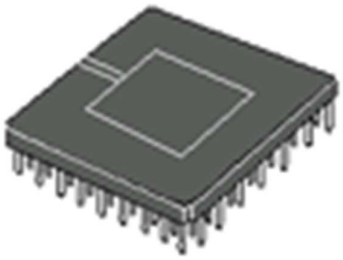

Figura 6 Encapsulado PGA

Fuente: (Herramientas Informaticas, 2012)

- **SOP:**

Los pines se disponen en 2 tramos más largos y se extienden en una forma denominada “gull wing formation”, este es el principal tipo de montaje

superficial y es ampliamente utilizado especialmente en los ámbitos de la microinformática, memorias y IC analógicos que utilizan un número relativamente pequeño de pines (Figura 7).

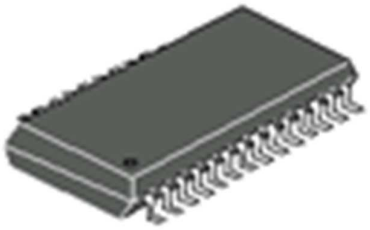

Figura 7 Encapsulado SOP

Fuente: (Herramientas Informaticas, 2012)

- **QFP:**

Es la versión mejorada del encapsulado SOP, donde los pines de conexión se extienden a lo largo de los cuatro bordes (Figura 8). Este es en la actualidad el encapsulado de montaje superficial más popular, debido que permite un mayor número de pines.

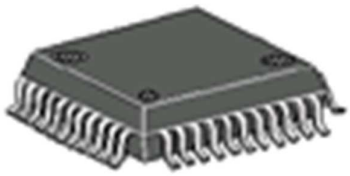

Figura 8 Encapsulado QFP

Fuente: (Herramientas Informaticas, 2012)

- **SOJ:**

Las puntas de los pines se extienden desde los dos bordes más largos dejando en la mitad una separación como si se tratase de 2 encapsulados en uno. Recibe éste nombre porque los pines se parecen a la letra “J” cuando se lo mira desde el costado (Figura 9).

Figura 9 Encapsulado SOJ

Fuente: (Herramientas Informaticas, 2012)

- **TCP**:

El chip de silicio se encapsula en forma de cintas de películas, se puede producir de distintos tamaños, el encapsulado puede ser doblado. Se utilizan principalmente para los drivers de los LCD (Figura 10).

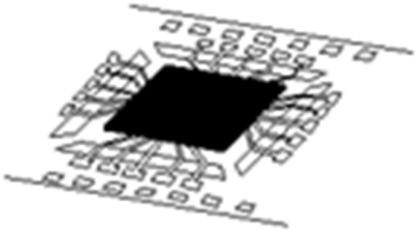

Figura 10 Encapsulado TCP

Fuente: (Herramientas Informaticas, 2012)

- **BGA:**

Los terminales externos, en realidad esferas de soldadura, se sitúan en formato de tabla en la parte inferior del encapsulado. Este encapsulado puede obtener una alta densidad de pines, comparado con otros encapsulados (Figura 11).

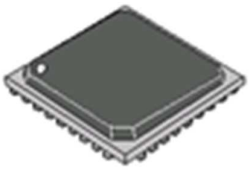

Figura 11 Encapsulado BGA

Fuente: (Herramientas Informaticas, 2012)

- **LGA:**

Es un encapsulado con electrodos alineados en forma de array en su parte inferior. Es adecuado para las operaciones donde se necesita alta velocidad debido a su baja inductancia. Además, en contraste con el BGA, no tiene esferas de soldadura por lo cual la altura de montaje puede ser reducida (Figura 12). (Herramientas Informaticas, 2012)

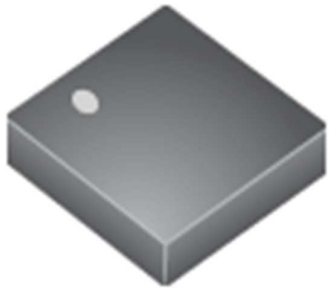

Figura 12 Encapsulado LGA

Fuente: (Herramientas Informaticas, 2012)

2.1.4 MICROCONTROLADORES Y MICROPROCESADORES

Históricamente, los microcontroladores aparecieron con posterioridad a los microprocesadores y han tenido evoluciones independientes. Los microprocesadores se han desarrollado fundamentalmente orientados al mercado de los ordenadores personales y las estaciones de trabajo, donde se requiere una elevada potencia de cálculo, el manejo de gran cantidad de memoria y una gran velocidad de procesamiento.

Un parámetro importante en los microprocesadores es el tamaño de sus registros internos (8, 16, 32 o 64 bits), que determina la cantidad de bits que pueden procesar simultáneamente.

Los microcontroladores se han desarrollado para cubrir las más diversas aplicaciones. Se usan en automoción, en equipos de comunicaciones y de telefonía, en instrumentos electrónicos, en equipos médicos e industriales de todo tipo, en electrodomésticos, en juguetes, etc.

Los microcontroladores están concebidos fundamentalmente para ser utilizados en aplicaciones puntuales, es decir, aplicaciones donde el

microcontrolador debe realizar un pequeño número de tareas, al menor costo posible.

En estas aplicaciones, el microcontrolador ejecuta un programa almacenado permanentemente en su memoria, el cual trabaja con algunos datos almacenados temporalmente e interactúa con el exterior a través de las líneas de entrada y salida de que dispone. (Pérez, 2007)

2.2 TARJETAS ENTRENADORAS (Plataformas de Desarrollo)

Las plataformas de desarrollo cumplen un importante rol dentro de la industria Electrónica, permitiendo reducir los tiempos involucrados en el diseño de una solución, aumentando la confiabilidad y velocidad de fabricación de un prototipo y, en ocasiones, transformándose en la base del producto final mismo.

Por lo general, estas son placas que integran microcontroladores, circuitos y componentes electrónicos que le proporcionan diversas capacidades básicas, como alimentación de energía o comunicación. De este modo, los desarrolladores ya no necesitan dedicarse a implementar una determinada funcionalidad para un proyecto, sino que simplemente deben elegir la plataforma de desarrollo que funcione con el microcontrolador de su preferencia y que cuente con las prestaciones adecuadas. (Microbyte, 2005)

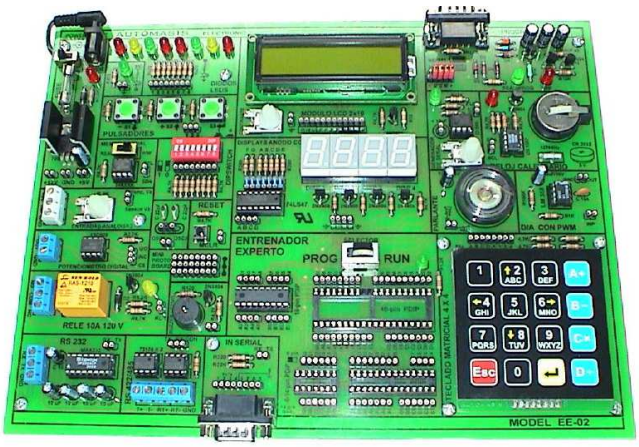

Figura 13 Tarjeta Entrenadora ee-02 de AUTOMASIS

Fuente: (Reyes, 2008)

2.2.1 ARDUINO VR. PINGÜINO

Entre las muchas placas de microcontroladores que salen cada mes una de las opciones más económicas es Pingüino, un clon de Arduino con un PIC de Microchip en lugar de un AVR. Con un entorno y librerías como las de Arduino.

Pingüino es un proyecto de "hardware libre" realizado por Jean Pierre Mandon, pretende que de una forma rápida y sencilla cualquiera pueda realizar y programar proyectos de electrónica con un microcontrolador. Sin necesidad de ser un experto en éstos se puede llegar a hacer cosas más o menos complejas. La filosofía es la misma que Arduino pero sin tanto interés comercial detrás. (Rivera M. A., 2010)

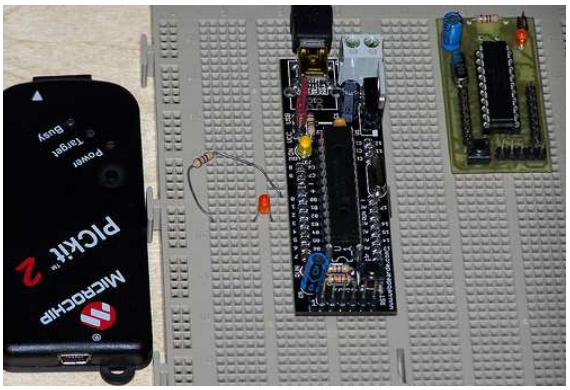

Figura 14 Tarjeta PINGÜINO

Fuente: (Rivera M. A., 2010)

2.3 ¿QUÉ ES ARDUINO?

Arduino es una herramienta para hacer que los ordenadores puedan sentir y controlar el mundo físico a través de tu ordenador personal. Es una plataforma de desarrollo de computación física (physical computing) de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software (programas) para la placa. (Cádiz, 2011)

2.3.1 FAMILIA ARDUINO

Hay muchas versiones de Arduino, en forma y tamaño. Esto nos ayuda en la variedad de aplicaciones que les podemos dar. Cada una de las placas o

versiones de Arduino tiene sus ventajas y desventajas, esto dependerá de las necesidades de nuestro proyecto.

Las aplicaciones que podemos realizar con Arduino están limitadas solamente por nuestra imaginación.
Cada placa soporta diferentes tipos de Shields, que dotan de capacidades extras a nuestras placas de Arduino, cabe destacar que no todas las placas de Arduino soportan Shields. (Ramos, 2014)

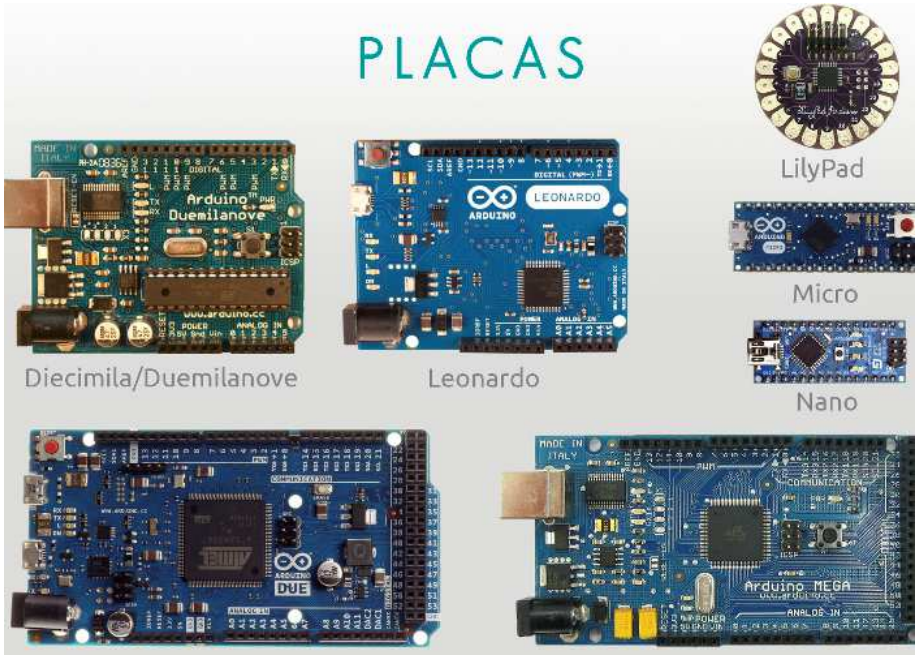

Figura 15 Familia ARDUINO

Fuente: (Ramos, 2014)

2.4 ¿QUÉ ES UNA SHIELD?

Son placas que pueden ser conectadas sobre la placa Arduino extendiendo sus capacidades, pudiendo ser apilada una encima de la otra. Las diferentes Shields siguen la misma filosofía que el conjunto original: son fáciles de montar, y baratas de producir. Es posible apilar varias Shields (Figura16). (Ramos, 2014)

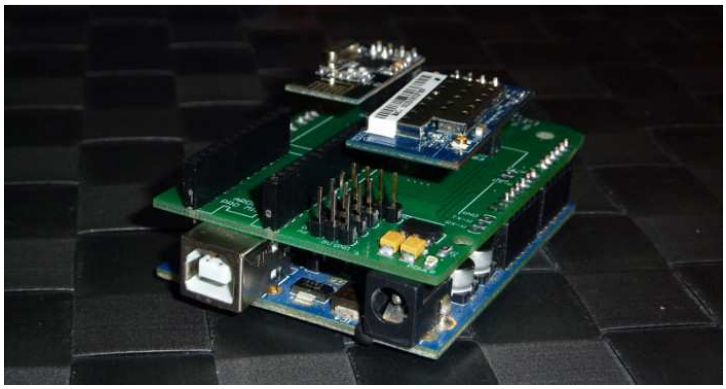

Figura 16 Shield montado sobre Arduino Uno

Fuente: (Ramos, 2014)

2.5 ARDUINO MEGA

Arduino Mega 2560 es una placa electrónica basada en el Atmega2560. Cuenta con 54 pines de entrada / salida digital (de los cuales 14 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, y un botón de reinicio (Figura 17).

Contiene todo lo necesario para apoyar al microcontrolador, basta con conectarlo a un ordenador con un cable USB o de potencia con un adaptador de CA a CC o una batería para empezar. Arduino Mega es compatible con la mayoría de Shields. (Arduino CC, 2010)

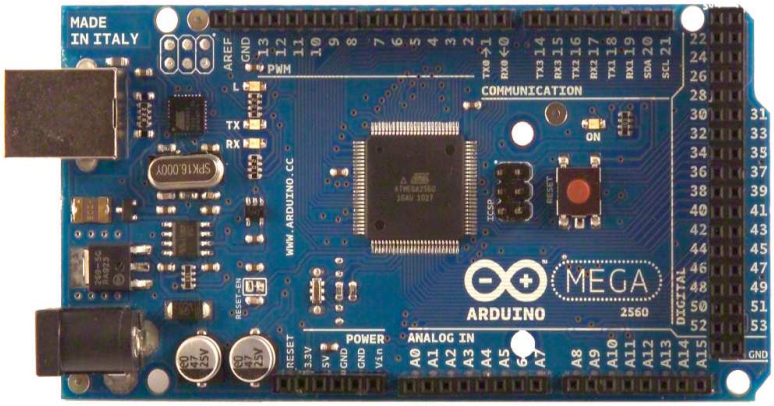

Figura 17 Arduino Mega

Fuente: (Arduino CC, 2010)

2.5.1 ESPECIFICACIONES TÉCNICAS

Tabla 1

Especificaciones Técnicas Arduino Mega

Microcontroller	**ATmega2560**
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Fuente: (Arduino CC, 2010)

2.5.2 COMPONENTES PRINCIPALES DE ARDUINO MEGA

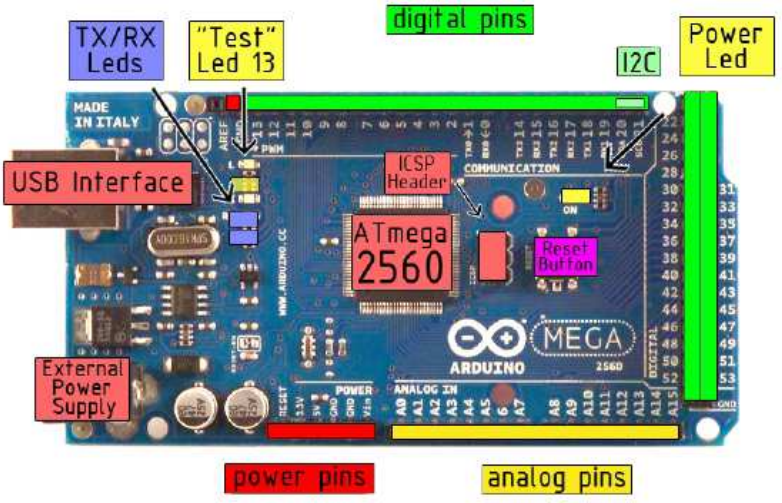

Figura 18 Componentes Arduino Mega

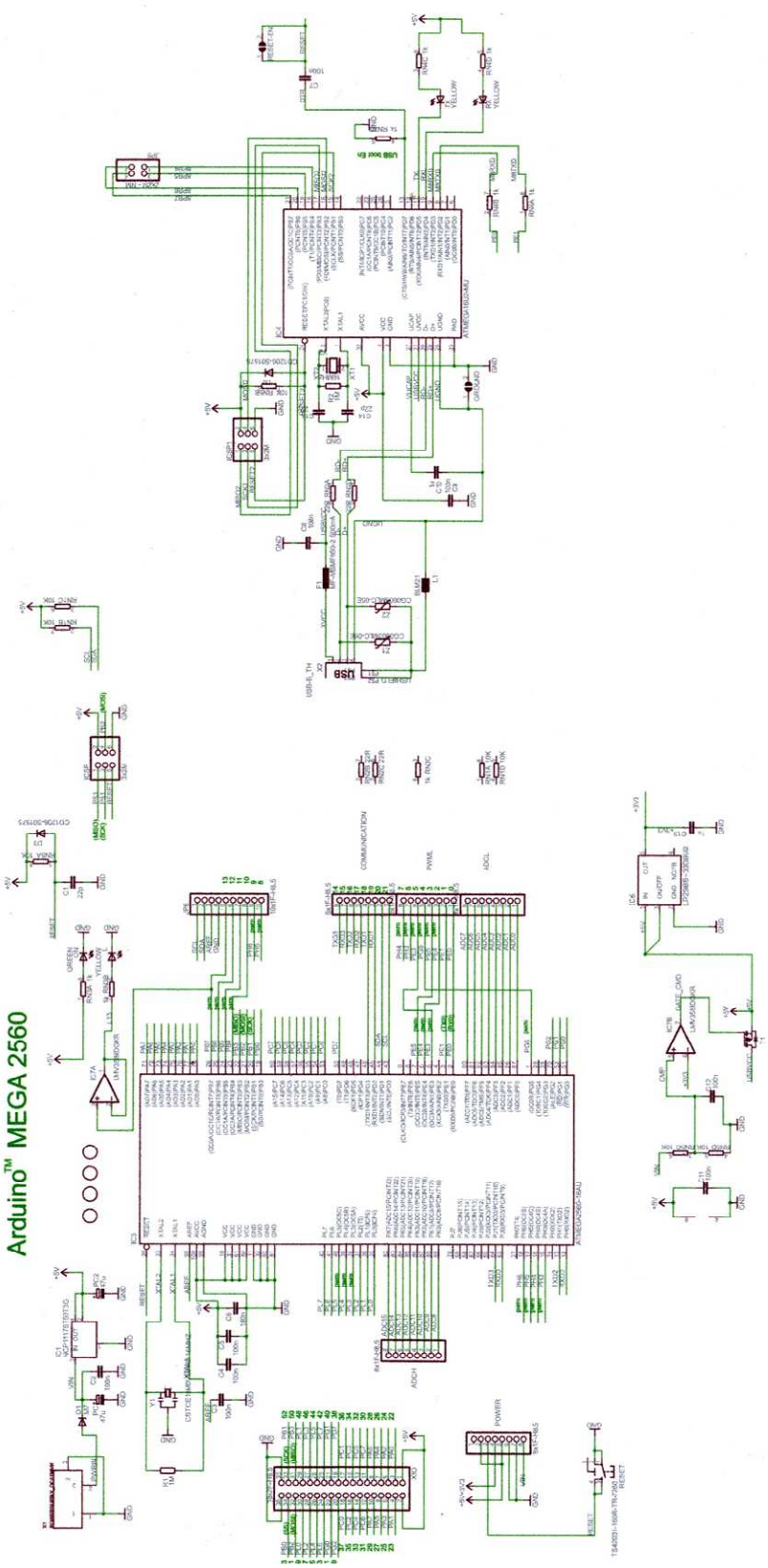

Figura 19 Esquema Interior Arduino Mega

Fuente: (Arduino CC, 2010)

2.6 ENTORNO DE DESARROLLO PARA ARDUINO

El Entorno de Desarrollo Arduino permite la conexión del ordenador con la placa para cargar los programas y comunicarse con ésta. El programa o "sketch" se escribe en el editor de texto (se puede cortar, copiar, pegar, etc., como en los editores de texto habituales). En el área de mensajes se muestra información mientras se cargan los programas y también muestra los errores.

La consola inferior muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas contiene los comandos más habituales (Figura 19):

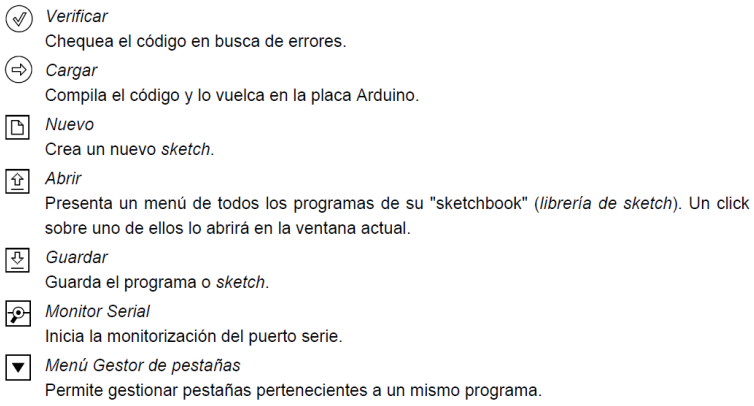

Figura 20 Barra de Herramientas

Fuente: (Technology, 2013)

Existen cinco menús: Archivo, Editar, Sketch, Herramientas y Ayuda. Los menús son sensibles al contexto, lo que significa que estarán disponibles sólo los elementos relevantes para la tarea que estemos realizando. A continuación se recogen sólo algunas de las opciones más habituales, aunque hay más.

- **Archivo**
  - *Ejemplos*

    Abre un submenú con los ejemplos que trae el programa.
  - *Guardar como...*

    Permite guardar el archivo con otro nombre u otra ubicación.

- **Editar**
  - *Deshacer*

    Deshacer: permite volver atrás si se ha modificado algo.
  - *Rehacer*

    Rehacer una acción.
  - *Cortar*

    Corta y almacena el texto seleccionado en el portapapeles para pegarlo en otro lugar.
  - *Copiar*

    Copia y almacena el texto seleccionado en el portapapeles para pegarlo en otro lugar.
  - *Pegar*

    Pega el texto que se encuentra en el portapapeles en el lugar seleccionado.
  - *Seleccionar todo*

    Selecciona todo el código.
  - *Buscar…*

    Busca una palabra en el código.

- **Herramientas**
  - *Formato automático*

    Da formato al código proporcionando estética: por ejemplo realiza tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo serán.
  - *Monitor Serial*

    Abre el monitor serie donde se visualizan los datos del puerto serie o se envían datos al mismo. Para enviar datos a la placa, teclee el texto y pulse el botón "send" o enter. Seleccione la velocidad (*baud rate*) en

el menú desplegable que coincida con el configurado en **Serial.begin** dentro de su "*sketch*".

- *Tarjeta*

Selecciona la placa que estás usando (Arduino Mega 2560).

- *Puerto Serial*

Este menú contiene todos los dispositivos series (reales o virtuales) de su equipo. Se refrescará automáticamente cada vez que abra el menú **Herramientas**. Antes de subir nuestro *sketch* a la placa hay que seleccionar el elemento de este menú que representa a nuestra placa Arduino (normalmente COM3 o COM4). (Technology, 2013)

2.7 ESTRUCTURA DE UN Sketch

El código tiene 3 partes principales:

- La zona global
- La función **void setup()**
- La función **void loop()**

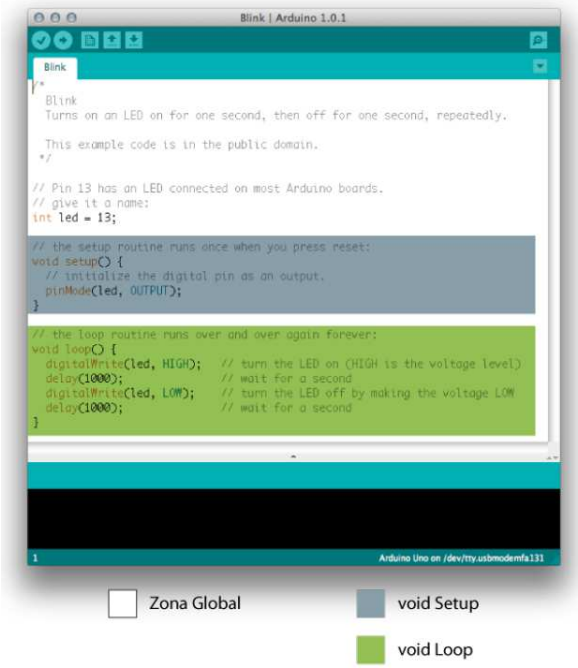

Figura 21 Partes de un Sketch

Fuente: (MEDIALAB, 2010)

2.7.1 setup() y loop()

- **Zona global**

  Aquí será donde se indica a Arduino los nombres de los pines y donde crearemos aquellas variables que queramos que existan en todo el programa. En el caso del ejemplo “parpadeo” (Figura 22), únicamente tenemos lo siguiente:

  ```
  int led = 13;
  ```

  Con esto se crea una variable en la que se guarda el número del pin que utilizaremos conectado al led.

- **La función void setup()**

  Esta función se ejecuta cada vez que se inicia Arduino (incluyendo al pulsar RESET). Una de las operaciones que se realiza en ***void setup()*** es la de configurar de los pines que vamos a utilizar. En el caso del ejemplo “parpadeo” (Figura 22):

  ```
  void setup() {
  pinMode(led, OUTPUT);
  }
  ```

  Como sólo se usa un pin, llamado “led”, sólo existe una función de configuración “pinMode” en la que indicamos que lo usaremos como salida.

- **La función void loop()**

  Esta función es el corazón de los programas creados con Arduino. Es una función que permanece en ejecución en forma de bucle infinito. Esto quiere decir que se ejecuta de comienzo a fin, de forma repetida, siempre. Esto queda más claro viendo el contenido del ejemplo “parpadeo” (Figura 22): (MEDIALAB, 2010)

Figura 22 Ejemplo "Parpadeo"

2.7.2 ESTRUCTURAS DE CONTROL

- **if...else (comparador si... sino)** (Figura 23).

```
Si el led está apagado
{
        enciendelo
}
sino
{
        apágalo
}
```

```
if(digitalRead(led) == LOW)
{
        digitalWrite(led, HIGH);
}
else
{
        digitalWrite(led, LOW);
}
```

Figura 23 Ejemplo "Comparador"

Fuente: (MEDIALAB, 2010)

- **for (bucle con contador)** (Figura 24)**.**

Para que (desde el valor 0; hasta el valor 100; aumenta de 1 en 1)
{
enciende el led
espera 1 segundo
apaga el led
espera un segundo
}
	int cont;
	for(cont=0; cont<100; cont=cont+1)
	{
	digitalWrite(led, HIGH);
	delay(1000);
	digitalWrite(led, LOW);
	delay(1000);
	}

Figura 24 Ejemplo "Bucle contador"

Fuente: (MEDIALAB, 2010)

- **while (bucle por comparación)** (Figura 25).

Mientras ocurra (hay luz)
{
Apaga Led
}
	while(INPUT==HIGH)
	{
	DigitalWrite = LOW;
	}

Figura 25 Ejemplo "Bucle comparador"

Fuente: (MEDIALAB, 2010)

2.7.3 ELEMENTOS DE SINTAXIS

Como se observa en el ejemplo anterior (Figura 22), al concluir cada declaración, se finaliza con *";"*. Para realizar algún comentario dentro del programa y en una línea se antepondrá *"//"* al comentario, si se desea realizar un bloque de comentarios es decir en varias líneas, se utiliza la siguiente sintaxis *"/*… */"*. Mientras tanto los símbolos *"{"* y *"}"*, indican inicio y fin respectivamente de una función. (TOVAR, 2014)

2.7.4 OPERADORES

El uso de operadores es similar al utilizado en el lenguaje de programación C y C++, se clasifican en aritméticos (Tabla 2), booleanos (Tabla 3), compuestos (Tabla 4) y de comparación (Tabla 5). (TOVAR, 2014)

Tabla 2

Operadores aritméticos

OPERADOR	ACCIÓN REALIZADA
=	Operador de asignación.
+	Operador de suma.
-	Operador de resta.
*	Operador de multiplicación.
/	Operador de división.

Fuente: (TOVAR, 2014)

Tabla 3

Operadores booleanos

OPERADOR	ACCION REALIZADA
&&	AND Lógico
||	OR Lógico
!	NOT Lógico
&	Operador de bits AND
|	Operador de bits OR
~	Operador de bits NOT

Fuente: (TOVAR, 2014)

Tabla 4

Operadores compuestos

OPERADOR	ACCION REALIZADA
x++	Incrementa a x en 1 y devuelve el valor antiguo de x.
++x	Incrementa a x en 1 y devuelve el nuevo valor de x.
x--	Decrementa en 1 a x y devuelve el valor antiguo de x.
--x	Decrementa a x en 1 y devuelve el nuevo valor de x.

Fuente: (TOVAR, 2014)

Tabla 5

Operadores comparación.

OPERADOR	ACCION REALIZADA
= =	Igual.
! =	Diferente.
<	Menor que.
>	Mayor que
<=	Menor o igual que.
>=	Mayor o igual que.

Fuente: (TOVAR, 2014)

2.8 VARIABLES

Una variable es una forma de asignar o almacenar un valor para un uso posterior por el programa, estos datos pueden venir de un sensor o alguna constante existente, los cuales permiten realizar un cálculo o realizar una acción determinada. (TOVAR, 2014)

2.8.1 CONSTANTES

Las constantes son variables predefinidas en el lenguaje de Arduino. (Technology, 2013). Las cuales se utilizan para asignar un valor a una variable o configurar un pin como entrada/salida. (TOVAR, 2014)

2.8.2 TIPOS DE DATOS

Tabla 6

Tipos de Datos

Tipo	Tamaño	Rango	Descripción
boolean	1 bit	0 a 1	Entero de 1 bit
byte	8 bit	0 a 255	Entero
unsigned char	8 bit	0 a 255	Entro sin signo
char	8 bit	-128 a 127	Entero con signo
int	8 bit	-32,768 a 32,767	Entero de 8 bit con signo
word unsigned int	16 bit	0 - 65,535	Entero de 16 bit
long	16 bit	-2,147'483,648 a 2,147'483,647	Entero de 16 bit con signo
unsigned long	4 byte	0 a 4,294'967,295	Entero sin signo de 4 bits
float / double	4 byte	-3.4028235e+37 a 3.4028235e+38	Decimal con signo de 4 bytes

Fuente: (Alazte, 2016)

2.9 FUNCIONES

2.9.1 ENTRADAS/SALIDAS DIGITALES

- **pinMode(**pin, modo**)**
  Configura el pin especificado para comportarse en modo INPUT (entrada) o en modo OUTPUT (salida). No devuelve nada.
  Ver ejemplo (Figura 26)

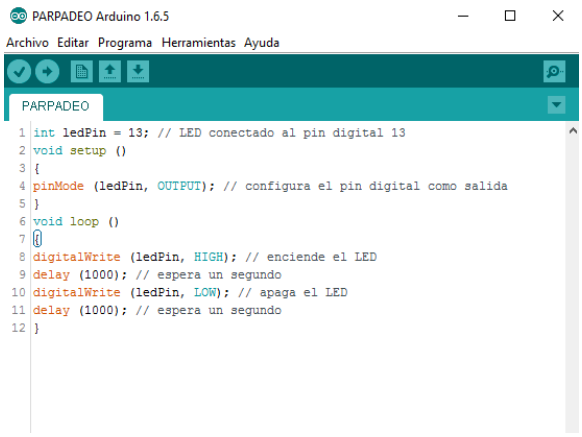

Figura 26 Ejemplo "Parpadeo de un led"

- **digitalWrite(**pin, valor**)**

  Escribe un valor HIGH o LOW en el pin digital especificado. No devuelve nada. Si el pin se ha configurado como OUTPUT (salida) con pinMode(), su tensión se establece en 5V para HIGH y a 0V (tierra) para LOW.

  Ver ejemplo (Figura 26).

- **digitalRead(**pin**)**

  Lee el valor del pin digital especificado. Devuelve o HIGH o LOW. (Technology, 2013)

  Ver ejemplo (Figura 27).

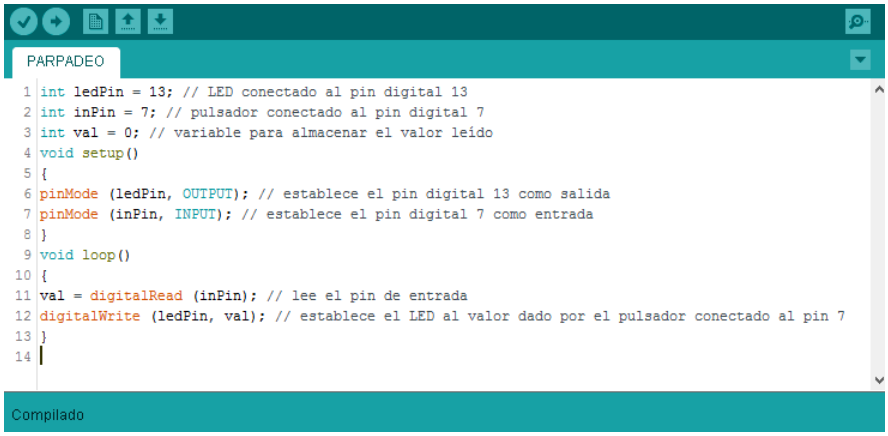

Figura 27 Ejemplo "Lectura de un pulsador"

2.9.2 ENTRADAS/SALIDAS ANALOGAS

- **analogRead(**pin**)**

  Lee el valor de tensión en el pin analógico especificado (0 a 5). La placa dispone de un convertidor analógico-digital que asignará a voltajes de entrada de entre 0 y 5 V valores enteros entre 0 y 1023, con una resolución de 10 bits. Por tanto, esta función devuelve un valor entero entre 0 y 1023.

  Ver ejemplo (Figura 28).

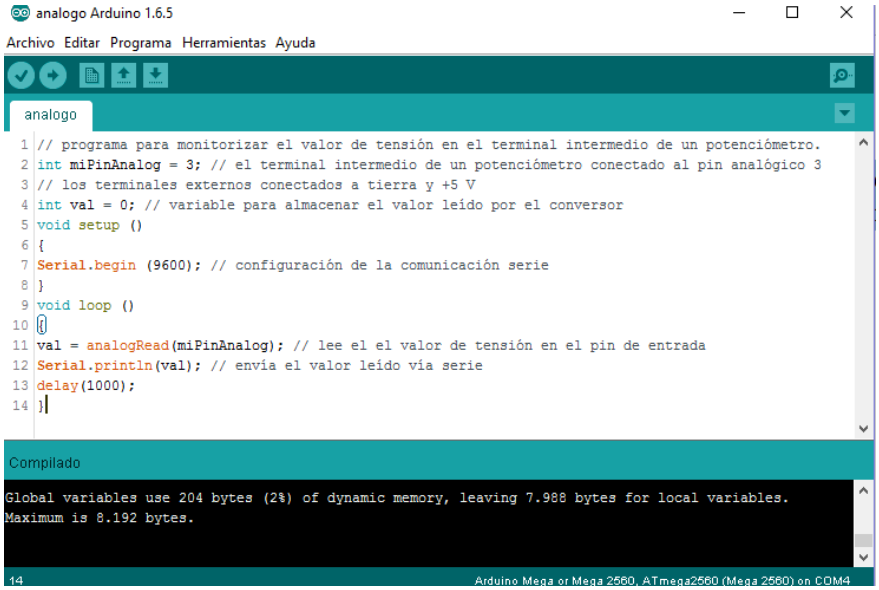

Figura 28 Ejemplo “Lectura de tensión de un potenciómetro”

- **analogWrite(**pin, valor**)**

  Escribe un valor (entre 0 y 255) pseudo-analógico (onda PWM) en el pin digital especificado. Se puede utilizar para encender un LED con brillo variable o hacer girar un motor a varias velocidades. No es necesario llamar a pinMode() para establecer el pin como salida para poder usar la función analogWrite(). (Technology, 2013)

  Ver ejemplo (Figura 29).

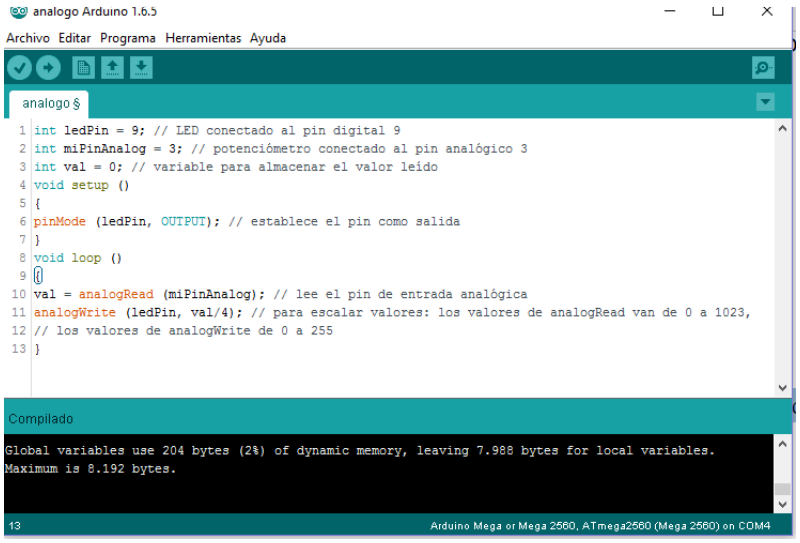

Figura 29 Ejemplo “Salida Análoga”

2.9.3 TEMPORIZADORES

- **delay(ms)**

  Pausa tu programa por la cantidad de tiempo especificada en milisegundos, donde 1000 es igual a 1 segundo.

  Ver ejemplo (Figura 22).

  ```
  delay(1000); //espera por un segundo
  ```

- **millis()**

  Devuelve el número de milisegundos desde que la placa Arduino empezó a ejecutar el programa actual como un valor long sin signo.

  ```
  value = millis(); //ajusta 'value' igual a millis()
  ```

  Nota: Este número se desbordará (resetear de nuevo a cero), después de aproximadamente 9 horas. (Technology, 2013)

2.9.4 FUNCIONES MATEMÁTICAS

- **min(x,y)**

  Calcula el mínimo de dos números de cualquier tipo de datos y devuelve el número más pequeño (Figura 30).

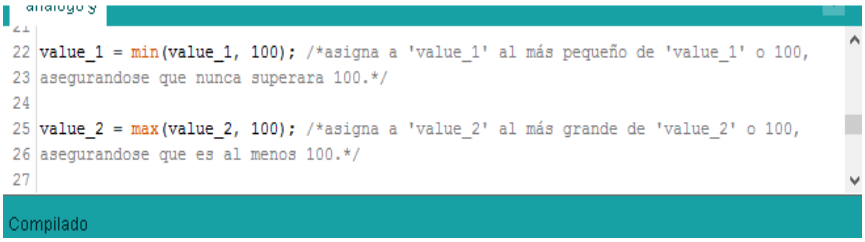

Figura 30 Utilización de min, max

- **max(x,y)**

  Calcula el máximo de dos números de cualquier tipo de datos y devuelve el número más grande (Figura 30). (Herrador, 2009)

- **abs (x)**

  Calcula el valor absoluto de un número. (Arduino CC, 2010)

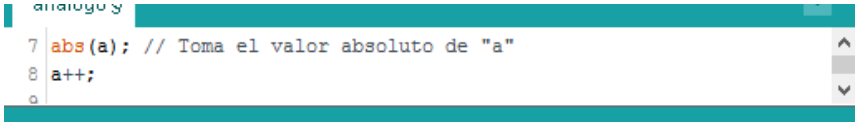

Figura 31 Valor absoluto de “a”

- **constrain(x, a, b)**

  Restringe un número para estar dentro de un rango. (Arduino CC, 2010) Ver ejemplo (Figura 32).

  - **Parámetros:**

    x: el número para limitar, todos los tipos de datos

    a: El extremo inferior de la gama, todos los tipos de datos

    b: el extremo superior de la gama, todos los tipos de datos

  - **Devoluciones**

    x: si **x** esta entre **a** y **b**

    a: si **x** es menor que **a**

    b: si **x** es mayor que **b**

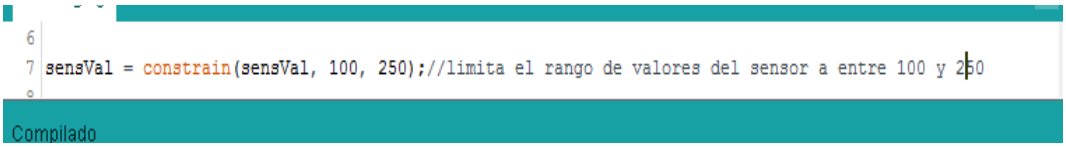

Figura 32 Ejemplo “Limite entre 100 y 250”

2.9.5 FUNCIÓN TRIGONOMÉTRICA atan2()

Hay otro camino, entre las funciones de "fabrica" de Arduino, está el arco tangente (***atan()*** *y* ***atan2(x,y)***). Como sé que la suma del cuadrado del seno y del cuadrado del coseno da uno, puedo calcular el seno a partir del coseno y el coseno a partir del seno y utilizar la función ***atan2()*** para saber el ángulo en radianes. (Castrillo, 2015) Ver ejemplo (Figura 33)

Figura 33 Ejemplo “Utilización de atan2()”

2.9.6 INTERRUPCIONES

El manejo de interrupciones en Arduino, nos permiten ejecutar eventos de manera asíncrona, es decir que el código principal reaccione a eventos externos sin necesidad de estarlos monitoreando continuamente.

- ***attachInterrup(pin, función, modo)***

  Esta función nos permite el manejo de interrupciones en un programa, si utilizamos esta función, definiremos los siguientes parámetros:

  Pin: especifica el número de entrada de interrupción

  Función**:** Se define el nombre de la función a ser llamada cuando ocurre la interrupción.

  Modo**:** Define la transición del pin para activar la interrupción.

Los modos en que se activa la interrupción son los siguientes:

LOW: Activa cuando el pin está en nivel bajo.
HIGH: Activa cuando el pin está en nivel alto
CHANGE: Activa cuando hay un cambio de estado en el pin.
RISING: Se activa cuando hay un cambio de nivel bajo a alto.
FALLING: Activa la interrupción cuando detecta un cambio de nivel alto a bajo. (TOVAR, 2014)

Tabla 7

Tarjetas y Pin de Interrupciones

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	all digital pins

Fuente: (Arduino CC, 2010)

- **interrupts()**
  Vuelve a habilitar las interrupciones (después de que han sido desactivadas por *noInterrupts()*).

- **noInterrupts()**
  Deshabilita las interrupciones (se puede volver a activarlos con *interrupts()*). (Arduino CC, 2010)

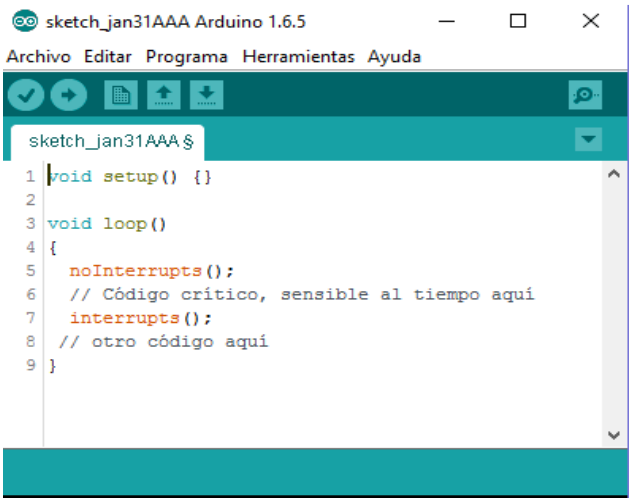

Figura 34 Activación de desactivación de Interrupciones

2.10 COMUNICACIÓN I2C

El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz.

La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos.

Descripción de las señales

SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.
GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET).

Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas. (Carletti E. J., 2006)

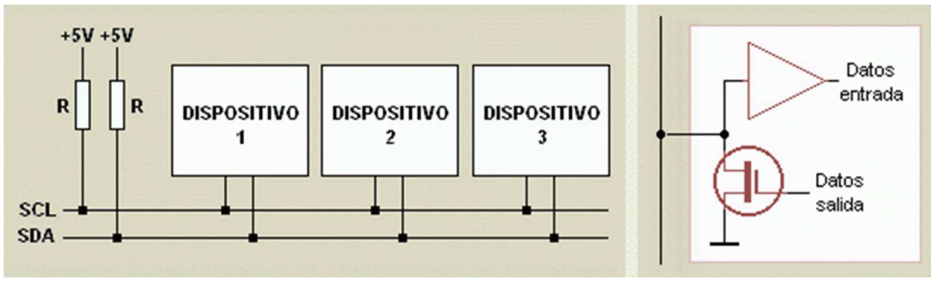

Figura 35 Comunicación I2C

Fuente: (Carletti E. J., 2006)

2.11 MOTORREDUCTOR JGA25-370 CON ENCODER

Este motorreductor JGA25-371 DC cuenta con un codificador integrado que ofrece una resolución de 12 pulsos por revolución, lo que garantiza un control preciso de la velocidad del motor.

Condiciones de Trabajo Estándar:

- **Tensión nominal:** 12 V DC constante entre los terminales del motor.
- **Dirección de rotación:** CW cuando se ve desde el lado del eje de salida.
- **Temperatura y humedad de funcionamiento:** Temperatura rango de -10°C a 50°C, Rango de humedad de 30% a 80%.
- **Temperatura de almacenamiento:** Rango de temperatura de -20°C a + 60°C.

Condiciones de medición:

- **Posición del motor:** Ubicar horizontalmente cuando se mide.
- **Fuente de alimentación:** fuente de alimentación regulada 5VDC.
- **Temperatura ambiental y humedad:** Rango de temperatura de 15°C a 30°C.

Características eléctricas (en la fase inicial después de 30 segundos acondicionarán)

- **Sin corriente de carga:** 50 ± S% ma.
- **Velocidad sin carga:** 201 rpm ± 5%.
- **Carga nominal:** 0.53Kg.cm
- **Corriente de carga nominal:** 250 ± 5% ma.
- **Velocidad nominal de carga:** 160 rpm ± 10%.
- **Potencia:** 1,25 W.

Características mecánicas

- **Aspecto externo:** Ver (Figura 37)
- **Pesos:** Aproximadamente: 100g.
- **Vida:** Sobre 500H. (Impulse, 2012)

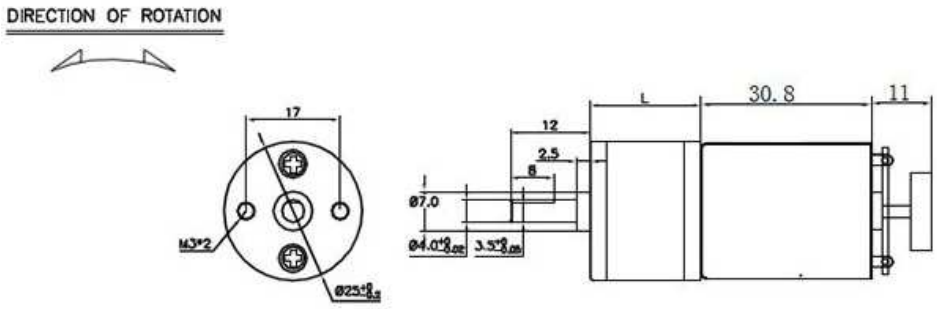

Figura 36 Dimensiones Motorreductor

Fuente: (Impulse, 2012)

Figura 37 Motorreductor **JGA25-370**

Fuente: (Impulse, 2012)

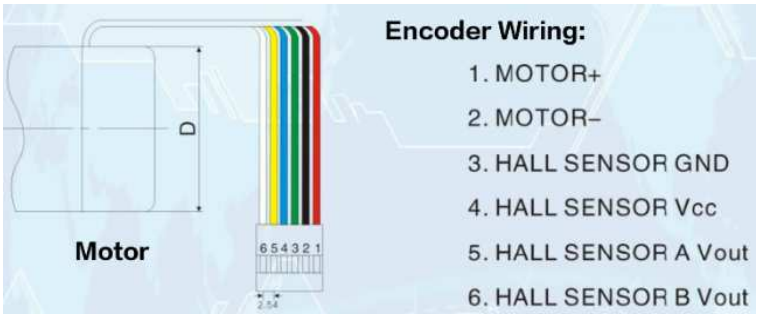

Figura 38 Diagrama de conexión Motorreductor

Fuente: (Impulse, 2012)

2.12 PUENTE H (H-BRIDGE)

Un Puente en H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica. Los puentes H están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos.

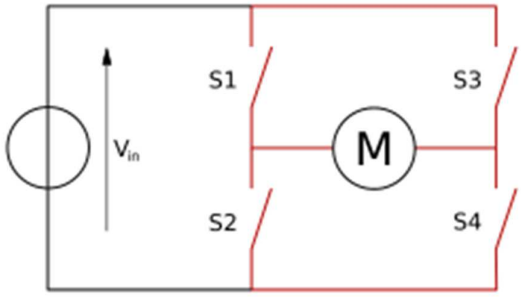

Figura 39 Estructura de un puente H (marcado en rojo)

Fuente: (Carletti E. , 2015)

El término "puente H" proviene de la típica representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 (Figura 39) están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Con la nomenclatura que estamos usando, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4. (Carletti E. , 2015)

Tabla 8

Tabla de verdad del puente H

S1	S2	S3	S4	Resultado
1	0	0	1	El motor gira en *avance*
0	1	1	0	El motor gira en *retroceso*
0	0	0	0	El motor se detiene bajo su inercia
1	0	1	0	El motor frena (*fast-stop*)

Fuente: (Carletti E. , 2015)

2.12.1 L298P

El L298 es un circuito integrado monolítico que puede llevar 15 o 20 pines. Funciona en alta tensión, es un doble puente completo diseñado para aceptar niveles lógicos TTL estándar y cargas inductivas tales como relés, solenoides, motores DC y motores paso a paso. (Datasheet ST, 2000)

Posee dos entradas de habilitación para habilitar o deshabilitar el dispositivo independientemente de la entrada señales.

Se proporciona entrada de alimentación de modo que la lógica trabaja en un voltaje más bajo. (Datasheet ST, 2000)

Tabla 9

Características técnicas L298p

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

Fuente: (Datasheet ST, 2000)

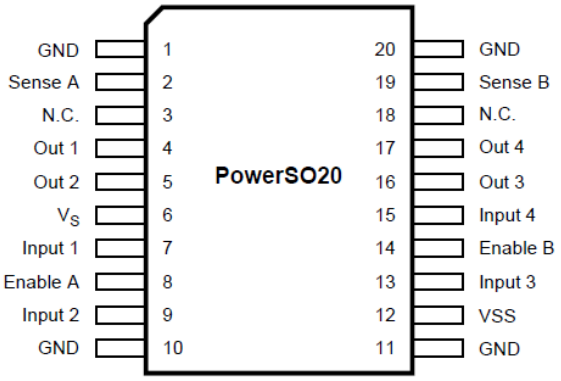

Figura 40 Diagrama de pines L298p

Fuente: (Datasheet ST, 2000)

2.13 SHIELD GIRÓSCOPO (MPU 6050)

El sensor MPU-6050 contiene un acelerómetro MEMS (Sistemas Microelectomecánicos) y un giroscopio MEMS en un solo chip. Es muy preciso, ya que contiene 16 bits de analógico a digital de hardware de conversión para cada canal. Para ello se captura la x, y, z y el canal al mismo tiempo. El sensor utiliza el microbús I2C para interconectarse con el Arduino. (Arduino CC, 2010)

Figura 41 MPU 6050

Fuente: (Arduino CC, 2010)

2.13.1 Diagrama de pines

- **VCC:** Alimentación 5v
- **GNG:** Tierra
- **SCL:** I2C serial clock
- **SDA:** I2C serial data

- **XDA:** Auxiliar I2C serial data
- **XCL:** Auxiliar I2C serial clock
- **ADO:** Direcciona la comunicación I2C
- **INT:** Interrupción para ver Disponibilidad de datos

2.14 CONTROL PROPORCIONAL INTEGRAL DERIVATIVO (PID)

Sistemas de **lazo cerrado** o sistemas con **realimentación** o feedback. La toma de decisiones del sistema no depende sólo de la entrada sino también de la salida.

El sistema es más flexible y capaz de reaccionar si el resultado que está obteniendo no es el esperado; los sistemas a los que podemos llamar robots casi siempre son de lazo cerrado.

Un sistema de riego en lazo cerrado, no se detendrá al cabo de un tiempo fijo, sino cuando detecte que se está consiguiendo el objetivo buscado. Y se pondrá en marcha, no a una hora determinada, sino en cualquier momento. (Educa Lab, 2012)

Un controlador PID es un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial. Este calcula la desviación o error entre un valor medido y un valor deseado.
El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral, y el derivativo.

- El valor Proporcional depende del error actual.
- El Integral depende de los errores pasados
- Derivativo es una predicción de los errores futuros.

La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control como la posición de una válvula de control o la potencia suministrada a un calentador.

Cuando no se tiene conocimiento del proceso, históricamente se ha considerado que el controlador PID es el controlador más adecuado. Ajustando estas tres variables en el algoritmo de control del PID, el controlador

puede proveer una acción de control diseñado para los requerimientos del proceso en específico.

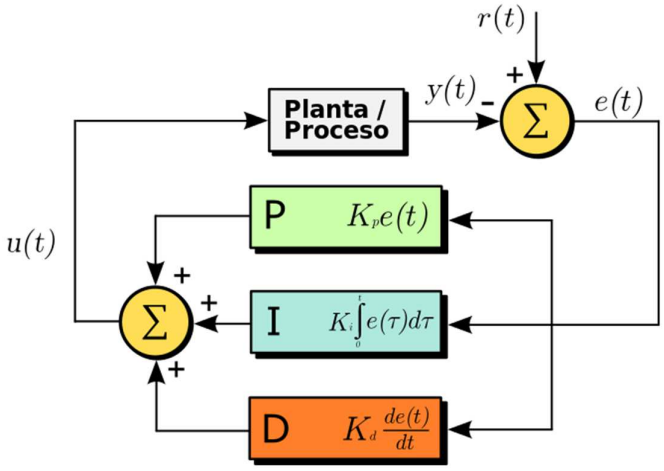

Figura 42 Diagrama de bloques Controlador PID

Fuente: (Klug, 2012)

Para el correcto funcionamiento de un controlador PID que regule un proceso o sistema se necesita, al menos:

- Un sensor, que determine el estado del sistema (termómetro, caudalímetro, manómetro, etc).
- Un controlador, que genere la señal que gobierna al actuador.
- Un actuador, que modifique al sistema de manera controlada (resistencia eléctrica, motor, válvula, bomba, etc).

Las 3 señales PID sumadas, componen la señal de salida que el controlador va a utilizar para gobernar al actuador. La señal resultante de la suma de estas tres se llama variable manipulada y no se aplica directamente sobre el actuador, sino que debe ser transformada para ser compatible con el actuador utilizado.

$$y(t) = MV(t) = Kpe(t) + Ki\int_0^t e\,(r)dr + Kd\frac{de}{dt}$$

Las tres componentes de un controlador PID son:

- Parte Proporcional (Kp)
- Parte Integral (Ki)

- Parte Derivativa (Kd)

El peso de la influencia que cada una de estas partes tiene en la suma final, viene dado por la parte proporcional, el tiempo integral y el tiempo derivativo, respectivamente. Se pretenderá lograr que el bucle de control corrija eficazmente y en el mínimo tiempo posible los efectos de las perturbaciones.

2.14.1 Parte proporcional

La respuesta proporcional es la base de los tres modos de control, si los otros dos, control integral y control derivativo están presentes, éstos son sumados a la respuesta proporcional. “Proporcional” significa que el cambio presente en la salida del controlador es algún múltiplo del porcentaje del cambio en la medición.

Este múltiplo es llamado “ganancia” del controlador. Para algunos controladores, la acción proporcional es ajustada por medio de tal ajuste de ganancia, mientras que para otros se usa una “banda proporcional”.

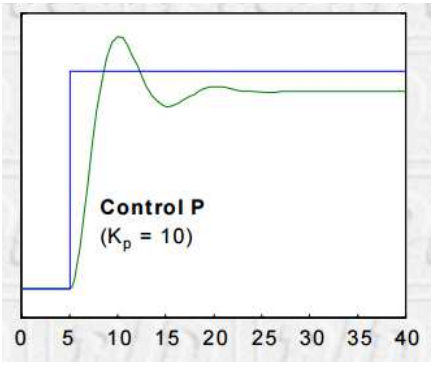

Figura 43 Control Proporcional

Fuente: (García, 2007)

2.14.2 Parte integral

La acción integral da una respuesta proporcional a la integral del error. Esta acción elimina el error en régimen estacionario, provocado por el modo proporcional.

Por contra, se obtiene un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.

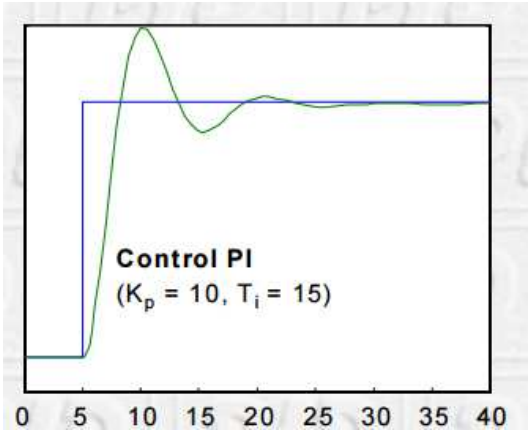

Figura 44 Control Proporcional Integral

Fuente: (García, 2007)

2.14.3 Parte derivativa

La acción derivativa da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se disminuye el exceso de sobreoscilaciones. (Klug, 2012)

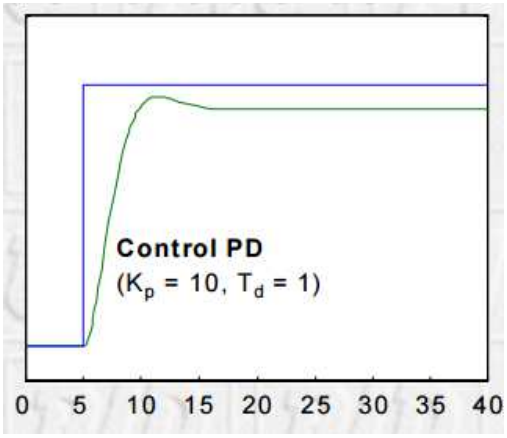

Figura 45 Control Proporcional Derivativo

Fuente: (García, 2007)

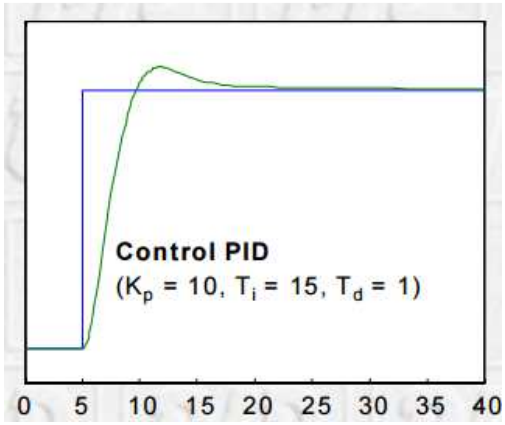

Figura 46 Control PID

Fuente: (García, 2007)

2.14.4 METODOS DE SINTONÍA

- **Método de tanteo**

  Este método requiere que el controlador y el proceso estén instalados completamente y trabajando en su forma normal. El procedimiento general se basa en poner en marcha el proceso con bandas anchas en todas las acciones, y estrecharlas después poco a poco individualmente, hasta obtener la estabilidad deseada.

  Es necesario que pase un tiempo suficiente después de cada desplazamiento del punto de consigna, para observar el efecto total del último ajuste obteniendo algunos ciclos de la respuesta ante la perturbación creada. En procesos muy lentos ello puede requerir hasta 2 o 3 horas.

  Para ajustar los controladores proporcionales, se empieza con una banda proporcional ancha y se estrecha gradualmente observando el comportamiento del sistema hasta obtener la estabilidad deseada. Hay que hacer notar que al estrechar la banda proporcional, aumenta la inestabilidad y que al ampliarla se incrementa el error de offset, tal como se ve. (Figura 47).

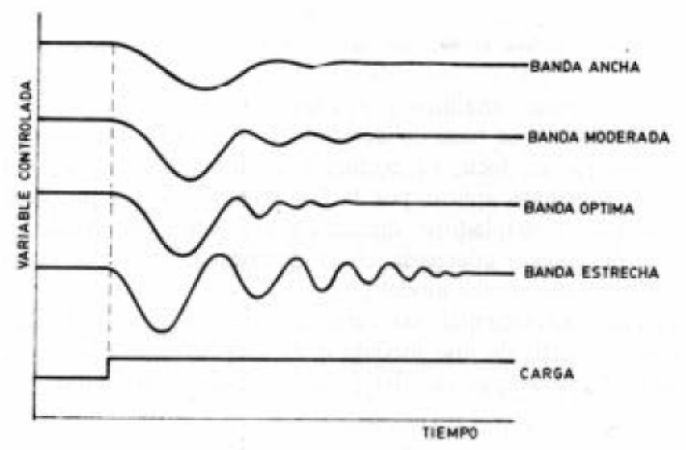

Figura 47 Método de tanteo

Fuente: (Rivera F. , 2010)

- **Método de ganancia límite**

Este método de lazo cerrado fue desarrollado por Ziegler y Nichols, en 1941y permite calcular los tres términos de ajuste del controlador a partir de los datos obtenidos en una prueba rápida de características del bucle cerrado de control. El método se basa en estrechar gradualmente la banda proporcional con los ajustes de integral y derivada en su valor más bajo, mientras se crean pequeños cambios en el punto de consigna, hasta que el proceso empieza a oscilar de modo continuo. Tanto la ganancia como el periodo crítico (Pcr) pueden ser determinados en forma experimental. (Rivera F. , 2010)

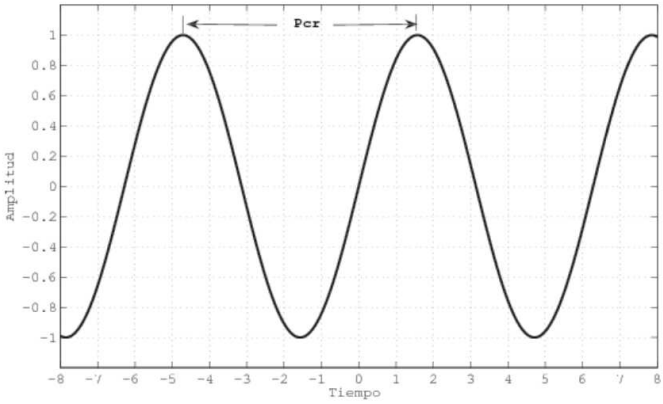

Figura 48 Método de Ganancia límite

Fuente: (Rivera F. , 2010)

- **Métodos de ajuste automático**

El instrumento controlador dispone de un algoritmo de autoajuste de las acciones de control que le permite sintonizar con una amplia gama de procesos industriales. Existen varias formas de realizar el autoajuste:

a) La aplicación de una señal de prueba al proceso, y el análisis de su respuesta con la obtención de un modelo matemático y el diseño analítico del controlador (método de Nishikawa, Sannomiya, Ohta y Tanaka), o bien el uso de las formas del método de ganancia límite de Ziegler y Nichols (método de Chindambara y método de Kraus y Myron)).

b) El análisis continuo u ocasional (ante una perturbación o una modificación del S.P.) del proceso sin aplicar señales de prueba, sin perturbar pues, el proceso, pero con el inconveniente de no detectar los cambios lentos del mismo. (Rivera F. , 2010)

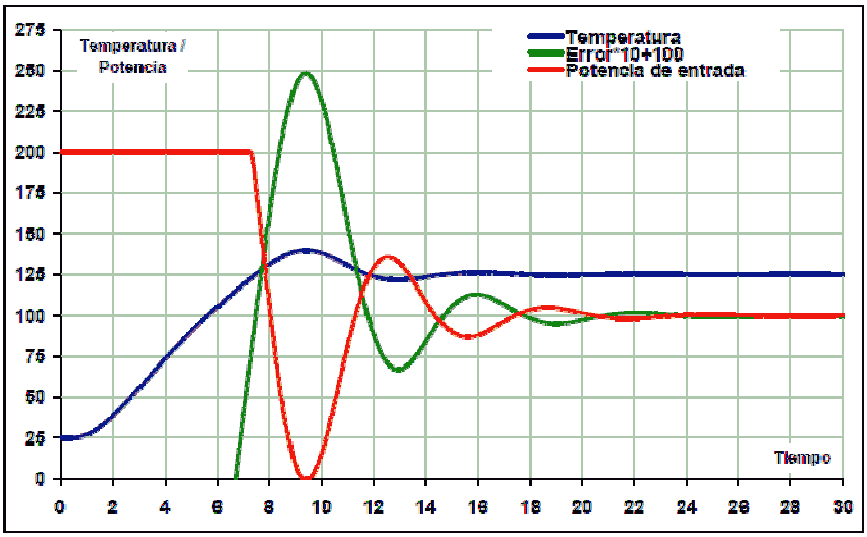

Figura 49 Método de Ajuste Automático por el método de Nichols

Fuente: (Rivera F. , 2010)

CAPÍTULO III

DESARROLLO DEL TEMA

3.1 Preliminares

En el presente capítulo se detalla paso a paso como se realiza la implementación de un vehículo de dos ruedas auto-equilibrado para prácticas de microcontroladores en donde se utilizará la placa Arduino Mega y su respectiva programación.

Se empleará la comunicación I2C de la shield MPU 6050 e intercambiaremos información hacia el microcontrolador, el mismo que nos permitirá controlar los motores mediante un driver.

3.2 Implementación del vehículo

2.2.1 Comprobación de materiales

Para la puesta en servicio del vehículo se empezará revisando cada elemento que se encuentre en perfectas condiciones de funcionamiento las cuales se describirán paso a paso a continuación.

2.2.1.1 Comprobación de Motores

Para la prueba de los motores se realiza guiándose en la (Figura 38) donde se encuentran el diagrama de pines. Usaremos los siguientes pasos:

- Obtener aproximadamente 12VDC de una fuente de voltaje (Figura 50).
- Conectar +V de la fuente al terminal 1 del motor y GND al terminal 2 del mismo, el motor girará en sentido anti horario (Figura 51).
- Invertir la polaridad de la fuente, el motor deberá girar en sentido horario (Figura 52).
- Realizar el mismo procedimiento en el segundo motor.

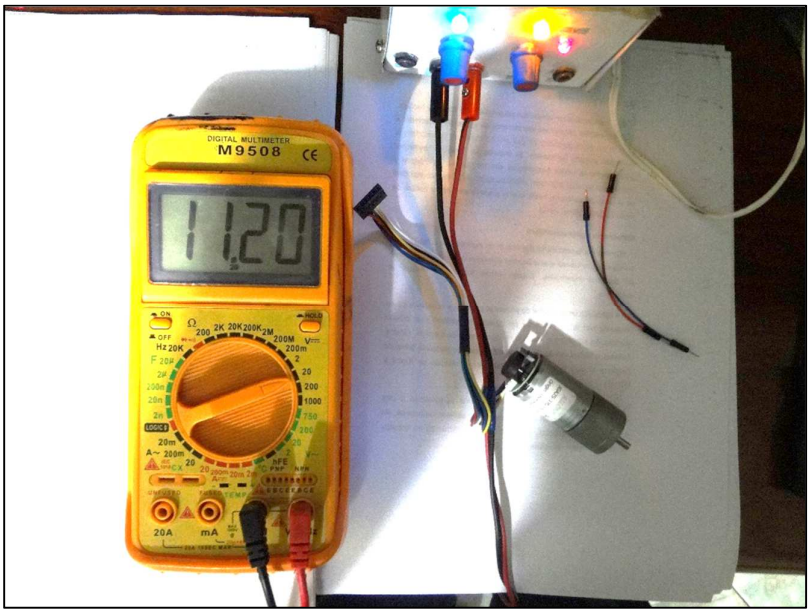

Figura 50 Voltaje de la fuente de alimentación

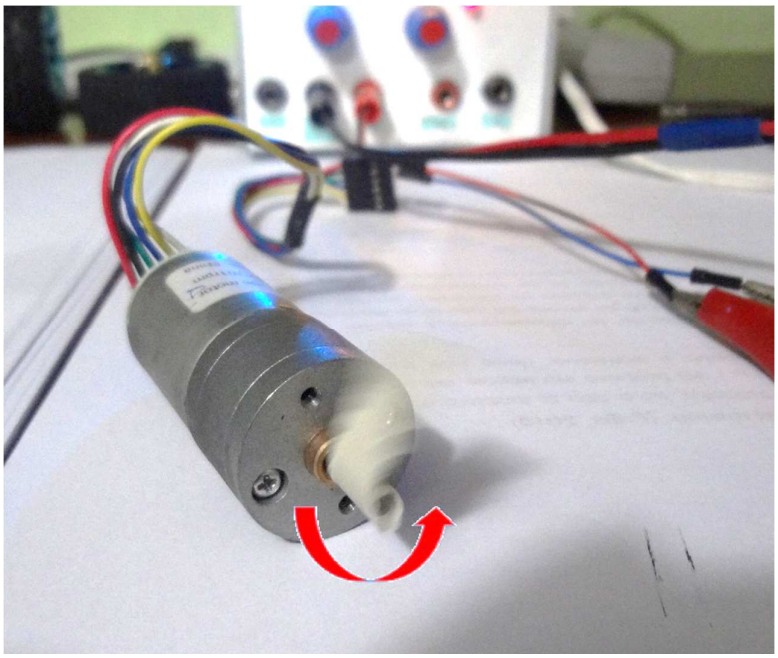

Figura 51 Giro del motor en sentido anti horario

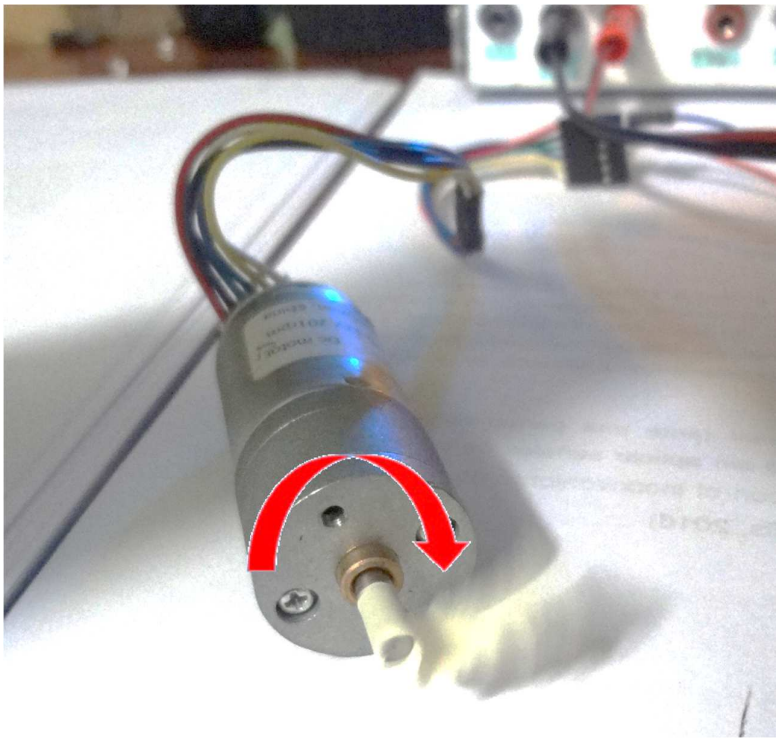

Figura 52 Giro del motor en sentido horario

2.2.1.2 Comprobación de Encoders

Para la prueba de los Encoders se debe guiar en el apartado 2.11 donde se encuentran el diagrama de pines. A continuación se debe realizar los pasos siguientes:

- Polarizar el Encoder con 5VDC, +V en el terminal 4 y GND en el terminal 3 (Figura 53).
- Seleccionar la posición de niveles lógicos en el multímetro y conectar en el terminal 5 (Figura 54).
- Girar el motor manualmente y observar el cambio de niveles lógicos (Figura 55).
- De forma similar conectar en el terminal 6 donde se observara el cambio de niveles lógicos.
- Realizar el mismo procedimiento en el segundo encoder.

Nota: Si el multímetro carece de comprobador de niveles lógicos se puede utilizar un osciloscopio o instrumentos similares.

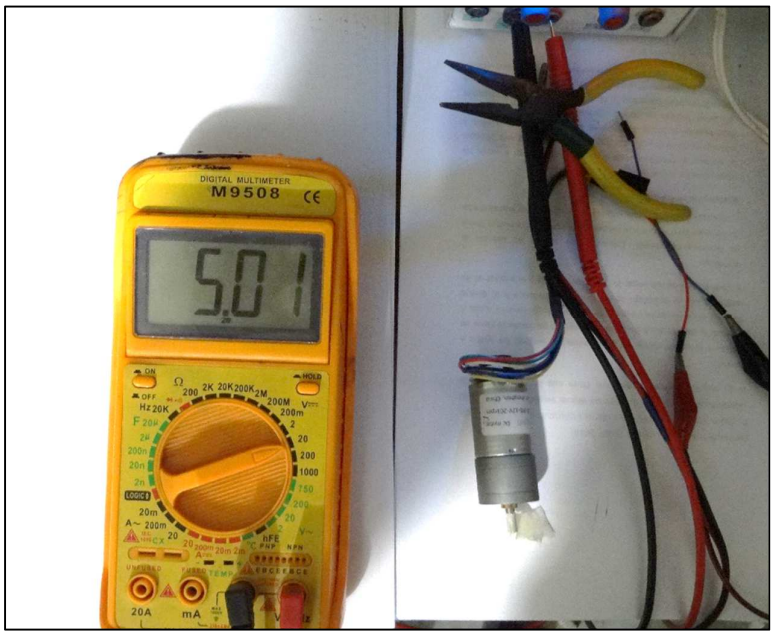

Figura 53 Voltaje de alimentación del Encoder

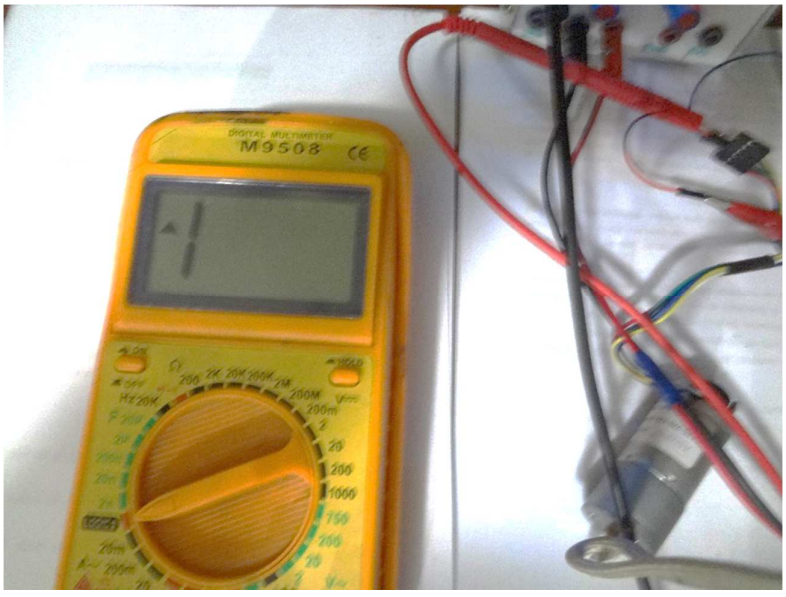

Figura 54 Comprobador de niveles lógicos en alto

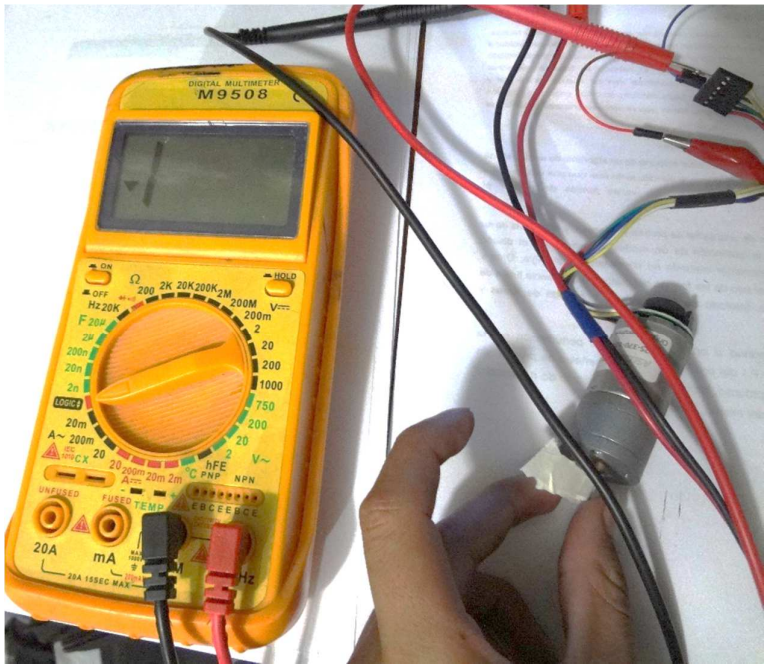

Figura 55 Comprobador de niveles lógicos en bajo

2.2.1.3 Comprobación de la Batería

Para comprobar el estado de la batería se debe medir voltaje entre sus terminales, el valor optimo seria 12VDC, si se encuentra con un valor de voltaje muy bajo se procede a cargar.

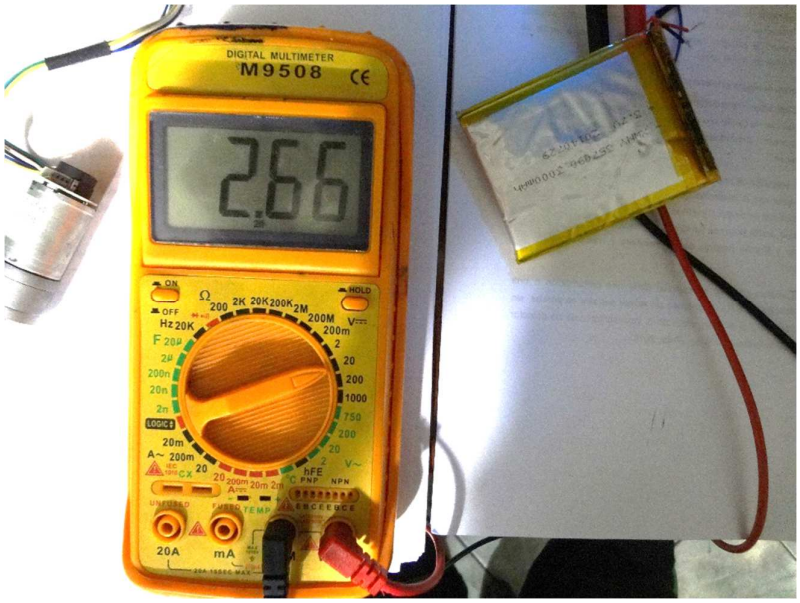

Figura 56 Batería descargada

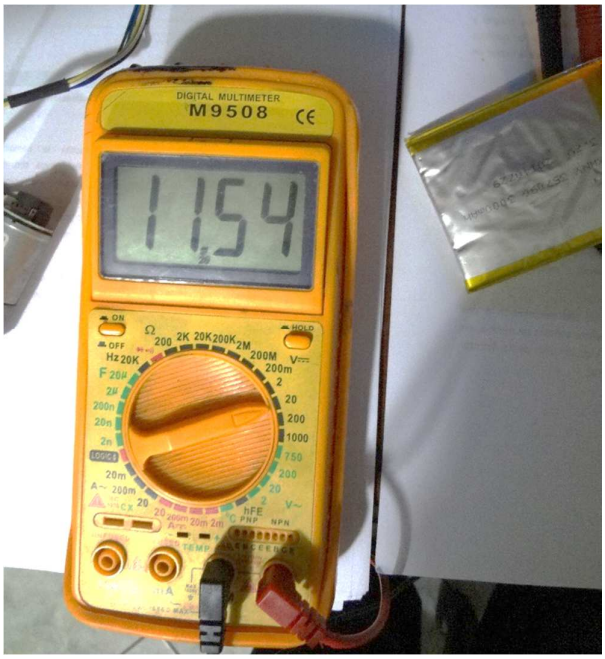

Figura 57 Batería cargada

2.2.1.4 Comprobación de Arduino Mega

Para saber el estado de la tarjeta Arduino Mega, basta con cargar un pequeño programa y comprobar su funcionamiento, para lo cual se realiza de la siguiente manera:

- Abrir un nuevo IDE en Arduino (Figura 58).
- Dirigirse a Archivo, Ejemplos, Basic, Blink, donde se abrirá un nuevo Sketch (Figura 59).
- Conectar Arduino Mega a la PC.
- Seleccionar la tarjeta y el puerto de comunicación, en este caso es el COM4 (Figura 60).
- Cargar el Sketch a Arduino Mega.
- Conectar un led al Pin 13 este deberá oscilar en un intervalo de 1s (Figura 61) y (Figura 62).

- Esta acción indica que la tarjeta Arduino Mega se encuentra en perfecto estado para cargar cualquier programa realizado.

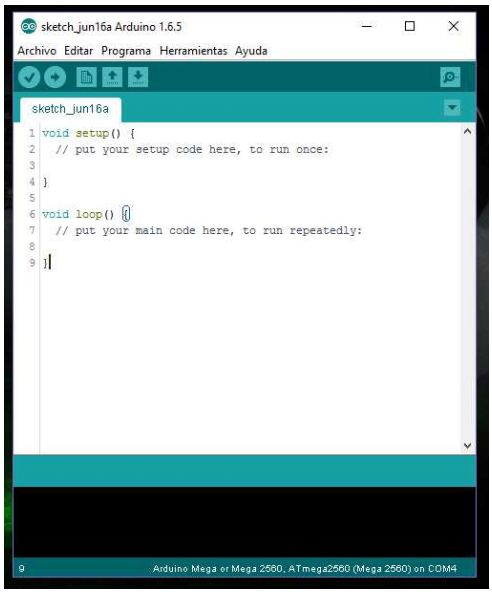

Figura 58 Nuevo IDE Arduino

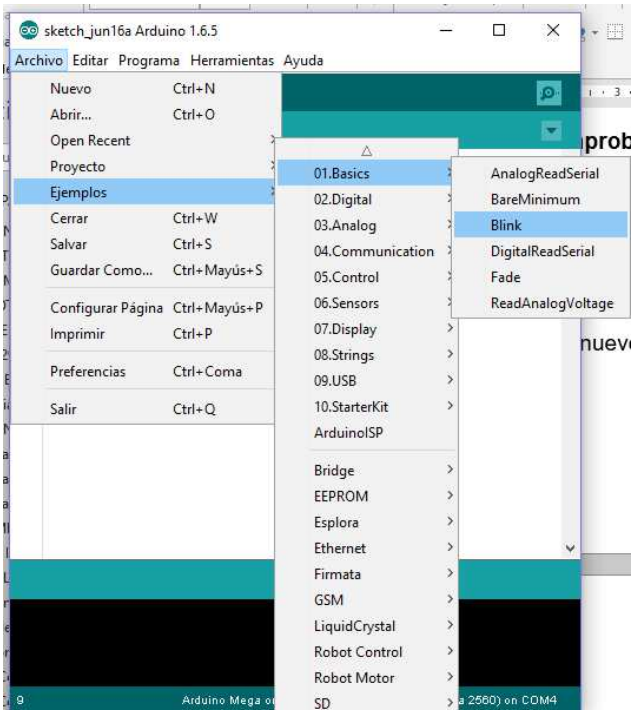

Figura 59 Ejemplo Blink

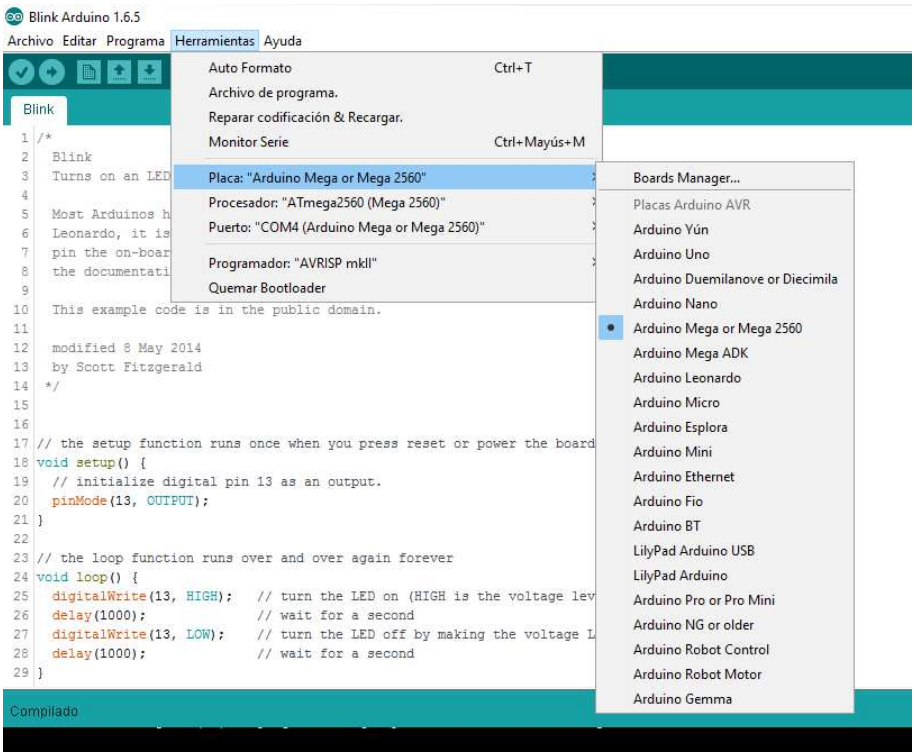

Figura 60 Selección de Placa y Puerto de comunicación

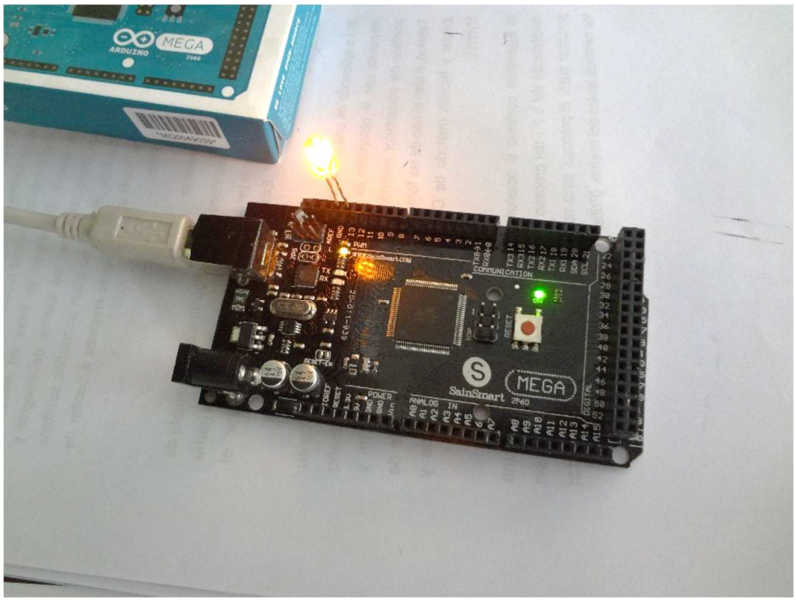

Figura 61 Ejemplo led encendido 1s

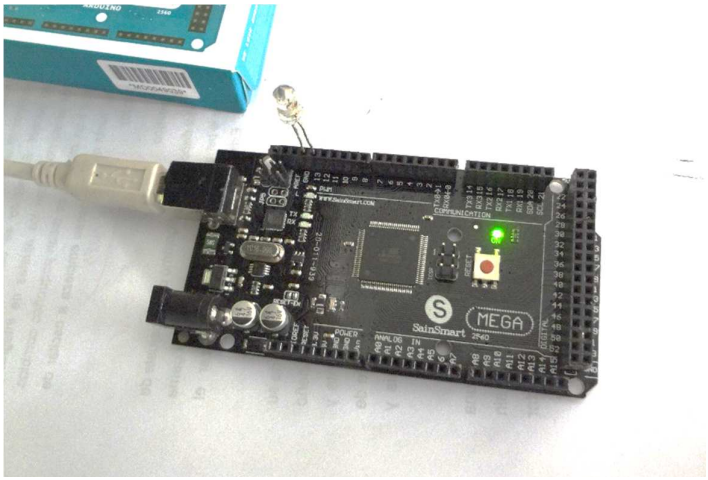

Figura 62 Ejemplo led apagado 1s

2.2.1.5 Comprobación de Shield SainSmart (Puente H)

La Shield SainSmart que se está utilizando es completa ya que posee en ella un integrado L298P, conectores directos para varios accesorios entre ellos también se puede conectar el sensor MPU6050, para comprobar el correcto funcionamiento del puente H se debe realizar los siguientes pasos:

- Energizar la Shield con 9VDC (Figura 63).
- Conectar el Pin 22 y 5 a VCC (+5V) y el Pin 23 a GND (Figura 64).
- En la Shield donde van ubicados los terminales del motor, colocar el multímetro donde debe dar un valor aproximado a los 9V (Figura 65).
- Conectar el Pin 23 y 5 a VCC y el Pin 22 a GND (Figura 66).
- El multímetro marcará un valor aproximado a los -9V, es decir cambiará su polaridad, lo que significa que el motor cambiará el sentido de giro una vez implementado (Figura 67).
- Conectar el Pin 5 a GND y sin importar cual sea la conexión de los pines 22 y 23 en el multímetro se visualizará 0V (Figura 68) y (Figura 69).

Nota: Para mejor comprensión del funcionamiento del puente H mediante los pines en la Shield ver la siguiente tabla (Tabla 10).

Tabla 10

Pines del puente H en la Shield

PIN SHIELD	NOMBRE	DESCRIPCIÓN
23	TN1	CONTROL MOTOR A-R
22	TN2	CONTROL MOTOR A-L
5	ENA	ENTRADA PWM A (500Hz)
24	TN3	CONTROL MOTOR B-R
25	TN4	CONTROL MOTOR B-L
4	ENB	ENTRADA PWM B (500Hz)

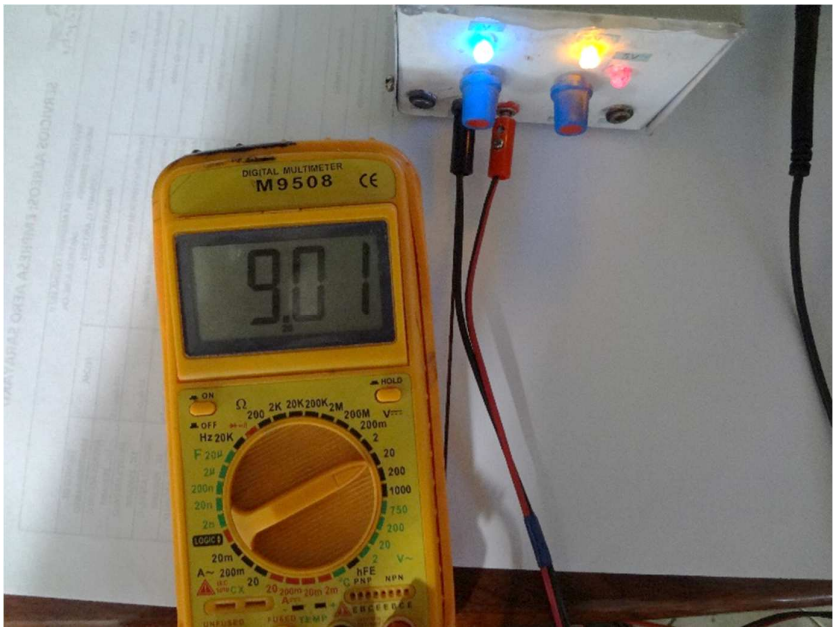

Figura 63 Voltaje de alimentación Shield

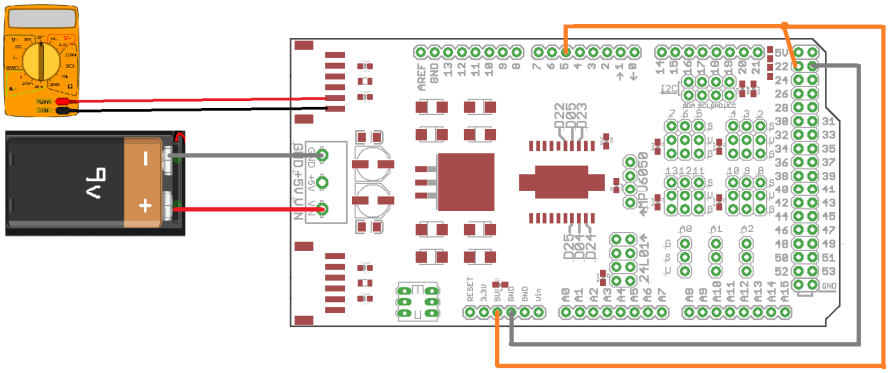

Figura 64 Conexión giro horario del motor A

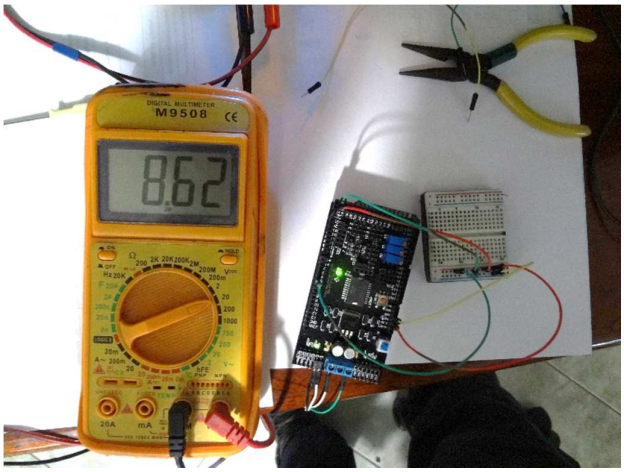

Figura 65 Voltaje de salida del puente H (Giro horario motor A)

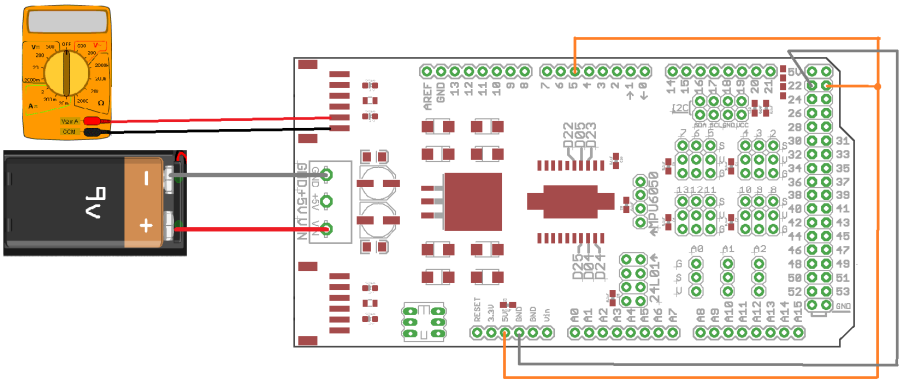

Figura 66 Conexión giro anti horario del motor A

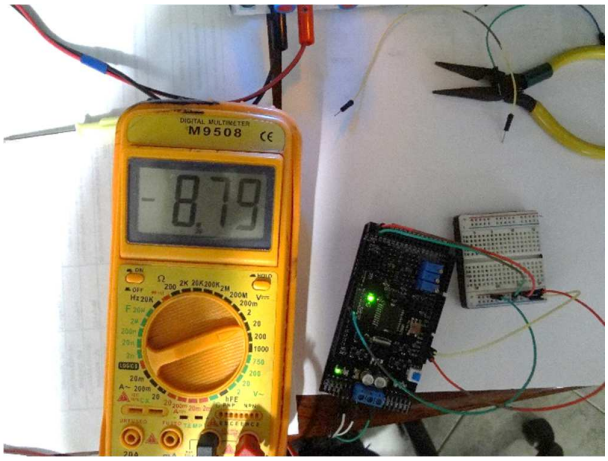

Figura 67 Voltaje de salida del puente H (Giro anti horario motor A)

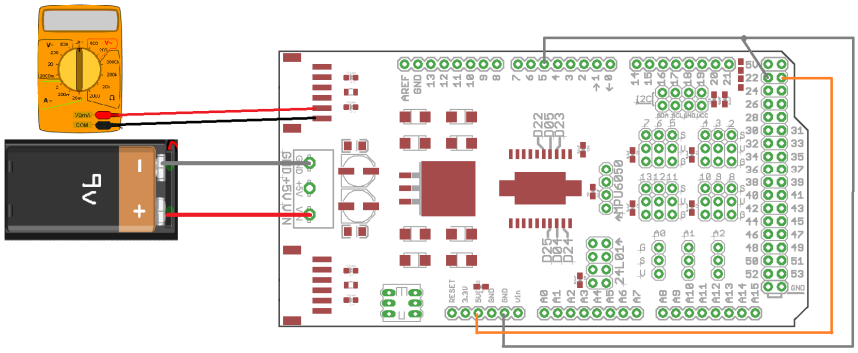

Figura 68 Conexión de parada del motor A

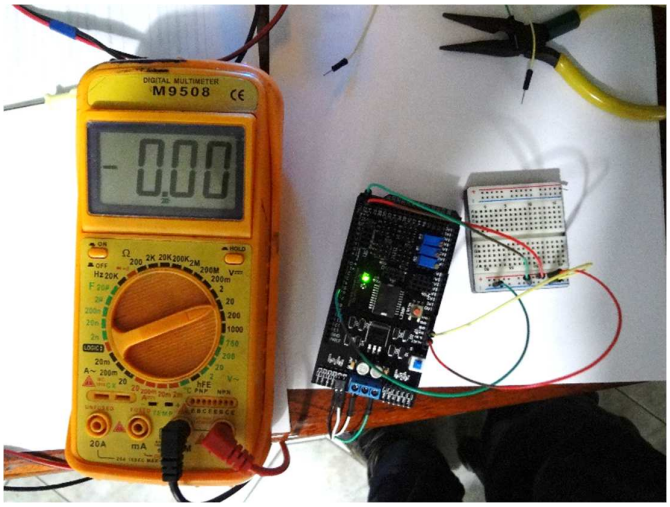

Figura 69 Voltaje de salida del puente H (Motor A parado)

De forma similar se comprueba el funcionamiento del siguiente puente H con sus respectivos pines de la siguiente manera:

- Energizar la Shield con 9VDC (Figura 63).
- Conectar el Pin 24 y 4 a VCC (+5V) y el Pin 25 a GND (Figura 70).
- De forma similar nos deberá marcar 9V en el multímetro (Figura 71).
- Conectar el Pin 25 y 4 a VCC y el Pin 24 a GND (Figura 72).
- El valor marcado en él, multímetro cambiara su polaridad (Figura 73).
- De forma similar en cualquier otra forma de conexión el valor deberá marcar 0V (Figura 74) y (Figura 75).

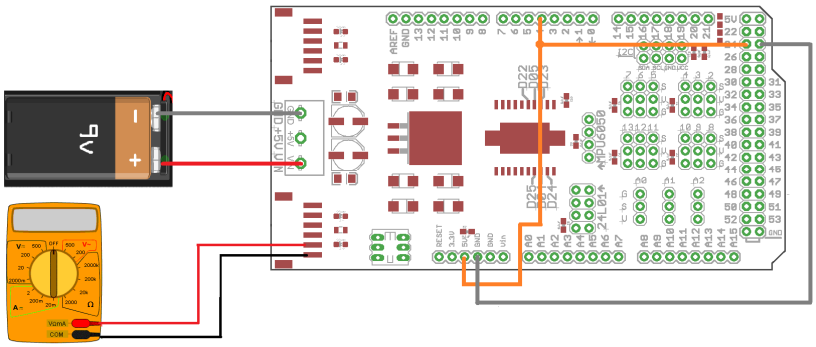

Figura 70 Conexión giro horario del motor B

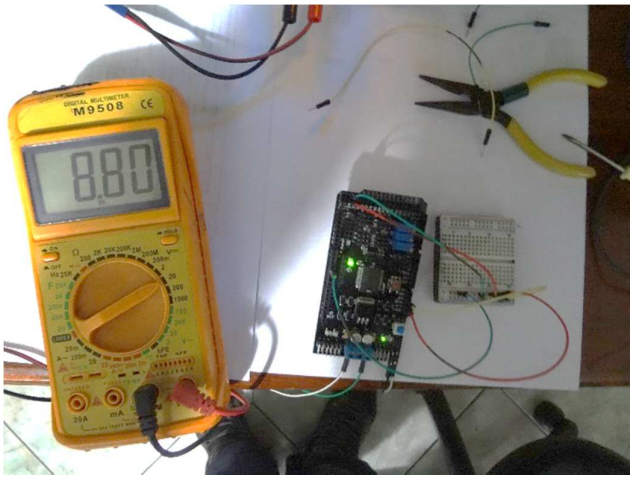

Figura 71 Voltaje de salida del puente H (Giro horario motor B)

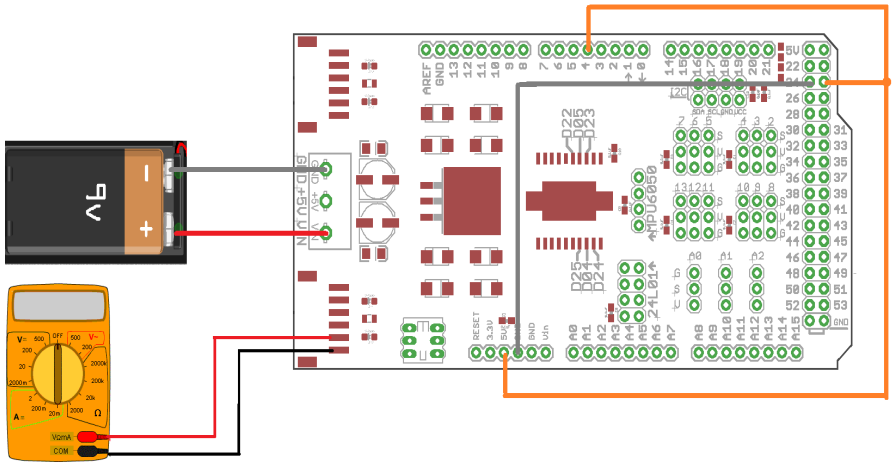

Figura 72 Conexión giro anti horario del motor B

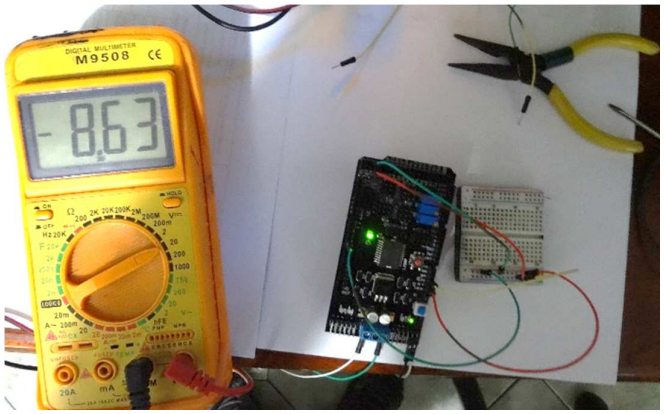

Figura 73 Voltaje de salida del puente H (Giro anti horario motor B)

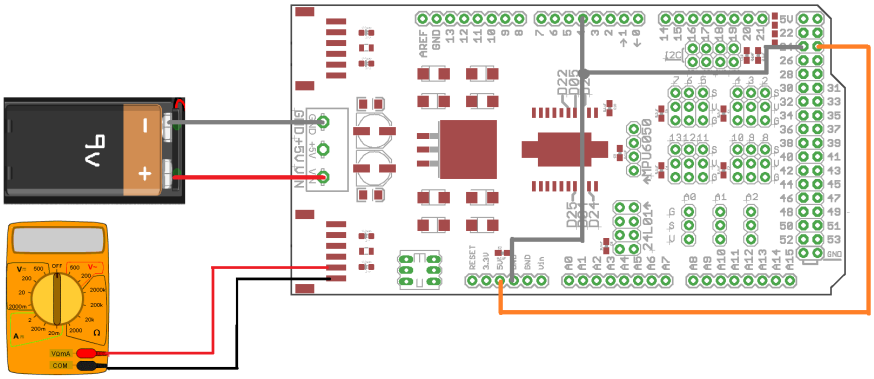

Figura 74 Conexión de parada del motor B

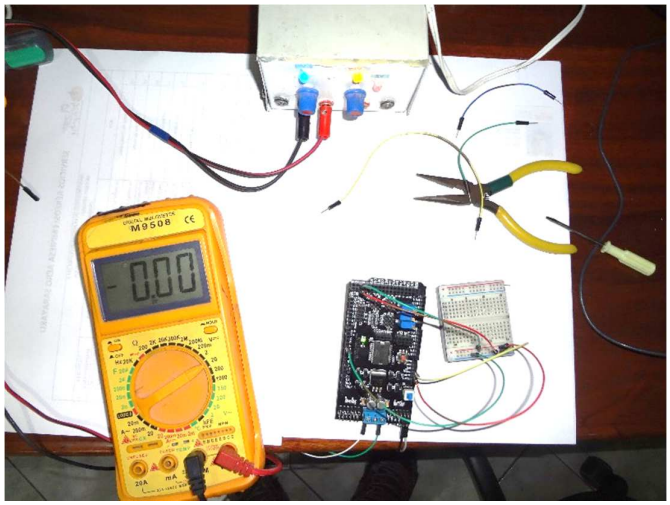

Figura 75 Voltaje de salida del puente H (Motor B parado)

2.2.1.6 Comprobación de Shield MPU6050

Para poder comprobar el correcto funcionamiento de la Shield y con ayuda del puerto serie se observa los datos enviados del giróscopo y acelerómetro, se debe realizar los siguientes pasos:

- Conectar de la siguiente forma la Shield a Arduino (Figura 76).
- Abrir un nuevo IDE y dirigirse a Archivo, Ejemplos, MPU6050, Examples, MPU6050_raw (Figura 77).
- Se abrirá el programa llamado MPU6050_raw, y cargar el programa a Arduino (Figura 78).
- Iniciar el Monitor Serie en donde los valores deberán oscilar cuando la Shield sea movida en sus ejes (Figura 79).

Nota: Para visualizar los valores obtenidos por la Shield el Monitor Serie debe ubicarse a una velocidad de 38400 Baudios (Figura 80).

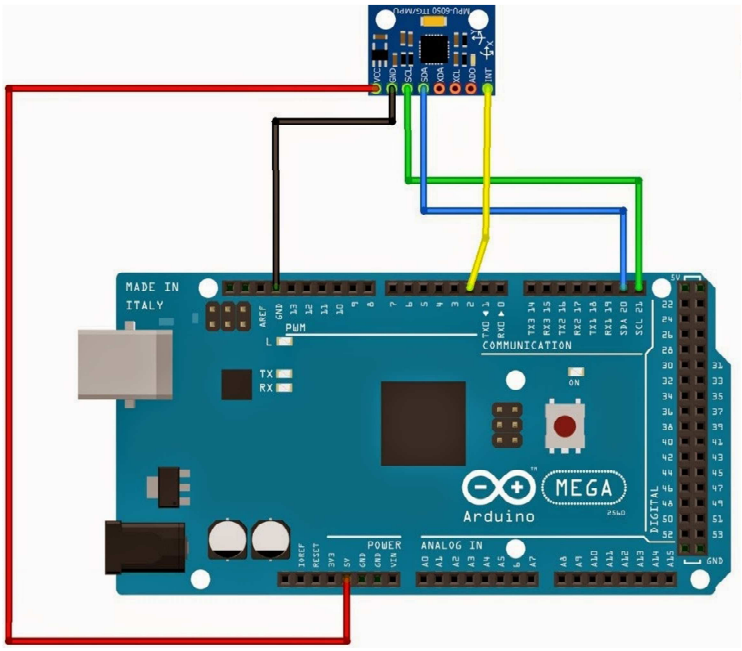

Figura 76 Conexión del MPU6050 al Arduino Mega

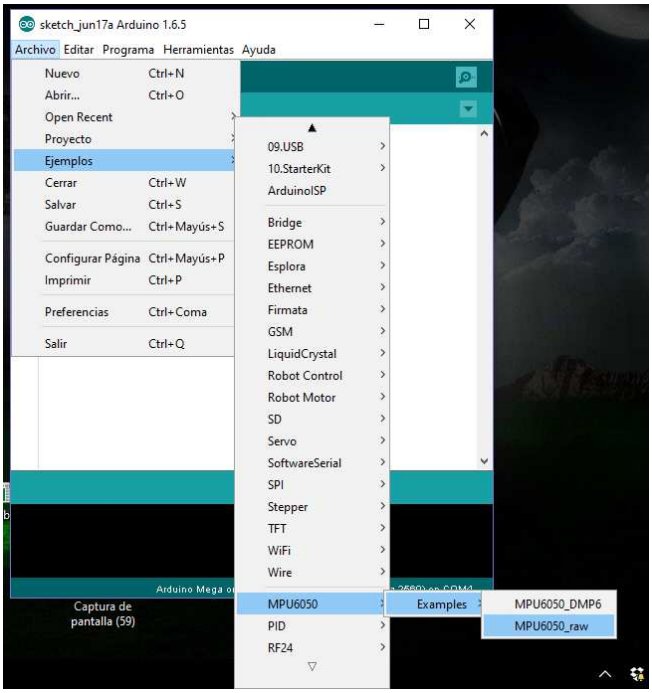

Figura 77 Ejemplo de la utilización del MPU6050

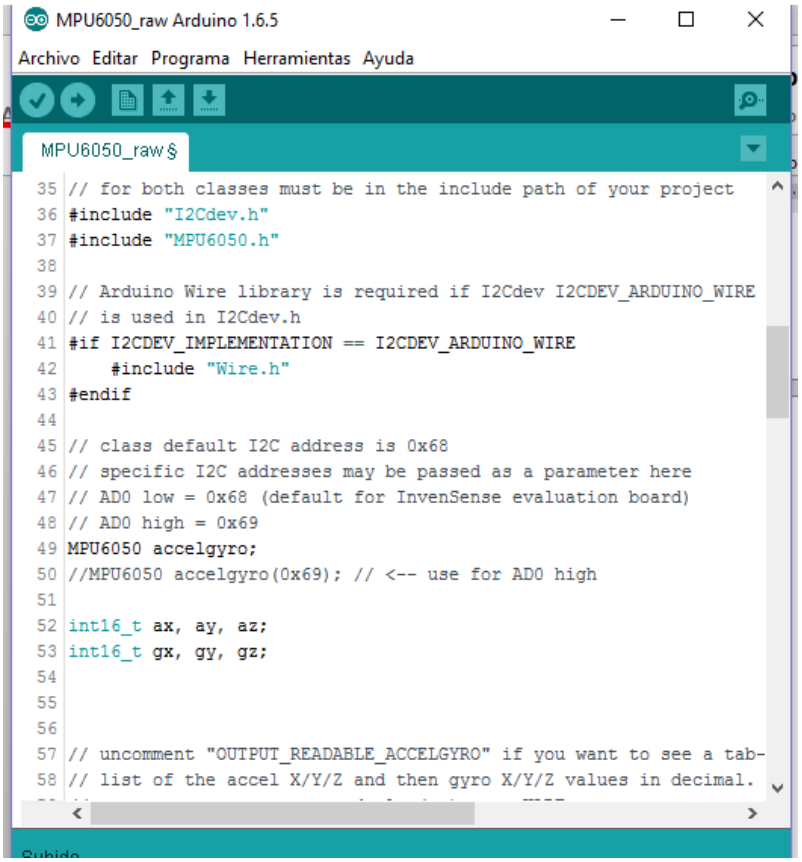

Figura 78 Programa Mpu6050_raw

Figura 79 Icono del Monitor Serie

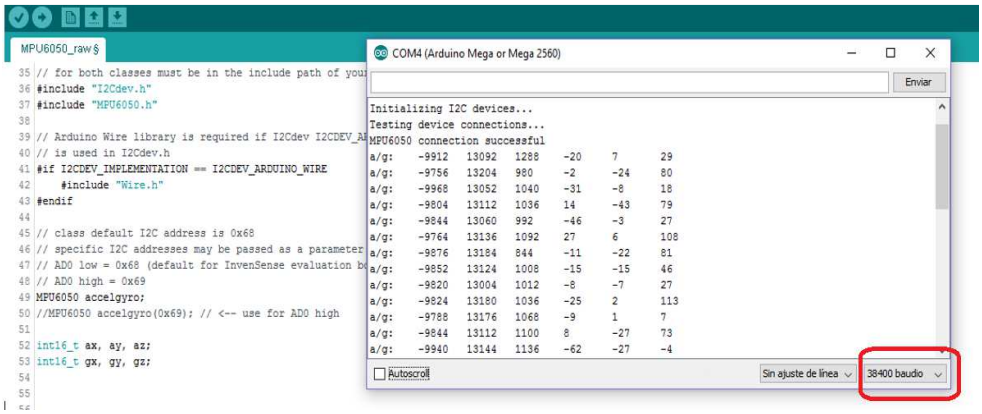

Figura 80 Ubicación de la velocidad en Baudios

3.2.2 Ensamblaje de la estructura física del vehículo

Para el ensamblaje del vehículo se deberá tener previamente todas las partes estructurales y las partes electrónicas en perfecto estado.

- Colocar los soportes de los motores en la base del vehículo como se muestra (Figura 81).

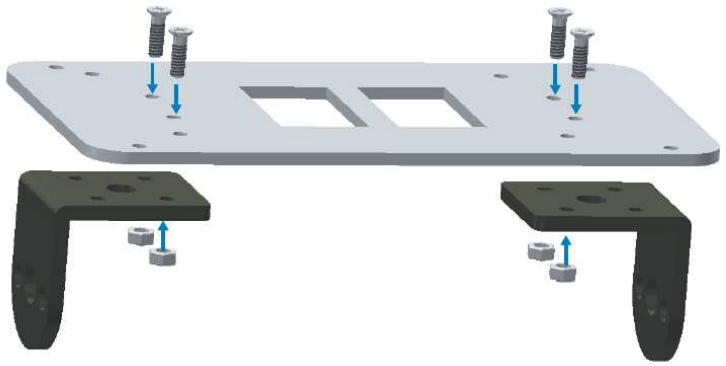

Figura 81 Instalación de los soportes de los motores

- Fijar los dos motores a los soportes, sostenidos por los dos tornillos (Figura 82).

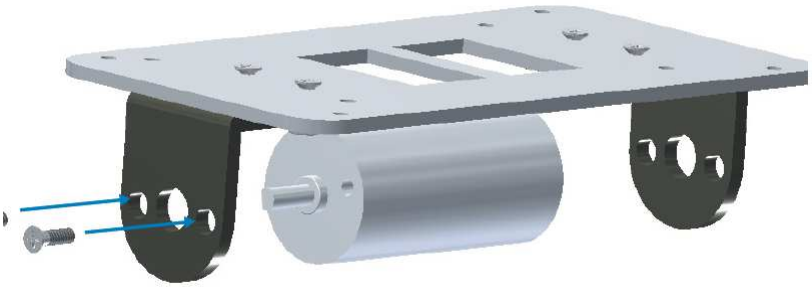

Figura 82 Fijación del motor al soporte

- Colocar el acople de las ruedas en el rotor del motor (Figura 83).

Figura 83 Instalación del acople

- Instalar ambas ruedas en los motores (Figura 84).

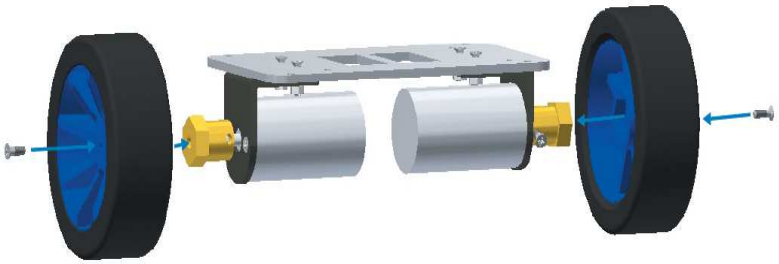

Figura 84 Instalación de las ruedas

- Colocar los soportes para la tarjeta Arduino (Figura 85).

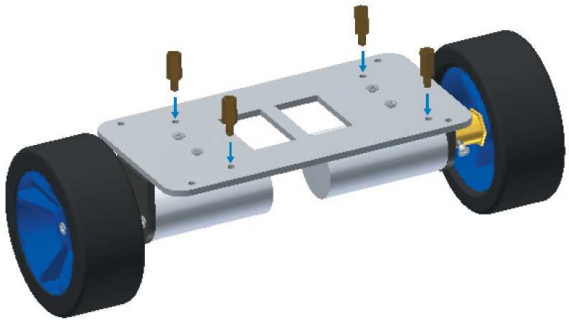

Figura 85 Instalación de los soportes para la tarjeta

- Ajustar los soportes mediantes las tuercas por debajo de la base (Figura 86).

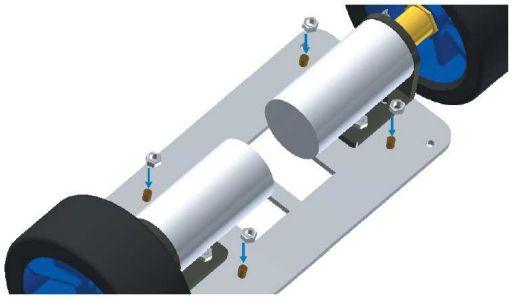

Figura 86 Fijación de soportes

- Colocar la tarjeta Arduino sobre los soportes y asegurar con tornillos (Figura 87).

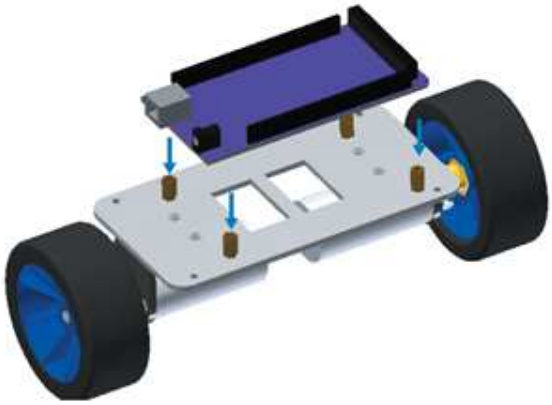

Figura 87 Instalación de la tarjeta

- Sobre la tarjeta Arduino colocar la Shield para control de los motores (Figura 88).

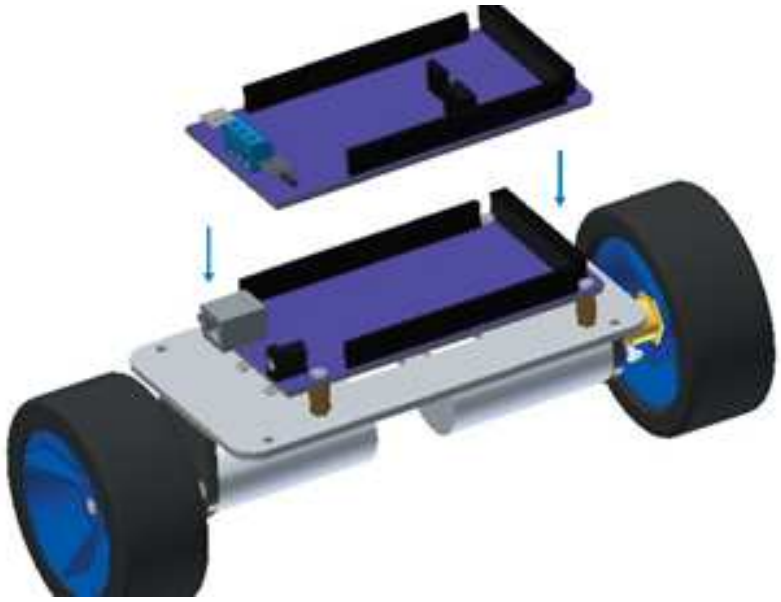

Figura 88 Colocación de la Shield

- El vehículo deberá quedar de forma similar a la (Figura 89).

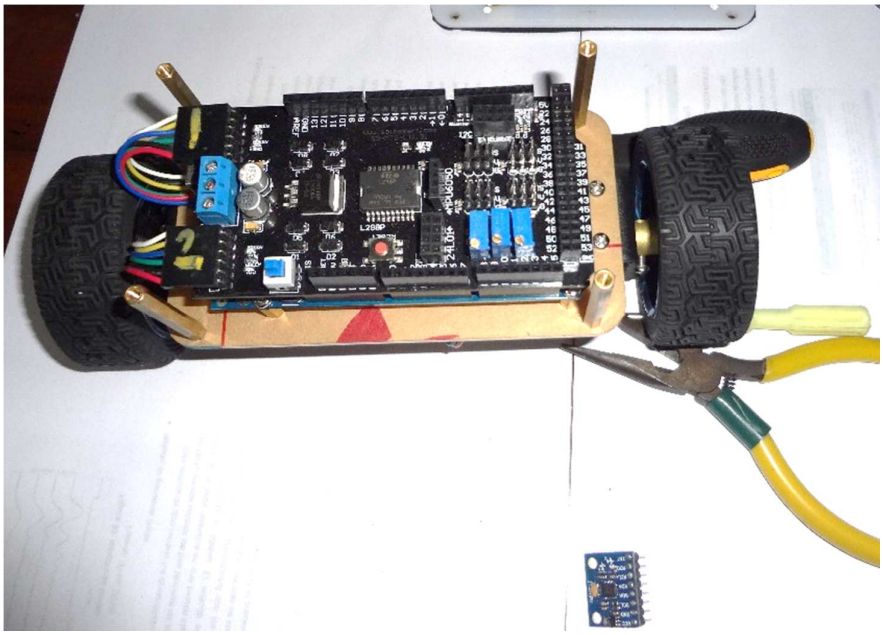

Figura 89 Vehículo semi-armado

- Instalar el MPU6050 sobre la Shield de la siguiente forma (Figura 90).

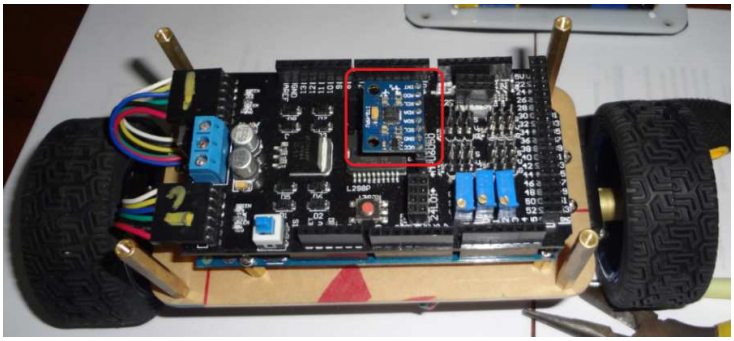

Figura 90 Colocación del MPU6050

- Colocar los terminales de la fuente en las borneras respectivas, en este caso se utiliza una batería como fuente (Figura 91).

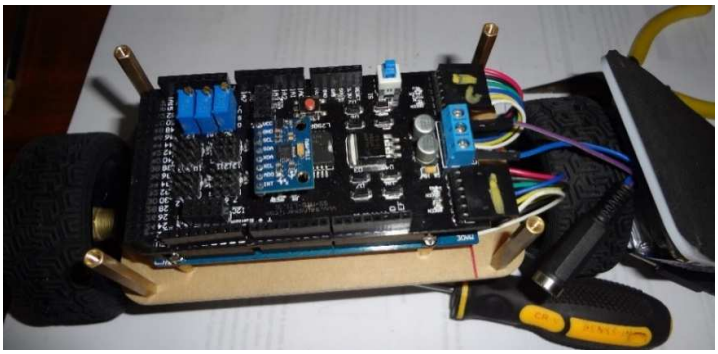

Figura 91 Alimentación del vehículo

- El vehículo ensamblado completamente quedara de esta manera (Figura 92).

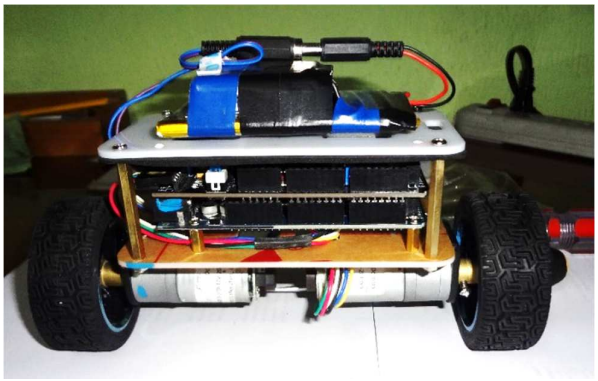

Figura 92 Vehículo completamente armado

3.3 Diseño del Sketch

3.3.1 Programación en la zona global

- Se declara las librerías a usar, en caso de dar un error en la compilación, tomar en cuenta que las librerías se encuentren instaladas (Figura 93).

- Crear el objeto ***MPU accelgyro;*** el cual será utilizado luego, y se define los valores por default, estos pueden ser cambiados para realizar pruebas (Figura 94).

- Declaramos todas las variables a utilizar (Figura 95).

```
#include <Wire.h> //LIBRERIA QUE PERMITE LA COMUNICACION I2C
#include <I2Cdev.h> //LIBRERIA QUE PERMITE CONECCION DEL SENSOR A ARDUINO
#include <MPU6050.h> // LIBRERIA DEL SENSOR
```

Figura 93 Declaración de librerías

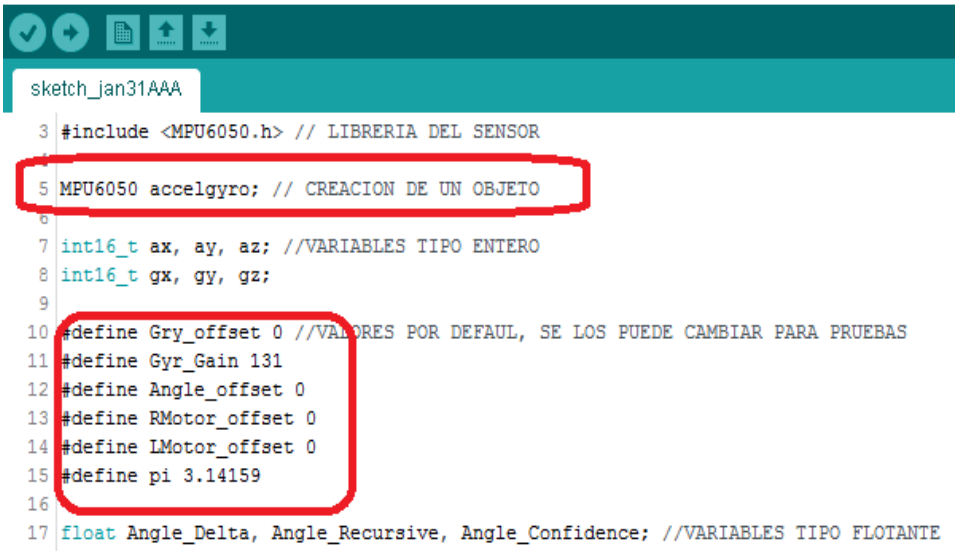

Figura 94 Objeto y valores por default

sketch_jan31AAA

```
int16_t ax, ay, az; //VARIABLES TIPO ENTERO
int16_t gx, gy, gz;

#define Gry_offset 0 //VALORES POR DEFAUL, SE LOS PUEDE CAMBIAR PARA PRUEBAS
#define Gyr_Gain 131
#define Angle_offset 0
#define RMotor_offset 0
#define LMotor_offset 0
#define pi 3.14159

float Angle_Delta, Angle_Recursive, Angle_Confidence; //VARIABLES TIPO FLOTANTE

float kp, ki, kd;
float Angle_Raw, Angle_Filtered, omega, dt;
float Turn_Speed = 0, Run_Speed = 0;
float LOutput, ROutput, Input, Output;

unsigned long preTime, lastTime; // VARIABLES LARGAS SIN SIGNO
float errSum, dErr, error, lastErr;
int timeChange;

float Sum_Right, Sum_Right_Temp, Sum_Left, Sum_Left_Temp, Distance, Distance_Right, Distance_Left, Speed;

int TN1 = 23; // SALIDA DIGITAL A,R
int TN2 = 22; // SALIDA DIGITAL A,L
int ENA = 5;  // SALIDA PWM A
int TN3 = 24; // SALIDA DIGITAL B,R
int TN4 = 25; // SALIDA DIGITAL B,L
int ENB = 4;  // SALIDA PWM B
```

Figura 95 Declaración de variables

3.3.2 Programación dentro de void setup()

- Se inicia la comunicación ***Wire.begin();*** y se declara en que pines se tendrá los pulsos PWM, en nuestro caso los Pines 4 y 5 de Arduino, el cual producirá la velocidad de los motores (Figura 96).

Nota: Estos comandos son diferentes a los de Arduino ya que se está programando directamente desde el ATMEGA.

```
void setup()
{
   Wire.begin(); // INICIA COMUNICACION WIRE


  TCCR3A = _BV(COM3A1) | _BV(WGM31) | _BV(WGM30);// GENERA SEÑAL PWM EN EL TIMER 3, PIN5 ARDUINO

  TCCR3B = _BV(CS31); //CONFIGURACION TIMER 3 PARA GENERAR PWM


  TCCR0A = _BV(COM0B1) | _BV(WGM01) | _BV(WGM00); // GENERA SEÑAL PWM EN EL TIMER 0, PIN4 ARDUINO

  TCCR0B = _BV(CS01) | _BV(CS00);// PRESCALER, ES UNA FUENTE DE RELOJ QUE TIENE TODOS LOS CONTADORES
```

Figura 96 Configuración de pines PWM

- Se inicia el Acelerómetro y se toma 200 muestras de la función en 0.6875 uS ***Filter()*** (Figura 97)**.**

```
109
110 void Filter()
111 {
112
113   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // OBTIENE DATOS DEL GUIROSCOPO Y ACELEROMETRO EN LOS EJES X,Y,Z.
114   Angle_Raw = (atan2(ay, az) * 180 / pi + Angle_offset); // TRANSFORMA DE RADIANES A GRADOS
115   omega = gx / Gyr_Gain + Gry_offset; // VALOR DEL GUIROSCOPO
116 // FILTROS YA CREADOS PARA TENER UN ANGULO MAS EXACTO
117   unsigned long now = micros(); // TIEMPO ACTUAL
118   timeChange = now - preTime;
119   preTime = now;
120   dt = timeChange * 0.000001; // TRANSFORMACION A SEGUNDOS
121   Angle_Delta = (Angle_Raw - Angle_Filtered) * 0.64;
122   Angle_Recursive = Angle_Delta * dt + Angle_Recursive;
123   Angle_Confidence = Angle_Recursive + (Angle_Raw - Angle_Filtered) * 1.6 + omega;
124   Angle_Filtered = Angle_Confidence * dt + Angle_Filtered;
125   }
```

Figura 97 Función Filter()

- La función ***Filter()*** permite saber el ángulo en grados y corregir pequeños errores de medición generados por el MPU6050.
- Se toma el valor absoluto del ángulo siempre y cuando sea menor a 45 grados y dirige a ***Filter()*** (Figura 97) *y* ***myPID()*** (Figura 99)**.**
- Caso contrario nos define los pines de entrada y salida e interrupciones (Figura 98).

Nota: Los parámetros PID fueron tomados aleatoriamente para pruebas de funcionamiento.

```
60   if (abs(Angle_Filtered) < 45)
61   {
62     omega = Angle_Raw = Angle_Filtered = 0;
63     Output = error = errSum = dErr = 0;
64     Filter();
65     myPID();
66   }
67   pinMode(IN1, OUTPUT);
68   pinMode(IN2, OUTPUT);
69   pinMode(IN3, OUTPUT);
70   pinMode(IN4, OUTPUT);
71   pinMode(ENA, OUTPUT);
72   pinMode(ENB, OUTPUT);
73   pinMode(18, INPUT);
74   pinMode(2, INPUT);
75
76   attachInterrupt(4, State_A, FALLING); // ACTIVA INTERRUPCIONES (para saber q encoder se activo primero)
77   attachInterrupt(1, State_B, FALLING);
```

Figura 98 Declaración de pines e interrupciones

```
void myPID()
{
kp = 22.000; //CONSTANTE PROPORCIONAL
ki = 1;     //CONSTANTE INTEGRAL
kd = 1.6;  //CONSTANTE DERIVATIVA

  // CALCULAR LOS VALORES DE SALIDA DL PID
 error = Angle_Filtered;
 errSum += error; //errSum =errSum + error INCREMENTO DEL ERROR (INTEGRAL ES LA SUMA DE LOS ERRORES)
 dErr = error - lastErr; // ERROR MENOS ERROR ANTERIOR
 Output = kp * error + ki * errSum + kd * omega; //FORMULA DEL PID
 lastErr = error;
 noInterrupts(); //DESACTIVA INTERRUPCIONES
 Sum_Right = (Sum_Right + Sum_Right_Temp) / 2; //PROMEDIO
 Sum_Left = (Sum_Left + Sum_Left_Temp) / 2;
 Speed = (Sum_Right + Sum_Left) / 2; //DETERMINA LA VELOCIDAD EN BASE AL MOVIMIENTO DER Y IQ
 Distance += Speed + Run_Speed;
 Distance = constrain(Distance, -300, 300);//LIMITA L DISTAACIOA DE 300 A -300
 Output += Speed * 70 + Distance * 0.6;// VELOCIDAD DE GUIRO
 Sum_Right_Temp = Sum_Right;
 Sum_Left_Temp = Sum_Right;
 Sum_Right = 0;
 Sum_Left = 0;
 ROutput = Output + Turn_Speed; //VELOCIDAD DE GUIRO R
 LOutput = Output - Turn_Speed; //VELOCIDAD DE GUIRO L
 interrupts(); //ACTIVAR INTERRUPCIONES
}
```

Figura 99 Función myPID

```
void State_A()
{
  if (digitalRead(18))// LEE PIN 18 ENTRADA DE SEÑAL ENCODER A
  {
    Sum_Right ++;
  }
  else
  {
    Sum_Right --;
  }
}

void State_B()
{
  if (!digitalRead(2))// LEE PIN 2 ENTRADA DE SEÑAL ENCODER B (! INVERSION )
  {
    Sum_Left ++;
  }
  else
  {
    Sum_Left --;
  }
}
```

Figura 100 Función State_A() y State_B()

3.3.3 Programación dentro de void loop()

- El ***void loop()*** dirige nuevamente a ***filter()*** para obtener los datos en grados de la inclinación, cada instante va tomando valores del tiempo transcurrido (Figura 101).

- Compara el valor obtenido del ángulo si es menor a 45 grados y ejecuta la función ***myPID()*** (Figura 99) *y* ***PWMControl()*** (Figura 102)**.**

- La función ***PWMControl()*** es la encargada del giro y velocidad de los motores.

```
void loop()

{
  while (1)
  {

    Filter();
    if ((micros() - lastTime) > 10000)// CADA 10ms TOMA LOS VALORES
    {

      if (abs(Angle_Filtered) < 45)// ACTUA SIEMPRE QUE EL ANGULO SEA MENOR A 45 GRADOS
      {
        myPID();
        PWMControl();
      }
      else //caso contrario parar motores
      {
        digitalWrite(TN1, HIGH);
        digitalWrite(TN2, HIGH);
        digitalWrite(TN3, HIGH);
        digitalWrite(TN4, HIGH);
      }
      lastTime = micros();
    }
  }
}
```

Figura 101 Función void loop()

```
sketch_jan31AAA §

void PWMControl()
{
  if (LOutput > 0)
  {
    digitalWrite(TN1, HIGH);//ACTIVA MOTOR A R
    digitalWrite(TN2, LOW);
  }
  else if (LOutput < 0)
  {
    digitalWrite(TN1, LOW); //ACTIVA MOTOR A L
    digitalWrite(TN2, HIGH);
  }
  else
  {
    OCR3A = 0;//DESACTIVA CONTROL PWM TIMER 3
  }
  if (ROutput   0)
  {
    digitalWrite(TN3, HIGH); //ACTIVA MOTOR B R
    digitalWrite(TN4, LOW);
  }
  else if (ROutput < 0)
  {
    digitalWrite(TN3, LOW); //ACTIVA MOTOR B L
    digitalWrite(TN4, HIGH);
  }
  else
  {
    OCR0B = 0;//DESACTIVA CONTROL PWM TIMER 0
  }
  OCR3A = min(1023, (abs(LOutput * 4) + LMotor_offset * 4)); //MANDA SEÑAL PWM POR EL PIN5 MULTIMPLICA POR 4 POR LA RESOLUCION
  OCR0B = min(255, (abs(ROutput) + RMotor_offset)); // MANDA SEÑAL PWM POR PIN 4
}
```

Figura 102 Función PWMControl()

3.4 Compilación y carga del programa

- Se compila y verifica que no existan errores en el Sketch (Figura 103).

- Se procede a cargar el programa tomando en cuenta que tarjeta se utiliza y el puerto COM.

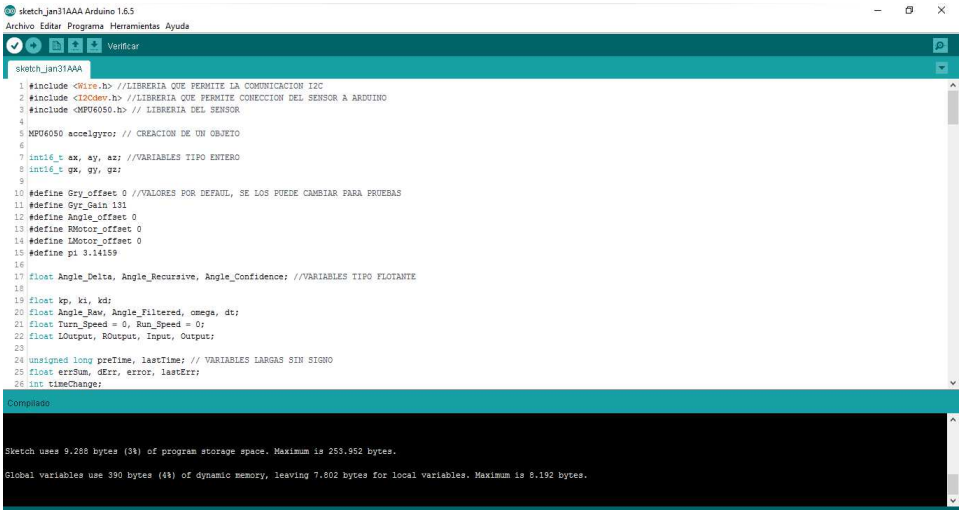

Figura 103 Compilación del programa

3.5 Pruebas de funcionamiento

En las pruebas de funcionamiento el vehículo empieza a oscilar de lado a lado (Figura 104) y no se mantiene estable, su oscilación va en aumento lo que produce que pierda el equilibrio y caiga (Figura 105).

Para lo cual se debe cambiar los parámetros PID hasta que el vehículo se mantenga lo más estable posible.

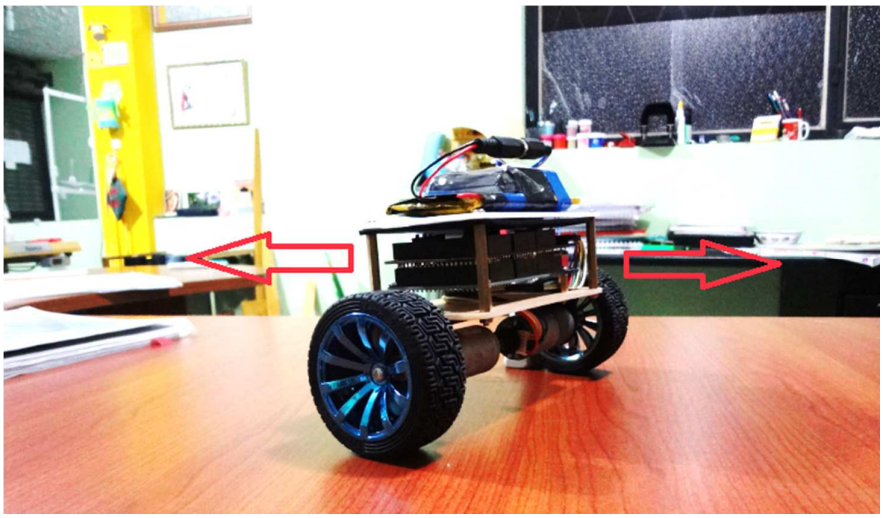

Figura 104 Oscilación del vehículo

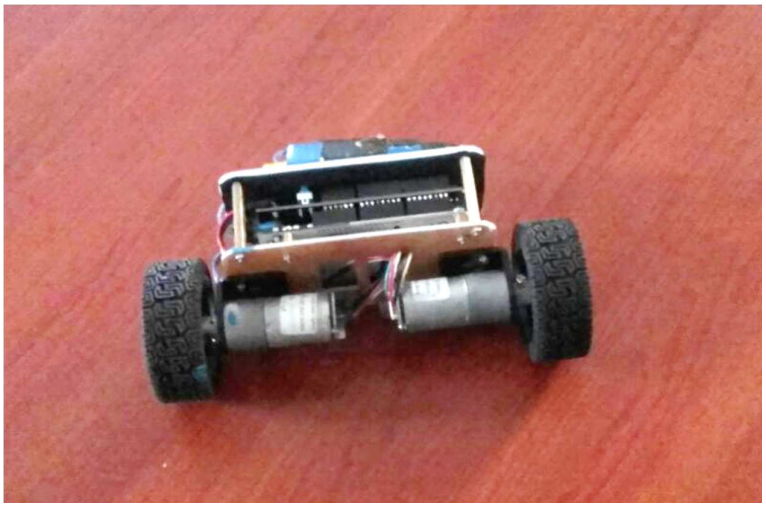

Figura 105 Vehículo caído

3.6 Sintonización PID

Para poder sintonizarlo se utilizara el método de ganancia límite y así conseguir una estabilidad en el vehículo, variando los valores PID.

Para facilitar esta tarea se deberá modificar el programa, haciendo uso de los potenciómetros integrados en la Shield y del monitor (Figura 108).

Los cambios realizados son los siguientes:

- Abrir el programa
- Declarar nuevas variables tipo flotantes, para lectura de los potenciómetros integrados en la Shield las cuales están conectadas a los puertos A0, A1, A2 (Figura 106).
- Las variables van a tener un rango de 0 a 30, estos serán los parámetros PID.
- Se inicia la comunicación Serial para poder visualizar los parámetros correctos en la PC.
- Poner los parámetros Proporcional, Integral y Derivativo en 0 (Figura 107) e ir variando cada uno hasta conseguir que el proceso se estabilice es decir que no existan tantas oscilaciones (Figura 109).

Figura 106 Lectura análoga A0, A1, A2

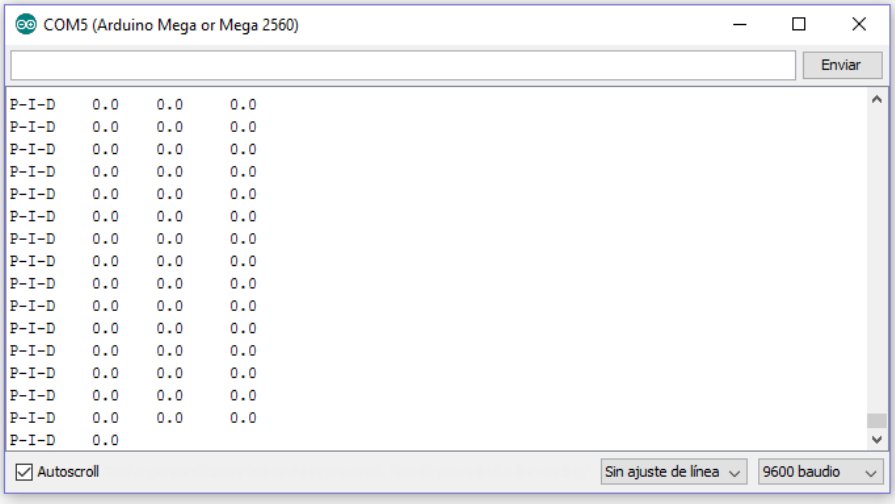

Figura 107 Comunicación serial

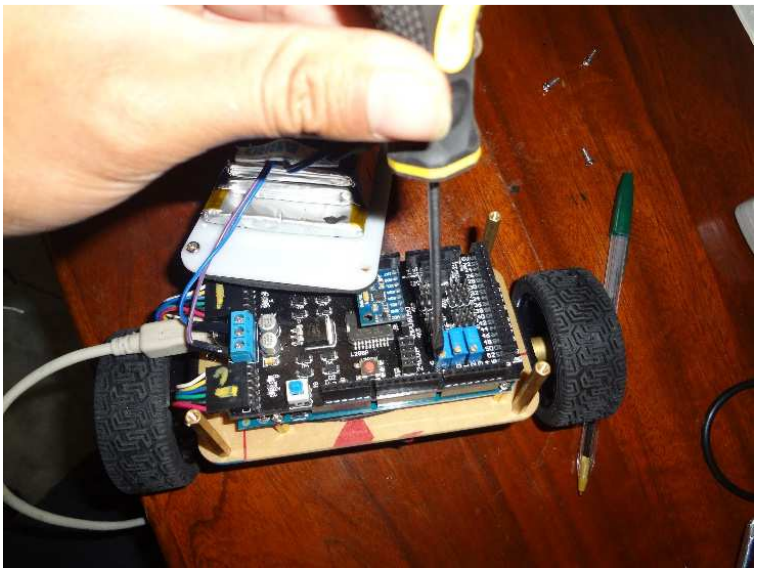

Figura 108 Variación de los potenciómetros

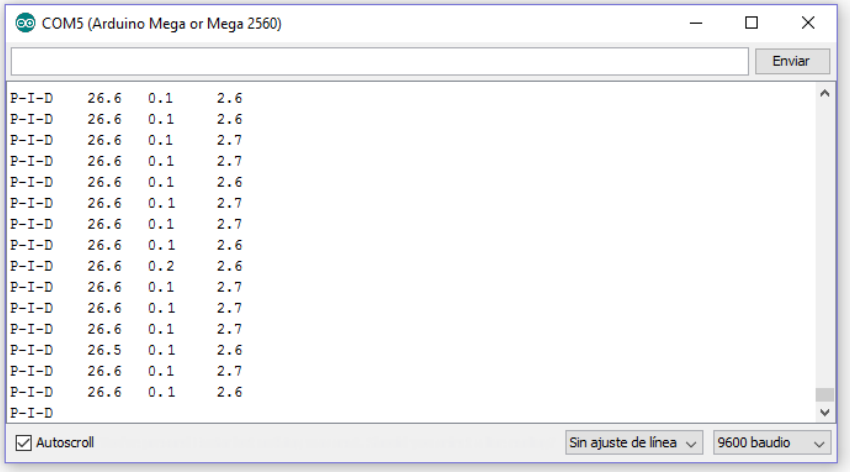

Figura 109 Parámetros PID encontrados

3.7 Prueba de funcionamiento final

En las pruebas finales se encontró que los parámetros correctos son kp=26, ki=0.1, kd=2,6 en donde el vehículo se encuentra lo más estable (Figura 110) y (Figura 111).

Los parámetros son sustituidos al final remplazando los valores en el programa original (Figura 112).

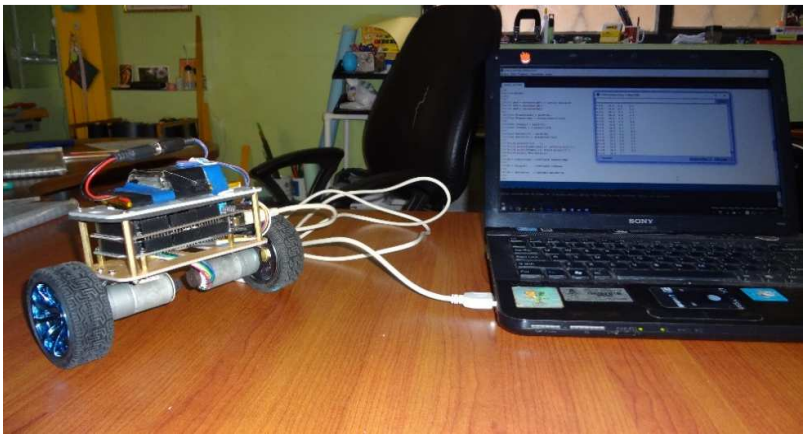

Figura 110 Pruebas de funcionamiento

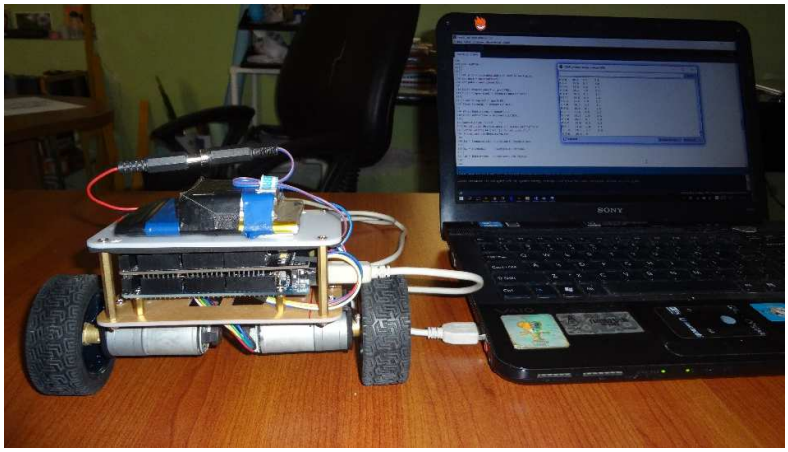

Figura 111 Pruebas de funcionamiento

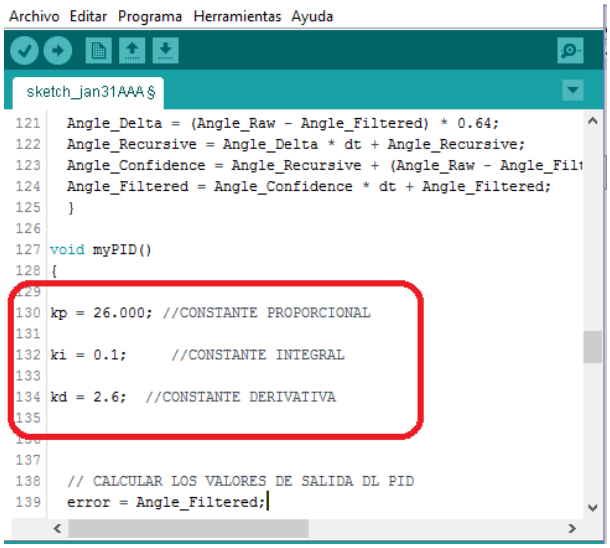

Figura 112 Valores PID en el programa original

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El entorno de Arduino es multiplataforma que permite la integración de diferentes partes y sensores siendo versátil en el desarrollo de controladores embebidos.

- Las librerías son de gran ayuda ya que nos sirven como punto de partida para el desarrollo de proyectos más sofisticados.

- El MPU 6050 proporciona datos obtenidos del movimiento en los tres ejes de los cuales solo se utilizan datos de dos ejes debido a los grados de libertad del vehículo.

- Arduino es compatible con una amplia gama de sensores gracias a la plataforma de código abierto ya que no solo maneja estándar propietario sino que también ocupa muchos otros estándares abiertos.

- La comunicación I2C permite la conectividad entre más dispositivos mediante solo la utilización de dos líneas de datos y una de tierra como es el caso del MPU6050.

- La aplicación del método de ganancia límite para sintonía del PID permite establecer los tres términos de ajuste del controlador a partir del procedimiento de calibración establecido.

- Una correcta utilización del centro de gravedad favorece a la implementación del algoritmo de control.

4.2 Recomendaciones

- Tener en cuenta los pines de polaridad de la Shield SainSmart ya que en caso de un corto circuito o mala polarización la tarjeta sufrirá daños irreparables al igual que Arduino Mega por el motivo que las dos se encuentran conectadas.

- Incluir las librerías necesarias a usar ya que Arduino no cuenta con algunas de ellas y son exclusivas de los fabricantes para lo cual se deberá descargarlas desde la página de los desarrolladores.

- Cambiar los parámetros de ganancia y offset del MPU6050 en la programación solo si presenta datos erróneos.

- Tomar en cuenta el tipo de comunicación que poseen los dispositivos externos a utilizar ya que Arduino cuenta con la mayoría de interfaces de comunicación.

- Revisar que exista una conexión de referencia a tierra entre los dispositivos que encuentren intercomunicando físicamente.

- Realizar procedimientos de calibración adecuados para sintonizar el PID ya sea por tanteo, ganancia limite o ajuste automático.

- Tener cuidado de la posición del centro de gravedad al ubicar los dispositivos utilizados y así evitar que el vehículo pierda estabilidad.

GLOSARIO DE TÉRMINOS

A

AC. Corriente Alterna.

ACTUADOR. Dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado.

ALEATORIO. Depende del azar o la casualidad.

ALGORITMO. Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problema.

ASÍNCRONO. No tiene un intervalo de tiempo constante entre cada evento.

AUTÓNOMO. Trabaja de forma independiente.

B

BUCLE. Sentencia que se realiza repetidas veces a un trozo aislado de código.

C

CC. Corriente Continua.

CLON. Sistema de computación basado en los diseños y desarrollos de otra compañía, fabricado para tener una compatibilidad del cien por ciento con el modelo original.

CODIFICADOR. Circuito combinacional con 2^N entradas y N salidas, cuya misión es presentar en la salida el código binario correspondiente a la entrada activada.

COLECTOR ABIERTO. Salida que se debe incluir una resistencia de carga externamente para que el circuito integrado nos proporcione un nivel alto. Esta resistencia se suele llamar “resistencia de Pull UP“.

CONTROL PID. Control Proporcional, Integral y Derivativo, mecanismo de control por realimentación ampliamente usado en sistemas de control industrial.

CPU. Unidad Central de Procesamiento.

E

EEPROM. Siglas de Electrically Erasable Programmable Read-Only Memory (ROM programable y borrable eléctricamente).

ENCAPSULADO. Resultado de la etapa final del proceso de fabricación de dispositivos con semiconductores.

ENCODERS. Codificador rotatorio, también llamado codificador del eje o generador de pulsos.

G

GANANCIA. Magnitud que expresa la relación entre la amplitud de una señal de salida respecto a la señal de entrada.

H

HZ. Unidad de medida de frecuencia.

I

I2C. Inter-Integrated Circuit, es un bus de datos serial desarrollado en 1982 por Philips Semiconductors.

IDE. Entorno de Desarrollo Integrado.

IMU. Unidad de Medida Inercial.

INTERFAZ. Conexión funcional entre dos sistemas o dispositivos de cualquier tipo dando una comunicación entre distintos niveles.

INTERRUPCIÓN. Señal recibida por el procesador para indicarle que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.

L

LIBRERÍAS. Conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

M

MEMS. Sistemas Microelectomecánicos**.**

N

NIVEL LÓGICO. Circuitos en los que solo existen dos estados posibles, alto y bajo.

O

OSCILADOR. Sistema capaz de crear perturbaciones o cambios periódicos en un medio.

P

POLARIDAD. Propiedad de los terminales (polos) de una batería o de una pila, que pueden ser positivos o negativos.

PWM. Modulación por ancho de pulsos (pulse-width modulation).

R

RESOLUCIÓN. Longitud de la palabra digital.

S

SENSOR. Objeto capaz de detectar magnitudes físicas o químicas.

SÍNCRONO. Sistema de comunicación, el transmisor debe coordinarse con el receptor antes del envío de datos.

U

UARTs. Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie.

V

VALOR ABSOLUTO. Es el número natural que resulta al suprimir su signo.

REFERENCIAS BIBLIOGRÁFICAS

Bibliografía

(s.f.).

Alazte, O. M. (2016). *Código Electrónica*. Obtenido de http://codigoelectronica.com/blog/arduino-tipos-de-datos

Arduino. (2010). Arduino Mega Technical specs. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/7370/1/AC-MECAT-ESPE-047427.pdf

Arduino CC. (29 de Septiembre de 2010). *Arduino*. Obtenido de https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Cádiz, U. d. (2011). Programación de Microcontroladores. Cádiz, España.

Carletti, E. (25 de Junio de 2015). *Argentina Robots*. Obtenido de http://robots-argentina.com.ar/MotorCC_PuenteH.htm

Carletti, E. J. (12 de Mayo de 2006). *Robots pasión por la robótica Argentina*. Obtenido de http://robots-argentina.com.ar/Comunicacion_busI2C.htm

Castrillo, F. (6 de Febrero de 2015). *Ahora Me Dedico Al Cacharreo*. Obtenido de http://ahoramededicoalcacharreo.blogspot.com/2015/02/funciones-matematicas-que-faltan-arco.html

Datasheet ST. (Enero de 2000). *Datasheet.* Obtenido de AllDataSheet.com

Educa Lab. (4 de Febrero de 2012). Obtenido de http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena11/4quincena11_contenidos_2b.htm

García, F. M. (11 de Enero de 2007). Obtenido de http://www.dia.uned.es/~fmorilla/MaterialDidactico/El%20controlador%20PID.pdf

Herrador, R. E. (2009). *Guía de Usuario de Arduino.* Córdoba.

Herramientas Informaticas. (22 de Septiembre de 2012). Obtenido de http://herramientasinformaticas2012.wikispaces.com/file/view/Encapsulados%20Electronicos.doc

Impulse, O. (1 de Mayo de 2012). *Open Impulse*. Obtenido de https://www.openimpulse.com/blog/products-page/25d-gearmotors/jga25-371-dc-gearmotor-encoder-201-rpm-12-v-2/

Klug, D. (2012). *Academia Educa*. Obtenido de http://www.academia.edu/16511961/Controlador_PID

MEDIALAB. (2010). Basicos Arduino.

Microbyte. (28 de Mayo de 2005). *Electro Industria*. Obtenido de http://www.emb.cl/electroindustria/articulo.mvc?xid=1539

Montes, J. A. (23 de Mayo de 2016). *Teorias PID*. Obtenido de http://www.automatas.org/hardware/teoria_pid.htm

Pérez, F. E. (2007). Microcontroladores Fundamentos y Aplicaciones con PIC. En F. E. Pérez, *Microcontroladores Fundamentos y Aplicaciones con PIC* (págs. 10-21). Barcelona: marcombo.

Ramos, J. (19 de Marzo de 2014). Obtenido de http://rootear.com/desarrollo/tipos-placas-arduino

Reyes, C. A. (2008). *Microcontroladores PIC Programación en Basic 3ra Edición.* RISPERGRAF.

Rivera, F. (2010). *Tecnológico Nacional de Mexico*. Obtenido de http://www.itap.edu.mx/documentos/tutoriales/APUNTES%20DE%20INSTRUMENTACION.doc

Rivera, M. A. (30 de Octubre de 2010). *C.I.R.E*. Obtenido de http://webdelcire.com/wordpress/archives/114

Sánchez., S. (19 de Mayo de 2013). *Blog de WordPress.com*. Obtenido de https://microcontroladoressesv.wordpress.com/empresas-fabricantes-de-microcontroladores/

Technology. (2013). *Apuntes de Arduino.*

TOVAR, G. S. (2014). *PROCESAMIENTO DIGITAL DE SEÑALES.* San Luis Potosí.

DATOS PERSONALES

Nombre:	SUNTAXI REIMUNDO JIMMY JEFFERSON
Nacionalidad:	Ecuatoriana
Fecha De Nacimiento:	02 DE MAYO DE 1994
Cedula de Ciudadanía:	1721732558
Teléfonos:	0979946289 / (02) 3525030
Correo Electrónico:	jeffersonsunta@gmail.com
Dirección Domicilio:	SANGOLQUI BARRRIO FAJARDO (Av. Mariana de Jesús y Av. El Inca)

ESTUDIOS REALIZADOS:

Primaria:

- Escuela “La Inmaculada” - Sangolqui

Secundaria:

- Unidad Educativa Fiscomisional Salesiana “Don Bosco” - Quito

Superior:

- Unidad de Gestión de Tecnologías “ESPE”

TITULOS OBTENIDOS:

- Bachiller Técnico Industrial en Electrónica

EXPERIENCIAS LABORALES:

- AERO-SARAYAKU Pasante – Técnico Electrónico

ACEPTACIÓN DEL USUARIO

Latacunga, 12 de Octubre del 2016

Yo, ING PABLO PILATÁSIG en calidad de encargado del Laboratorio de Instrumentación Virtual de la Unidad de Gestión de Tecnologías, me permito informar lo siguiente:

El proyecto técnico elaborado por el Sr. **SUNTAXI REIMUNDO JIMMY JEFFERSON**, con el tema " **IMPLEMENTACIÓN DE UN VEHÍCULO DE DOS RUEDAS AUTO-EQUILIBRADO PARA PRÁCTICAS DE MICROCONTROLADORES** ", ha sido efectuado de forma satisfactoria en las dependencias de mi cargo y que la misma cuenta con todas las garantías de funcionamiento, por lo cual extiendo este aval que respalda el trabajo realizado por el mencionado estudiante.

Por tanto me hago cargo de todas las instalaciones realizadas por el Sr. estudiante.

Atentamente,

_____

ING. PABLO PILATÁSIG

ENCARGADO DEL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL

Latacunga, 12 de Octubre del 2016

HOJA DE LEGALIZACIÓN DE FIRMAS

DEL CONTENIDO DE LA PRESENTE INVESTIGACIÓN SE RESPONSABILIZA EL AUTOR

_____

SUNTAXI REIMUNDO JIMMY JEFFERSON
ID L00052000

DIRECTOR DE LA CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

_____

Ing. Pablo Pilatásig Director de la Carrera de Electrónica Mención Instrumentación & Aviónica