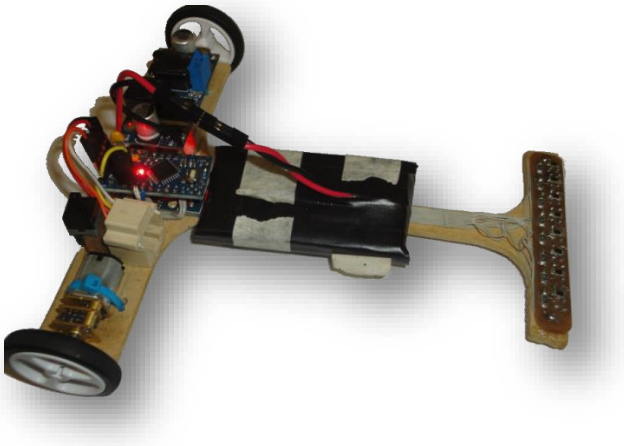
Universidad Tecnológica Metropolitana – Facultad de Ingeniería
Departamento de Electricidad
Club de Robótica UTEM (CRU)

Cátedra:

Taller de comunicación efectiva

Tema:

NEUTRINO: Robot Autónomo de Velocidad



Profesor:

Cecilia Harcha Rafachello

Autor:

Victor Arriagada Martinez

Diciembre 2014

Santiago – Diciembre 2014

0 Contenidos

1 Introducción

1.1 ¿Qué es un robot autónomo?

Un robot se define como una maquina conformada por un sistema electromecánico y que realiza tareas complejas facilitando el trabajo humano, la robótica es la ciencia que se encarga del diseño y desarrollo de estas máquinas, se apoya principalmente en áreas de conocimiento como la ingeniería en electrónica, ingeniería mecánica y sistemas informáticos.

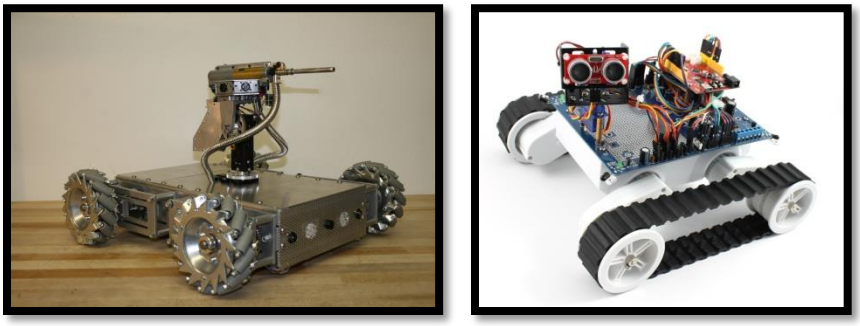


Figura 1.1.1: Robot omnidireccional (izquierda) y robot oruga (derecha).

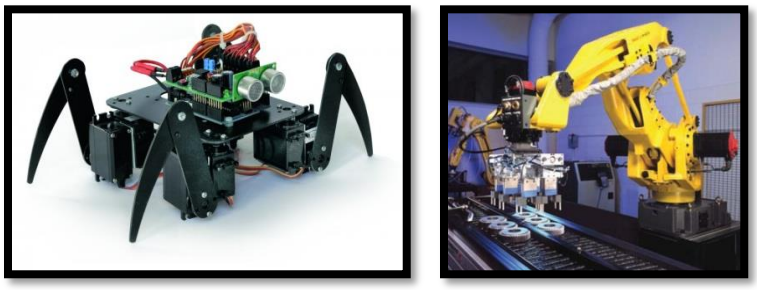


Figura 1.1.2: Robot araña (izquierda) y robot industrial tipo brazo (derecha).

Los robot pueden ser; terrestre, aéreo, acuático, anfibio, industriales, médicos, dependiendo de la tareas que debe realizar.

Figura 1.1.3: Robot antibombas (izquierda), robot cirujano (centro) y robot humanoide (derecha).

La autonomía de un robot es cuando funciona completamente autónomo esto quiere decir sin interacción directa de humanos, en donde la programación e instrucciones deben ser capaces de resolver el problema sin que tenga un operador directo.

1.2 Historia de NEUTRINO y del Club de Robótica UTEM

Neutrino se creó con el objetivo de concursar en una competencia de robótica, la idea nace a través del Club de Robótica UTEM que fue fundada por alumnos de la carrera ingeniería en electrónica en el año 2012 y es administrado por alumnos, donde los profesores solo participan como mentores y que aportan conocimiento al club, a lo largo de su vida el club ha desarrollado proyectos electrónicos y robóticos los cuales periódicamente expone a colegios, liceos y en eventos particulares, fomentando la tecnología y entretención de la robótica educativa, junto con lo anterior el club ha realizado talleres de arduino a los alumnos principalmente de primer año, con el objetivo de disminuir el desinterés y la deserción de los mismos.

El club está pasando por un proceso de independencia a con personalidad jurídica y así poder crecer y crear contactos tanto interno a la universidad como externo a ella con empresas del mismo interés del club.

2 Estructura básica de un robot

2.1 Estructura mecánica.

Sistema mecánico en donde se instala todos los sistemas del robot que a la vez protege de golpes, ruidos, señales e incluso de la manipulación sin autorización hacia el robot.

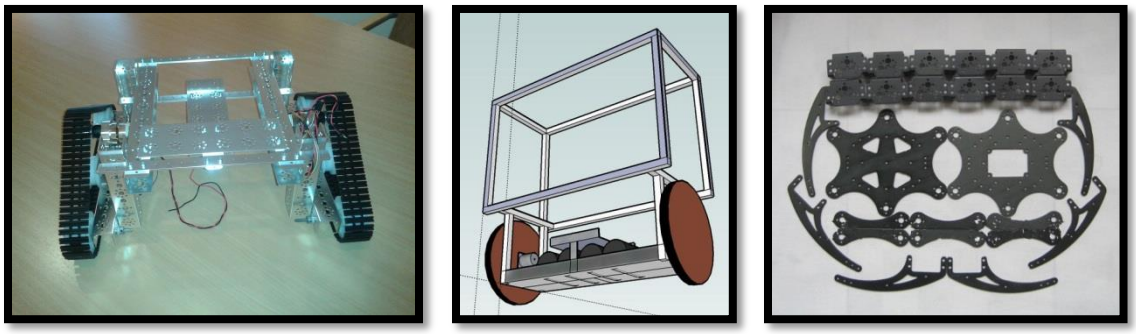


Figura 2.1.1: Base oruga (izquierda), estructura robot péndulo invertido (centro) y estructura robot araña (derecha).

2.2 Transceptores.

Son sistemas de comunicación normalmente inalámbrica, con el cual podemos enviar instrucciones o recibir información del robot.

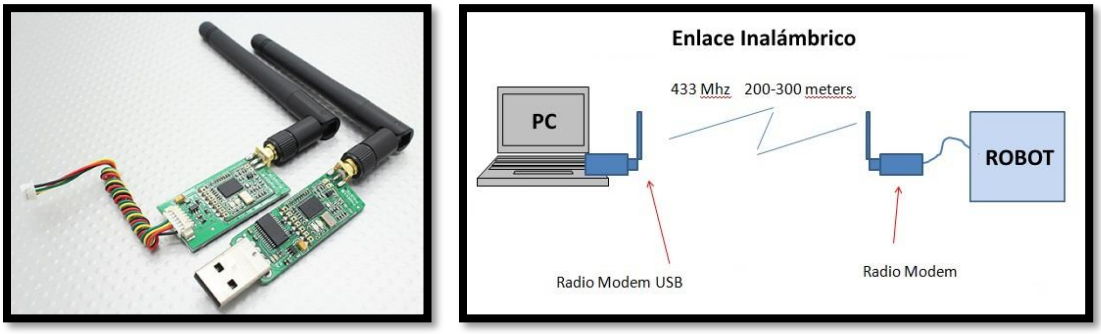


Figura 2.2.1: Módulos transceptores 433 mhz (izquierda) y esquema del enlace de los transceptores (derecha).

2.3 Sistema de accionamiento (actuadores).

Son sistemas que transforman una señal eléctrica a una mecánica como por ejemplo motores.

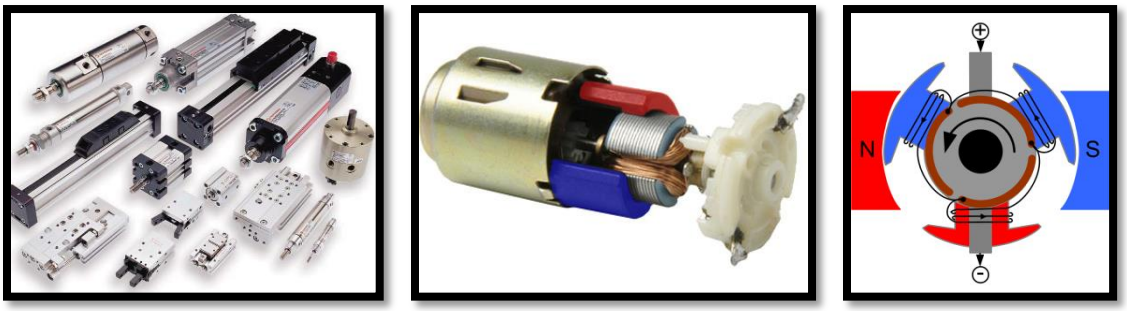


Figura 2.3.1: Motores lineales (izquierda), motor CC (centro) y estructura interna de un motor (derecha).

2.4 Sistemas sensoriales (sensores del entorno físico).

Son sistemas de instrumentos sensoriales que transforman una señal física a una eléctrica y así poder procesar las señales que recibe el robot, algunos de esos sensores son: sensores infrarrojo, de color, auditivos, tacto, luz, etc.

Figura 2.4.1: Sensores de un kit arduino (izquierda), sensor de sonido (centro) y sensor de distancia IR (derecha).

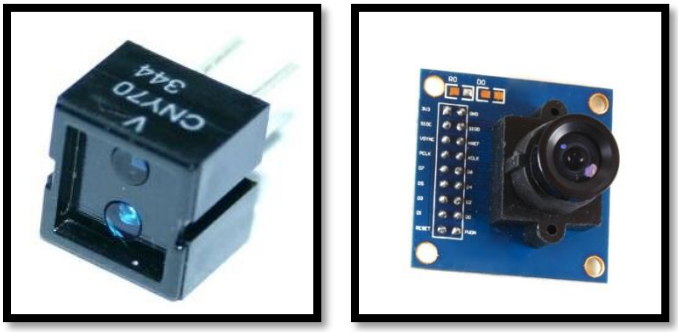


Figura 2.4.2: Sensor CNY70 (izquierda) y modulo cámara vga (derecha).

2.5 Sistemas de control (manual o automáticos).

Son sistemas donde se procesan la información de entrada (sensores) y condicionadas podemos enviarlas a los actuadores (motores) y hacer la acción necesaria para que el robot cumpla su trabajo.

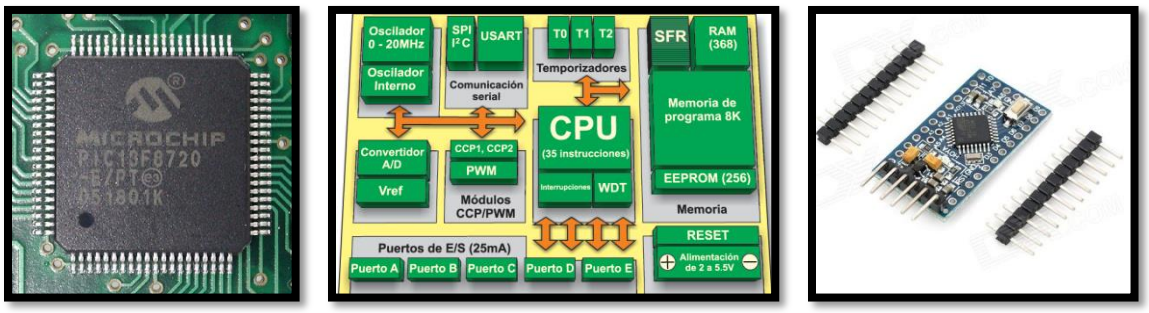


Figura 2.5.1: Micro controlador PIC SMD (izquierda), estructura interna de un micro (centro) y arduino mini (derecha).

2.6 Sistema de alimentación.

Son sistemas de energización del robot, normalmente se usan baterías, pero pueden ser incluso energía solar.

Figura 2.6.1: Pilas alcalinas (izquierda) y Baterías LIPO (derecha).

Figura 2.6.2: Batería de ácido (izquierda) y Paneles solares (derecha).

3 El robot de velocidad

Un robot de velocidad se caracteriza por tener una alta rapidez de procesamiento y de acción en los actuadores, es un robot móvil que tiene motores mucho más rápido que un robot convencional de exploración y que tiene la capacidad de tomar decisiones en una fracción de segundo, con esto podemos procesar una gran cantidad de información en muy poco tiempo y así corregir la dirección del robot a medida que se desplaza en el piso siguiendo una línea negra.

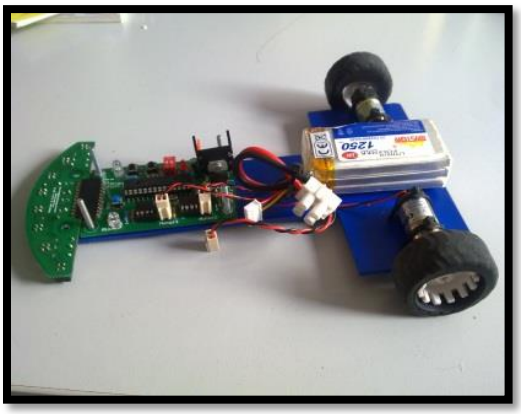


Figura 2.1: Típico robot de velocidad.

Figura 2.2: Pista de un robot de velocidad.

4 NEUTRINO

Neutrino es un robot autónomo de velocidad programado en lenguaje C que utiliza una plataforma llamada Arduino como sistema de control, diseñado y fabricado artesanalmente con el objetivo de representar a la Universidad Tecnológica Metropolitana del Estado de Chile a través del Club de Robótica UTEM en la competencia de robótica organizada por la Universidad Técnica Federico Santa María que se realizó en el mes de octubre de este año.

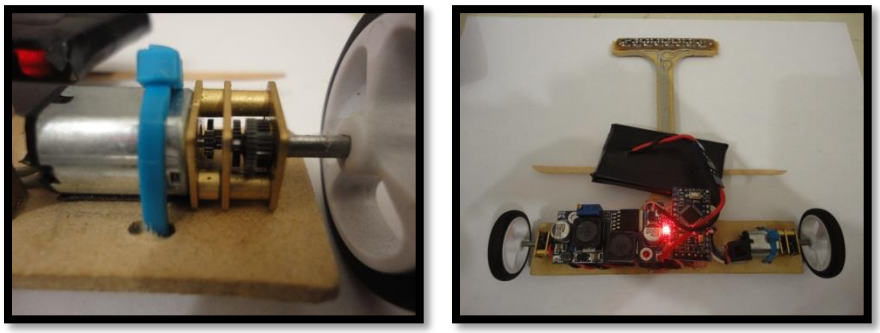


Figura 4.1: Motor de Neutrino (izquierda) y Neutrino v1.0 vista superior (derecha).

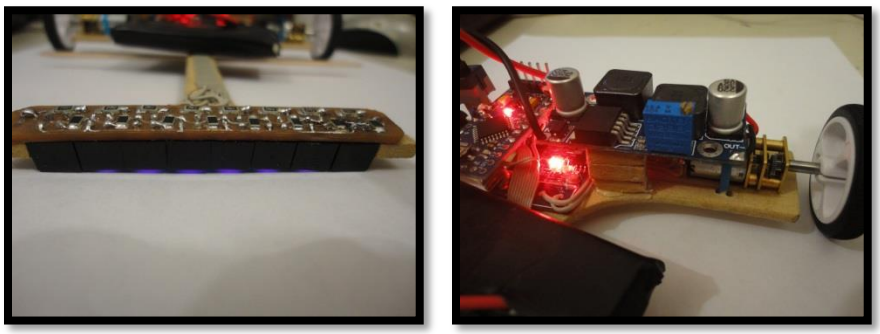


Figura 4.2: Vista de arreglo de sensores (izquierda) y Regulador de voltaje (derecha).

4.1 Marco o estructura.

Tiene una estructura física liviana, fabricada de madera trupan de 3 mm de espesor y que tiene un diseño apto para la detección rápida de la línea negra en el piso.

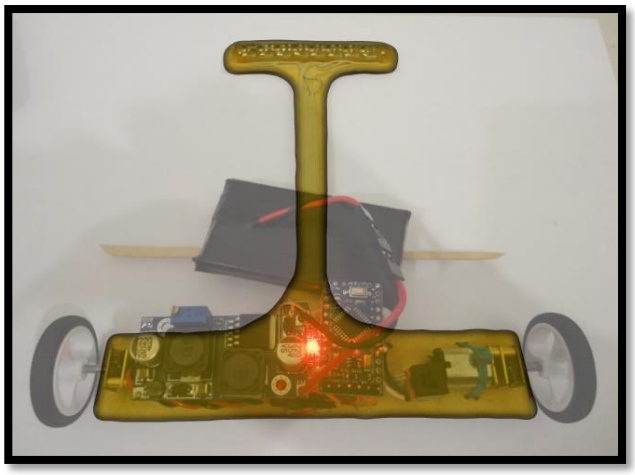


Figura 4.1.1: Estructura de madera para el soporte de los componentes (amarillo).

El corte de la madera se realizó con una caladora de mesa en el cual permite tener una muy buena precisión.

Figura 4.1.2: Maquina de corte trupan (izquierda) y hibrido de motor de maquina caladora de mesa (derecha).

4.2 Sistema de transmisión de datos.

Se usó temporalmente una comunicación inalámbrica en la etapa de construcción y prueba para analizar el comportamiento interno del robot (instrucciones y programación) así mismo para ajustar variables necesaria para el control PD que usa el robot; este módulo consta de un radio modem capaz de comunicar a una distancia inferior a 50 metros desde el computador hacia el robot.

Durante la competencia se eliminó este radio modem para cumplir con las bases propuestas por los organizadores.

Figura 4.2.1: Software configuración (izquierda), módulos transceptores (centro) y alcance de señal (derecha).

4.3 Actuadores.

Son motores de corriente continua y que tienen una caja reductora de engranajes para aumentar la fuerza del motor a cambio de velocidad, tienen una relación de 30:1 esto quiere decir que cuando el motor da 30 vueltas, la rueda solo da una vuelta, aun asa puede alcanzar una rapidez de 440 RPM (revoluciones por minuto o vueltas por minuto).

Estos motores son controlados por un módulo electrónico llamado puente H, el cual me permite controlar la rapidez, dirección y frenado del motor con señales eléctricas que provienen del micro controlador.

Figura 4.3.1: Motor y rueda unida (izquierda), motor cc (centro) y puente H (derecha).

4.4 Sensores.

El sistema de sensores consta de un arreglo de 7 sensores infrarrojo en línea, estos sensores son capaces de emitir una señal de luz infrarroja y detectar dicha luz con un fototransistor, cuando el sensor detecta negro, no circula corriente a través del fototransistor, y cuando está en blanco, circula una corriente el cual la puedo detectar eléctricamente hacia el micro controlador.

También tiene un sensor de voltaje que me permite testear la carga de la batería para inhabilitar los motores en caso que baje de un nivel preestablecido, así podemos proteger tanto la batería como al robot.

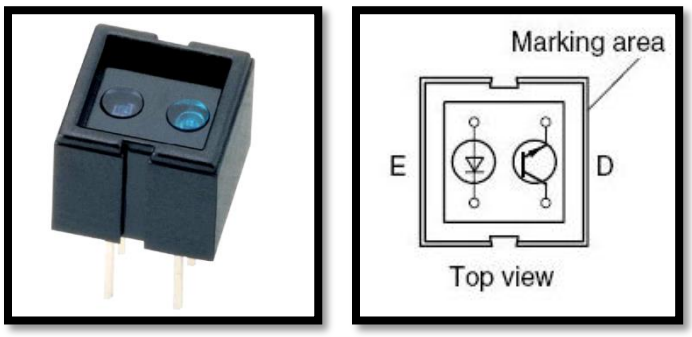


Figura 4.4.1: Sensor CNY70 (izquierda) y estructura interna del CNY70 (derecha).

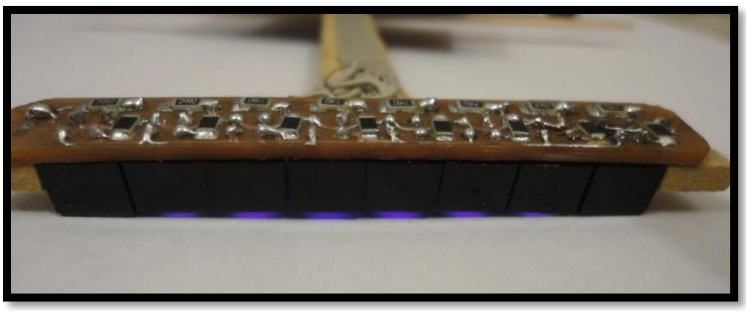


Figura 4.4.2: Arreglo de sensores CNY70.

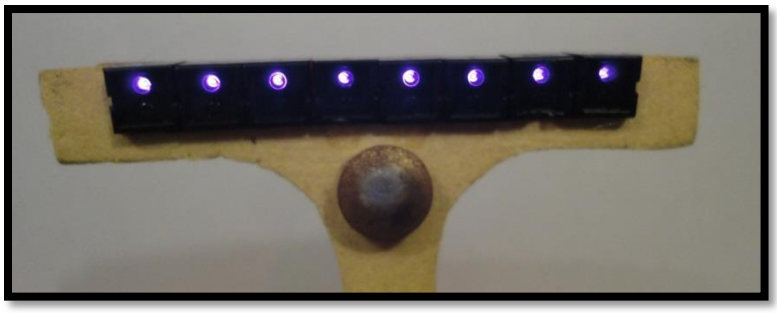


Figura 4.4.3: Arreglo de sensores CNY70, solo con cámara se ven las luces encendidas (transistores).

4.5 Sistema de alimentación.

En este sistema consta de una batería LIPO de una celda (3.7v) con una capacidad de 1300 mA/hora esto quiere decir que si una carga o un artefacto consume 1300 mili amperes, la batería será capaz de proporcionar energía por una hora, análogamente si el sistema solo consume 100 mili amperes, la batería durara alrededor de 13 horas.

El sistema consta también de un regulador de voltaje switching step up, (elevador de tensión) el cual me eleva el voltaje a 10 v aproximadamente para poder alimentar tanto los motores como la parte lógica (micro controlador y sensores).

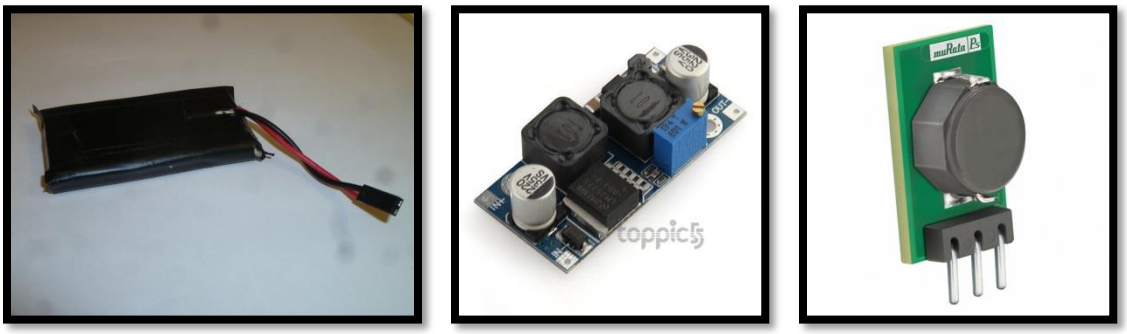


Figura 4.5.1: Batería LIPO de 1 celda modificada (izquierda), elevador de tensión (centro) y regulador 5V (derecha).

4.6 Sistema de control.

Es un sistema compuesto por un micro controlador atmega328 (Arduino mini) el cual es capaz de realizar las operaciones lógicas y aritméticas necesarias para realizar el procesamiento del robot. Se programa en un lenguaje C y se envía a través de un módulo serial diseñado para esta función.

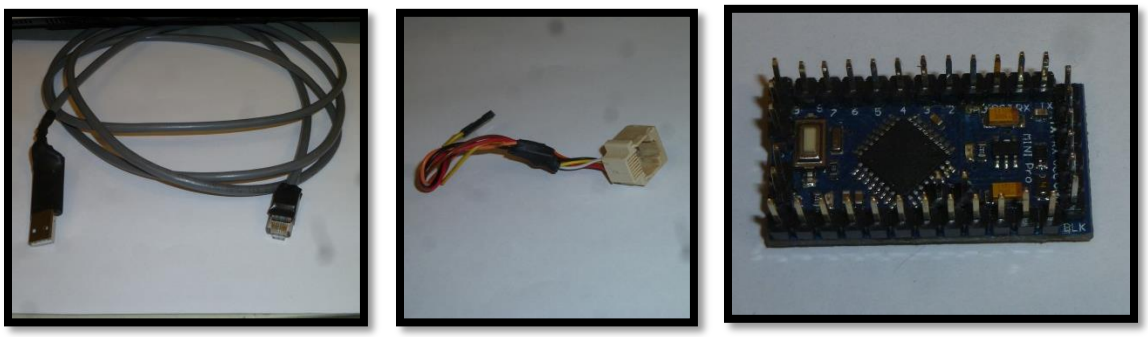


Figura 4.6.1: Cable programador (izquierda), conector para cable programador (centro) y arduino mini (derecha).

5 Características de NEUTRINO

5.1 Organización de componentes.

En la estructura va montado todos los sistemas, organizados y orientados de la manera más eficiente:

- En la zona trasera van los motores, las ruedas, el regulador de voltaje el micro controlador (Arduino) y el puente H.
- En la zona central está la batería.
- En la zona delantera va el arreglo de sensores y el pivote.

Todo esto está organizado dentro del marco o estructura que tiene una dimensión de aproximadamente 225 cm^2

En el siguiente esquemático electrónico se muestra como van las conexiones de los módulos al micro controlador, como también los motores, sensores y fuente de alimentación.

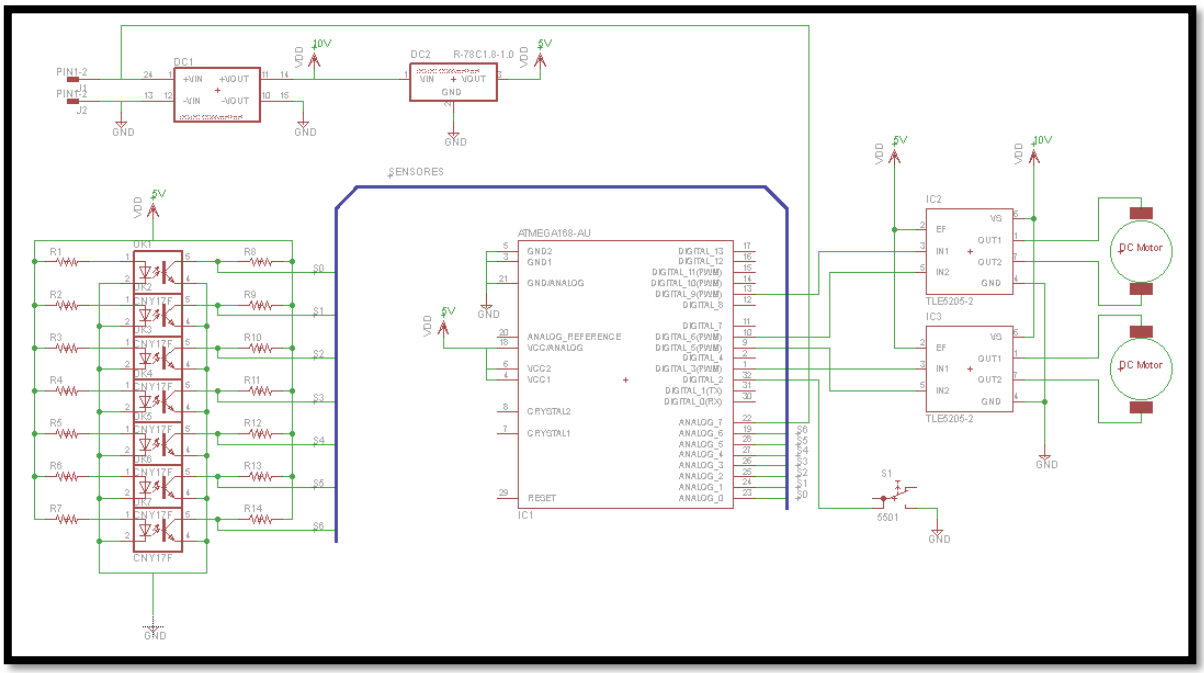


Figura 5.1.1: Esquemático electrónico del robot completo NEUTRINO.

5.2 Características de los componentes.

5.2.1 Batería LIPO.

Son baterías basadas en polímero de Litio que consiguen un almacenamiento muy superior de energía, son también muy ligeras, pesando cerca de la mitad de una equivalente, a pesar del precio elevado las ventajas de las baterías de Litio las han popularizado llegando a ser las más cotizadas en robótica.

Estas baterías se componen de varias celdas en serie dependiendo de las necesidades, en el caso de Neutrino, solo necesita una celda, la cual cuenta con un voltaje nominal de 3.7 v llegando a un máximo de 4v; debemos tener mucho cuidado al manipularla, sobretodo en su descarga ya que si el voltaje disminuye por debajo de los 3 voltios, podemos perder la batería para siempre, por ese motivo nuestro sistema debe tener un proceso de testeo de dicho voltaje para poder proteger la batería y el sistema.

Para cargar este tipo de batería es necesario usar un cargador diseñado para este fin.

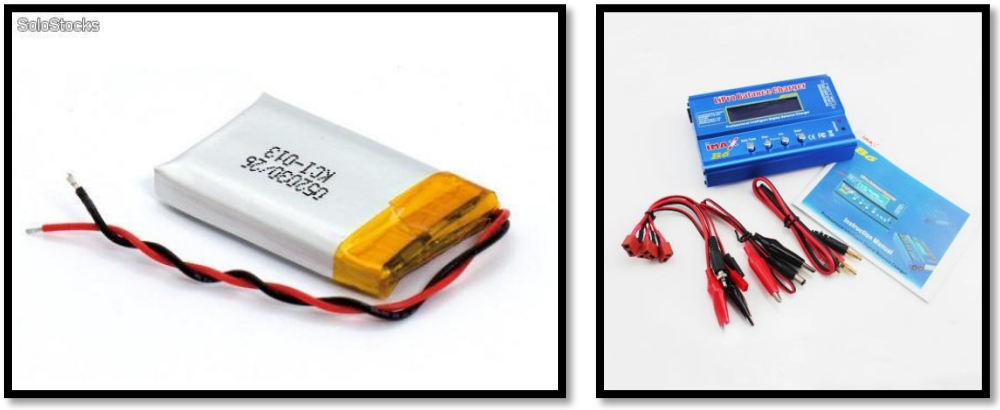


Figura 5.2.1.1: Batería LIPO 1 celda (izquierda) y cargador de baterías LIPO y otras (derecha).

5.2.2 Regulador de voltaje boost (elevador de tensión).

El convertidor Boost (o elevador) es un convertidor de voltaje que en su salida se obtiene una tensión continua mayor que en su entrada, en este caso la entrada será de 3.7v nominal y en su salida tendremos 10v nominal, con esto podemos alimentar los motores sin ningún problema.

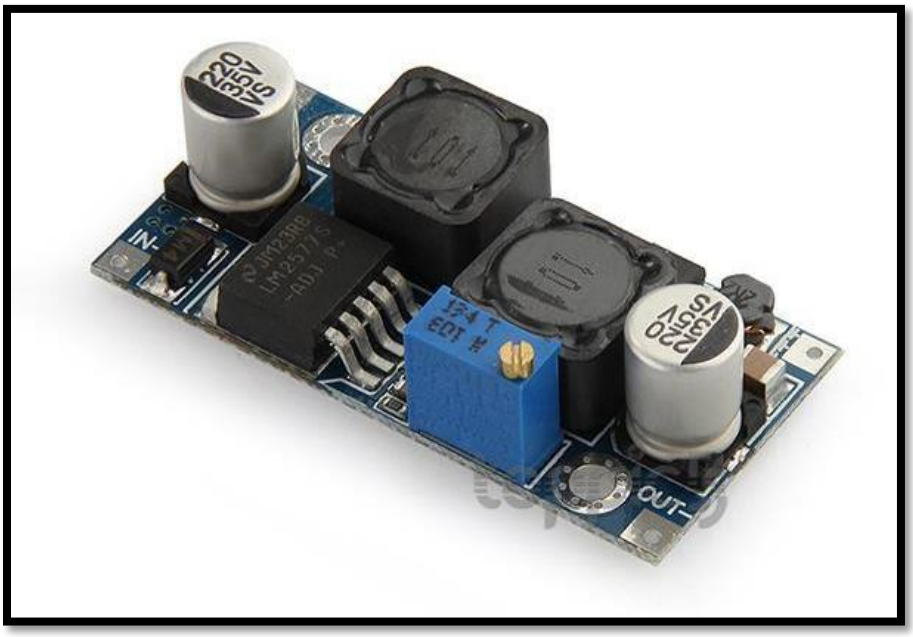


Figura 5.2.2.1: Regulador de voltaje como elevador de tensión.

Para poder controlar la salida solo hace falta ajustar con un atornillador de precisión el potenciómetro de ajuste que se encuentra en el regulador.

Figura 5.2.2.2: Potenciómetro de ajuste para fijar el voltaje de salida.

5.2.3 Regulador de voltaje 5v (reductor de voltaje).

Es un regulador switching que reduce el voltaje en su salida siendo menor en comparación a su entrada, lo usaremos para regular el voltaje a 5v que necesita los sensores y el micro controlador, este tipo de reguladores no son ajustables, siempre tendrán a su salida 5v estables.

Figura 5.2.2.1: Regulador de voltaje fijo a 5V.

5.2.4 Motores CC.

Neutrino usa un par de micro motores de corriente continua capaz de proporcionarle tracción diferencial al móvil y con una velocidad prometedora; su voltaje nominar es de 6v, pero su fabricante recomienda el uso de un voltaje mayor no superior a 9 v para una mayor potencia, sus dimensiones son de 12 x 10 x 26 mm lo cual son muy pequeños.

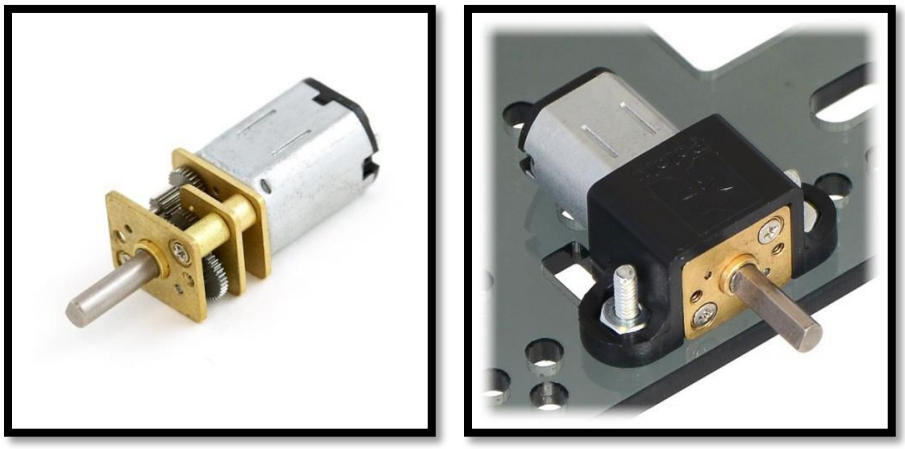


Figura 5.2.4.1: Motor CC con caja reductora (izquierda) y montaje a base del motor (derecha).

5.2.5 Ruedas.

Durante el proceso de fabricación, se diseñó con unas ruedas de 32mm de diámetro, los cuales al comienzo de la competencia fueron ineficientes al momento de participar, es por ello que se reemplazaron por unas de 60 mm diámetro.

Ambos set de ruedas son de plástico y son muy livianos, tienen unas gomas para que se adhiera si problema en cualquier superficie plana.

Figura 5.2.5.1: Ruedas de 32 mm (izquierda), rueda con motor (centro) y ruedas de distintos diámetros (derecha).

5.2.6 Puente H.

Para el control de los motores necesitamos un módulo llamado Puente H, este módulo consta de un circuito integrado diseñado para esta función con él se controla la velocidad por PWM, la dirección de giro y el frenado del motor, el modulo tiene la capacidad de controlar dos motores independiente con un máximo de 10v y 1.2 amperes por motor, cabe señalar que tiene una caída de tensión de 1.2 voltios por motor, lo que significa que si alimentamos el puente h con 10v, en los motores tendrá un voltaje de 8.8 v aproximadamente, es por esta razón que el elevador de tensión está ajustado a 10v.

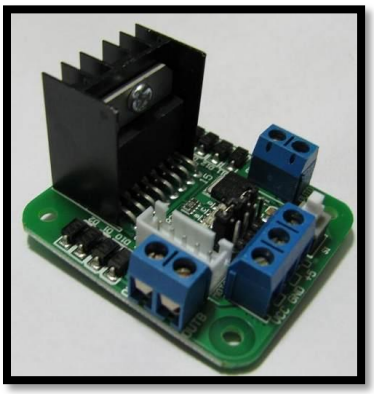


Figura 5.2.6.1: Puente H.

5.2.7 Sensores IR (Infrarrojos).

Los CNY70 son unos sensores reflectivos infrarrojos que emiten una señal cuando refleja la luz invisible al ojo humano frente a una superficie que permita esta acción, normalmente se usa a una distancia de 5mm de la superficie a sensar.

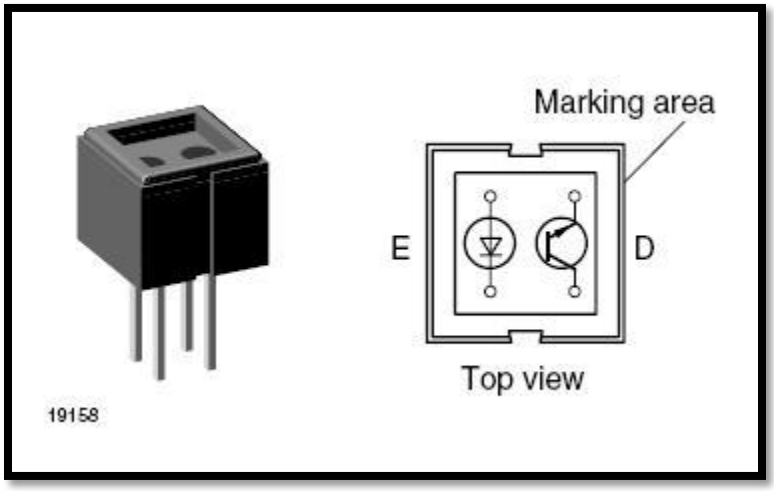


Figura 5.2.7.1: CNY70, imagen física y estructura interna.

En el Neutrino se diseñó una tarjeta PCB para colocar 8 sensores en paralelo, posteriormente se modificó y se eliminó uno de los sensores para que mejore su funcionamiento al detectar la línea negra del piso.

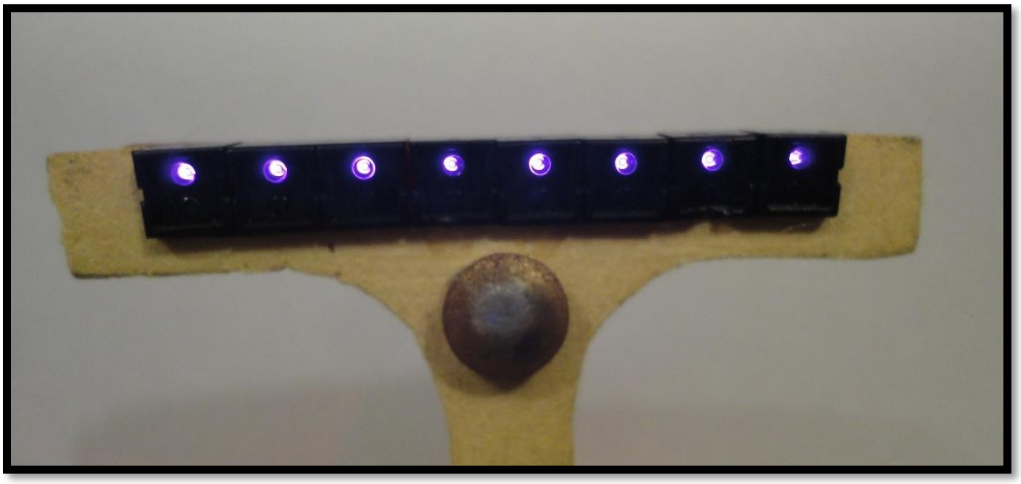


Figura 5.2.7.2: Vista inferior de arreglo sensores infrarrojos.

Para el uso de estos sensores es necesario incorporar 2 resistencias, una para limitar la corriente que llega al emisor y el otro para medir la corriente que circula por el fototransistor del receptor, así se puede discriminar cuando el sensor esta sobre la línea negra o el piso blanco.

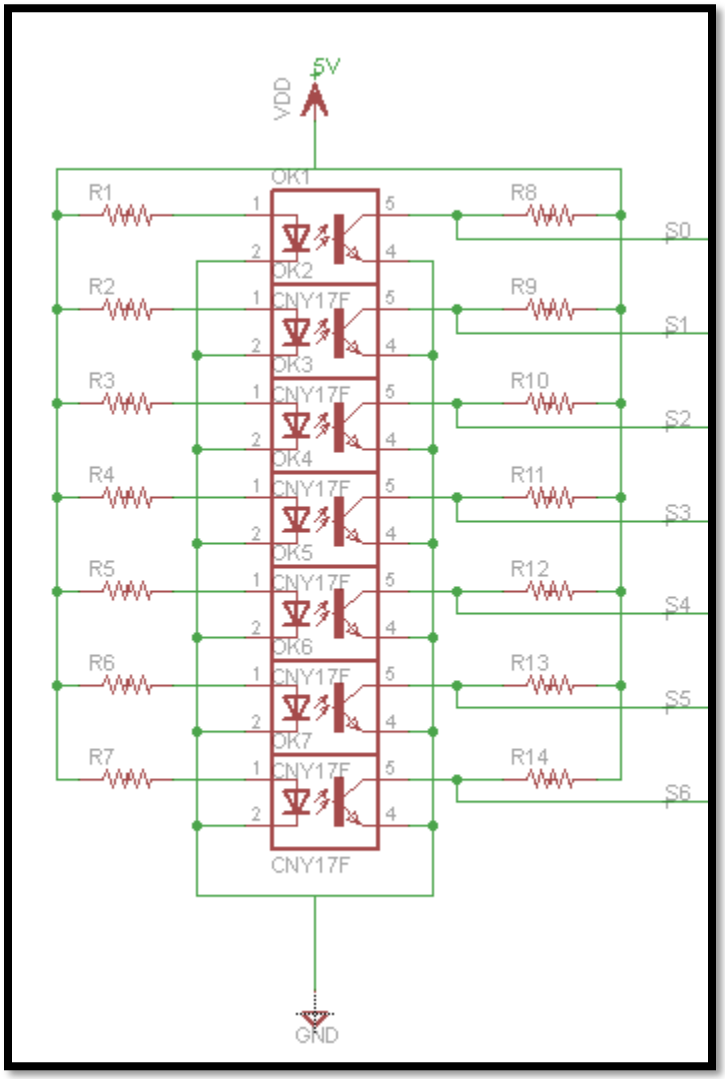


Figura 5.2.7.3: Esquemático electrónico de la etapa de sensores.

5.2.8 Micro controlador (Arduino).

Para poder controlar todos los componentes y procesar los datos se implementó el Arduino mini, este arduino es un módulo gobernado por el micro controlador ATmega 328 de la familia AVR de Atmel, opera con un voltaje de 5v, tiene 14 entradas o salidas digitales configurables de los cuales 6 pueden ser salidas PWM independientes, tiene 8 entradas análogas, soporta hasta 40ma por pin, memoria flash de 32kb (2k para el bootload), 2kb de RAM y 1kb de EEPROM, lo que significa que es una mini computadora con entradas, salidas, memoria ram, y CPU; es tan pequeño que no supera los 30 x 18 mm.

Este módulo se programa en un computador con código C y en un entorno desarrollador llamado Arduino IDE, este compilador se encarga de traducir el código C (instrucciones) a código máquina y lo envía a la memoria flash que luego lo ejecuta.

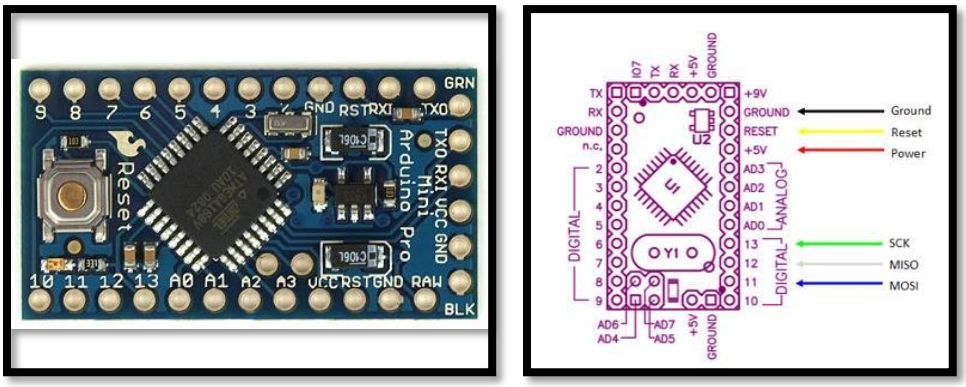


Figura 5.2.8.1: Fotografía Arduino mini (izquierda) y nomenclatura de los pines del Arduino mini (derecha).

Figura 5.2.8.2: Software Arduino IDE que sirve para programar las instrucciones al módulo Arduino.

6 Seguidor de líneas básico

Para que el robot pueda seguir líneas la forma más simple es leyendo el arreglo de sensores infrarrojo y dándole ciertas condiciones, se puede usar solo 2 sensores que estén encima de la línea y cuando uno sale de la línea los motores se accionan girando brevemente al sentido contrario para que vuelvan a leer la línea negra.

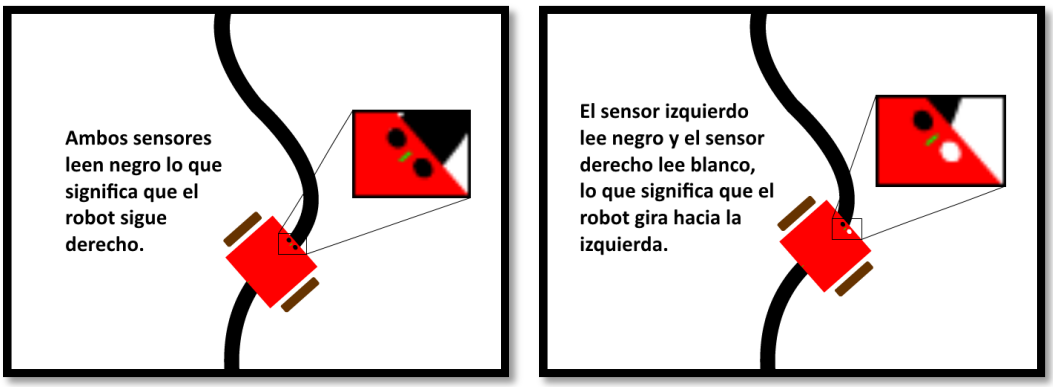


Figura 6a: Condiciones cuando el robot debe seguir derecho y cuando debe girar a la izquierda.

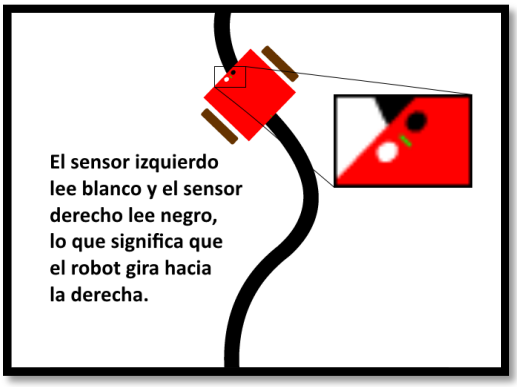


Figura 6b: Condiciones cuando el robot debe girar a la derecha.

6.1 Puesta en marcha del robot con sigue líneas básico.

Una vez todo montado y conectado se procede con crear el código de las instrucciones, en este informe no se explicara como programar, solo se explicara el cómo funciona el código para el funcionamiento de este robot en particular.

6.2 Código (Sigue_líneas_básico.ino)

```
void setup(){
  Serial.begin(115200);
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  digitalWrite(3,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
}

void loop(){
  bateria();
  if((analogRead(2)<500)&&(analogRead(4)<500)){
    digitalWrite(9,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(3,LOW);
    digitalWrite(5,HIGH);
  }else if((analogRead(2)<500)&&(analogRead(4)>500)){
    digitalWrite(9,LOW);
    digitalWrite(6,HIGH);
    digitalWrite(3,LOW);
    digitalWrite(5,HIGH);
  }else if((analogRead(2)>500)&&(analogRead(4)<500)){
    digitalWrite(9,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(3,HIGH);
    digitalWrite(5,LOW);
  }
}
```

7 Seguidor de líneas con control PID

Unos de los problemas de la lógica básica es que zigzagea mucho y pierde eficiencia al seguir la línea, es por este motivo que se usa un control automático llamado Control PID.

PID: Control por retroalimentación que calcula la deviación del error usando tres parámetros llamados Proporcional, Integral y Derivativo; con esto podemos controlar y corregir el error de forma automática y suave.

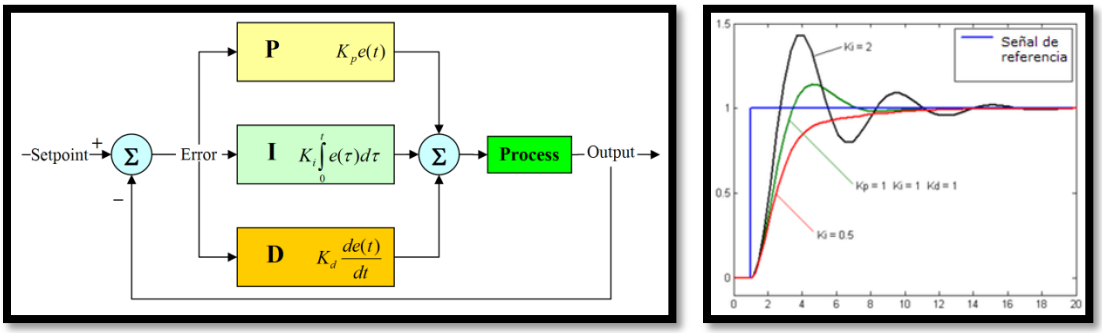


Figura 7a: Estructura del control PID (izquierda) y grafico de respuesta de un control PID (derecha).

Como se puede apreciar en la siguiente imagen la comparación de un sigue líneas sin PID y con PID.

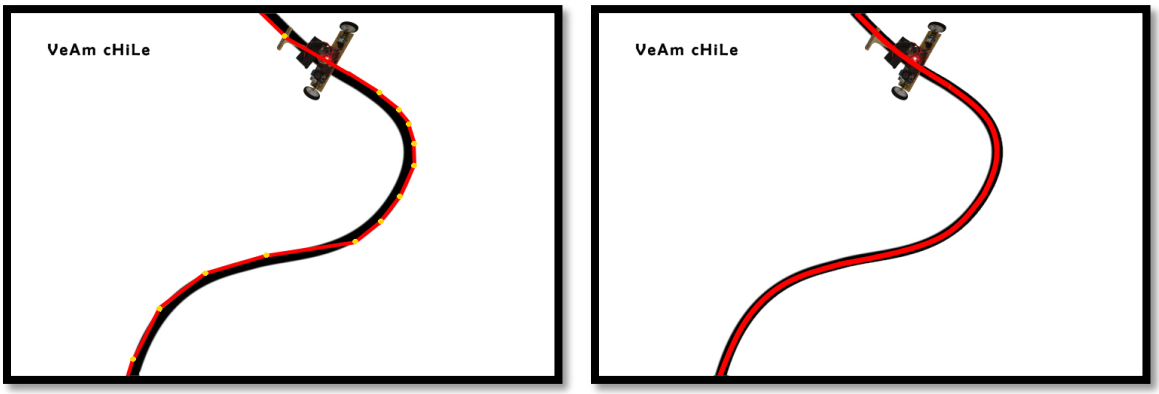
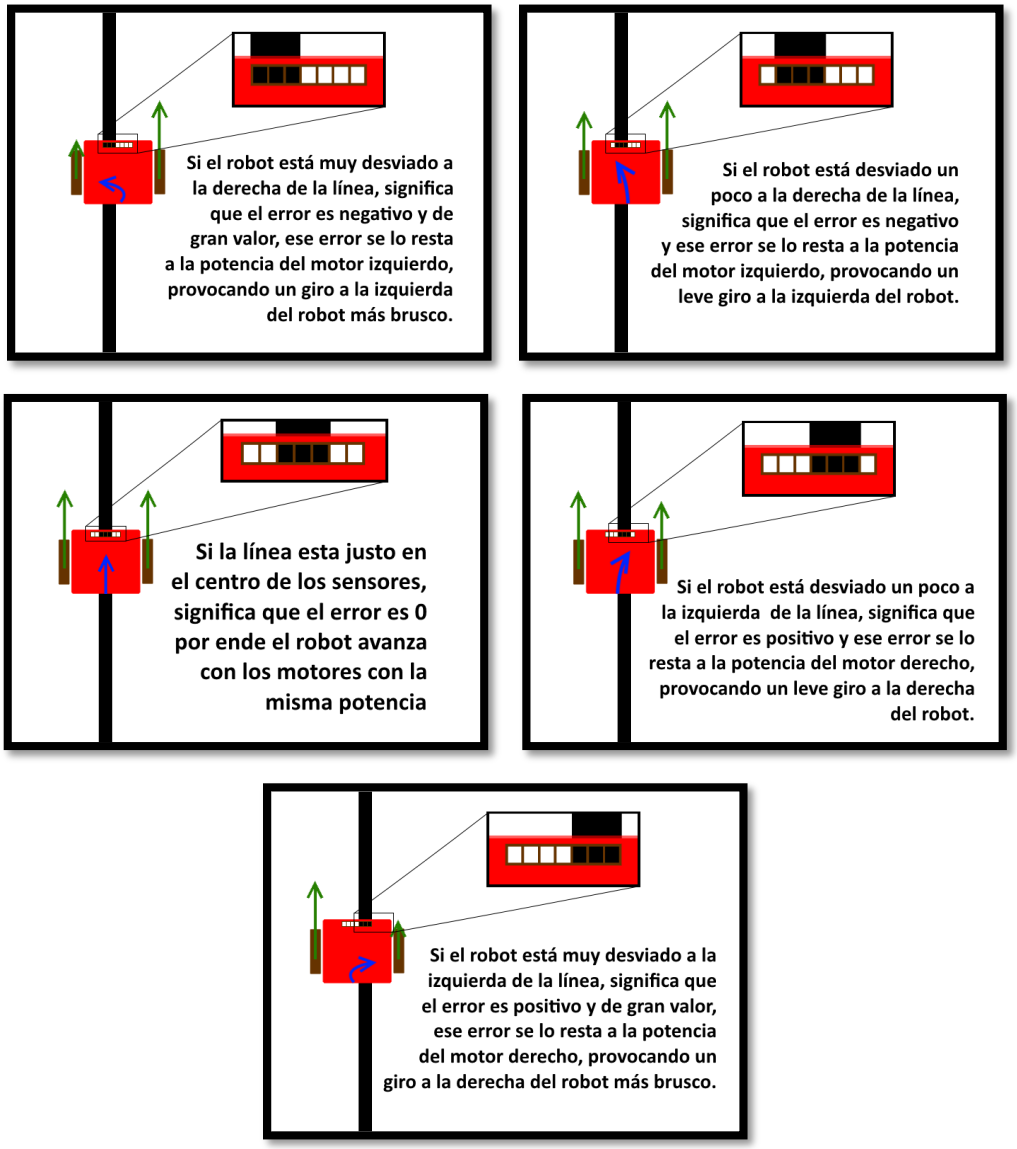


Figura 7b: Sin control PID, no es curvo y suave (izquierda) y con control PID es curvo y suave (derecha).

7.1 Lógica del seguidor con PID

Lo primero que se hace es leer todos los sensores del arreglo infrarrojo y transformarlo a una sola variable, después se fija un valor medio en donde está justo en el centro de la línea y se obtiene la diferencia, al desviarse un poco de la línea se genera una diferencia que la llamaremos ERROR, con este dato podemos obtener el cálculo proporcional, integral y derivativo, lo cual cada una se multiplica por un factor fijo llamados Kp, Ki y Kd respectivamente, y finalmente se suman los tres resultados que modificaran la velocidad de los motores provocando que el robot corrija su trayectoria y vuelva a la línea.



7.2 Código seguidor con control PD (Sigue_líneas_PID.ino)

```
#include <EEPROMex.h>
int time=0;
long sensors_average;
int sensors_sum;
float position;
float proportional;
float integral;
float derivative;
float last_proportional;
int Set_point=3000;
float right_speed;
float left_speed;
float error_value;
float Kp=1.28;
float Ki=0;
float Kd=12.37;
int max_speed=180;
long sensors[] = {0, 0, 0, 0, 0, 0, 0, 0};
int motor=1;
int valor_bateria;

void recibir(){
  if (Serial.available() > 0) {
    int variable=Serial.read();
       switch (variable){
          case 'A':
               Kp=Kp+0.1;
               EEPROM.writeFloat(0,Kp);
               Serial.println(EEPROM.readFloat(0));
          break;
          case 'a':
               Kp=Kp+0.01;
               EEPROM.writeFloat(0,Kp);
               Serial.println(EEPROM.readFloat(0));
          break;
          case 'B':
               Kp=Kp-0.1;
               EEPROM.writeFloat(0,Kp);
               Serial.println(EEPROM.readFloat(0));
          break;
         case 'b':
               Kp=Kp-0.01;
               EEPROM.writeFloat(0,Kp);
               Serial.println(EEPROM.readFloat(0));
          break;
          case 'C':
               Ki=Ki+0.1;
               EEPROM.writeFloat(10,Ki);
               Serial.println(EEPROM.readFloat(10));
          break;
          case 'c':
               Ki=Ki+0.01;
               EEPROM.writeFloat(10,Ki);
               Serial.println(EEPROM.readFloat(10));
          break;
          case 'D':
               Ki=Ki-0.1;
               EEPROM.writeFloat(10,Ki);
               Serial.println(EEPROM.readFloat(10));
          break;
          case 'd':
               Ki=Ki-0.01;
               EEPROM.writeFloat(10,Ki);
               Serial.println(EEPROM.readFloat(10));
          break;
          case 'E':
               Kd=Kd+0.1;
               EEPROM.writeFloat(20,Kd);
               Serial.println(EEPROM.readFloat(20));
          break;
          case 'e':
               Kd=Kd+0.01;
               EEPROM.writeFloat(20,Kd);
               Serial.println(EEPROM.readFloat(20));
          break;
          case 'F':
               Kd=Kd-0.1;
               EEPROM.writeFloat(20,Kd);
               Serial.println(EEPROM.readFloat(20));
          break;
          case 'f':
               Kd=Kd-0.01;
               EEPROM.writeFloat(20,Kd);
               Serial.println(EEPROM.readFloat(20));
          break;
          case 'G':
```

```
                max_speed=max_speed+5;
                if(max_speed>255){max_speed=255;}
                EEPROM.write(30,max_speed);
                Serial.println(EEPROM.read(30));
             break;
            case 'g':
                max_speed=max_speed+1;
                if(max_speed>255){max_speed=255;}
                EEPROM.write(30,max_speed);
                Serial.println(EEPROM.read(30));
             break;
             case 'H':
                max_speed=max_speed-5;
                if(max_speed<0){max_speed=0;}
                EEPROM.write(30,max_speed);
                Serial.println(EEPROM.read(30));
             break;
            case 'h':
                max_speed=max_speed-1;
                if(max_speed<0){max_speed=0;}
                EEPROM.write(30,max_speed);
                Serial.println(EEPROM.read(30));
             break;
              case 'M':
                motor=1;
             break;
              case 'm':
                motor=0;
             break;
             case 'P':
                 Serial.println(EEPROM.readFloat(0));
                 Serial.println(EEPROM.readFloat(10));
                 Serial.println(EEPROM.readFloat(20));
                 Serial.println(EEPROM.read(30));

             break;
             case 'K':
                float floatvalor = valor_bateria;
                floatvalor=floatvalor/100;
                Serial.println(floatvalor);
             break;

          }
  }
}

void setup(){
 //while(digitalRead(2));
 //Kp=EEPROM.readFloat(0);
 //Ki=EEPROM.readFloat(10);
 //Kd=EEPROM.readFloat(20);
 //max_speed=EEPROM.read(30);

 //EEPROM.writeFloat(0,Kp);
 //EEPROM.writeFloat(10,Ki);
 //EEPROM.writeFloat(20,Kd);
 //EEPROM.write(30,max_speed);


  Serial.begin(115200);
  Serial.setTimeout(50);
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(2,INPUT);
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  while(digitalRead(2));
  delay(3000);
  time=millis();
}

void sensors_read(){
  sensors_average = 0;
  sensors_sum = 0;
  for (int i = 0; i < 7; i++){
    sensors[i] = analogRead(i);
    sensors_average += sensors[i] * i * 1000;
    sensors_sum += int(sensors[i]);
  }
  // position = int(sensors_average / sensors_sum);
  // Serial.println(position);
```

```
}



void pid_calc(){
  position = int(sensors_average / sensors_sum);
  proportional = position-Set_point;
  integral = integral + proportional;
  derivative = proportional - last_proportional;
  last_proportional = proportional;
  error_value = int(proportional * Kp + ((integral * Ki)/100) + derivative * Kd);
   //  Serial.print(error_value);
  //   Serial.print("    ");
}

void calc_turn(){
  if (error_value< -255){
    error_value = -255;
  }
  if (error_value> 255){
    error_value = 255;
  }
  if (error_value < 0){
    right_speed = max_speed ;
    left_speed = max_speed + error_value;
  }else{
    right_speed = max_speed - error_value;
    left_speed = max_speed;
  }
}

void motores(int pwm1, int pwm2){
  int freno=1;
  if(pwm1>255){pwm1=255;}
  if(pwm1<-255){pwm1=-255;}
  if(pwm2>255){pwm2=255;}
  if(pwm2<-255){pwm2=-255;}

  if(pwm1>0){
   digitalWrite(9,LOW);
   analogWrite(6,pwm1);
  }else if(pwm1<0){
   digitalWrite(6,LOW);
   analogWrite(9,-pwm1);
  }else{
   digitalWrite(6,freno);
   digitalWrite(9,freno);
  }

  if(pwm2>0){
   digitalWrite(3,LOW);
   analogWrite(5,pwm2);
  }else if(pwm2<0){
   digitalWrite(5,LOW);
   analogWrite(3,-pwm2);
  }else{
   digitalWrite(5,freno);
   digitalWrite(3,freno);
  }
}

int bateria(){
  valor_bateria=(map(analogRead(7),0,1023,0,500));
  if(valor_bateria<310){
     delay(200);
     if(valor_bateria<310){
     motores(0,0);
     while(1);
   }
  }
}

void loop(){
  if((millis()-time)<100){
    max_speed=160;
    Kp=0.8;
    Kd=9.37;
    //Serial.println("1");
  }else if((millis()-time)>=1000){
    max_speed=255;
    Kp=1.18;
    Kd=14.00;
    //Serial.println("2");
  }
  Serial.println(max_speed);
  bateria();
```

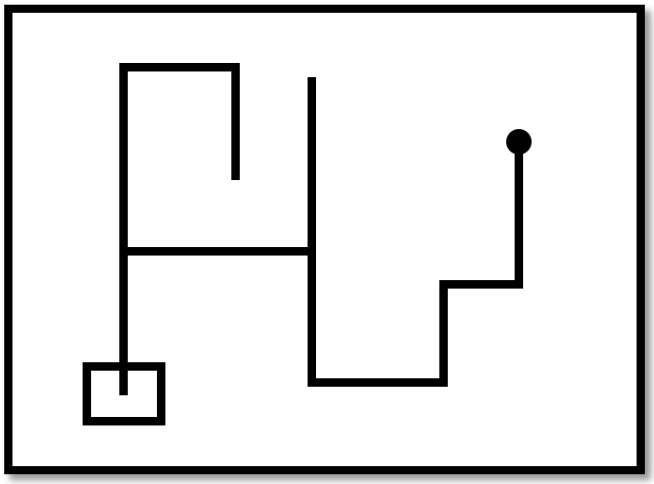
```
  //recibir();
  sensors_read();
  pid_calc();
  calc_turn();
  if(motor==1){
    motores(right_speed, left_speed);
  }else if(motor==0){
    motores(0, 0);
  }

  if((analogRead(0)>500)&&(analogRead(1)>500)&&(analogRead(2)>500)&&(analogRead(3)>500)&&(analogRead(4)>500)&&(analogRead(5)>500)&&(analogRead(6)>500)){
     //motor=0;
  }else if((analogRead(0)<500)){
    while(analogRead(3)>500){
     motores(0,255);
    }
  }else if((analogRead(6)<500)){
     while(analogRead(3)>500){
       motores(255,0);
     }
  }
}
```

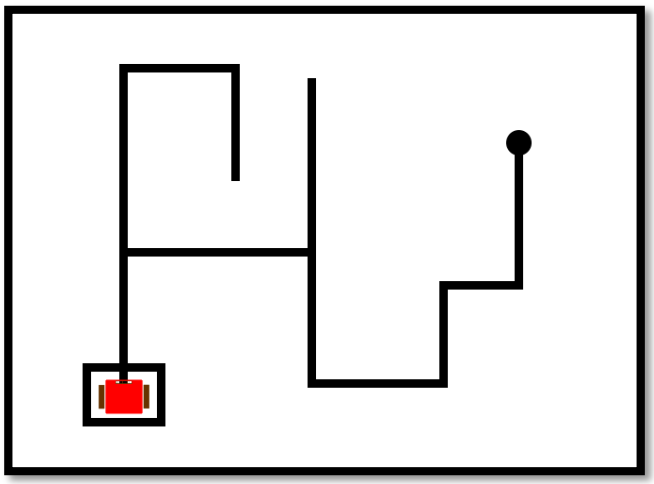
8 Laberinto

8.1 Lógica de cómo resolver un laberinto.

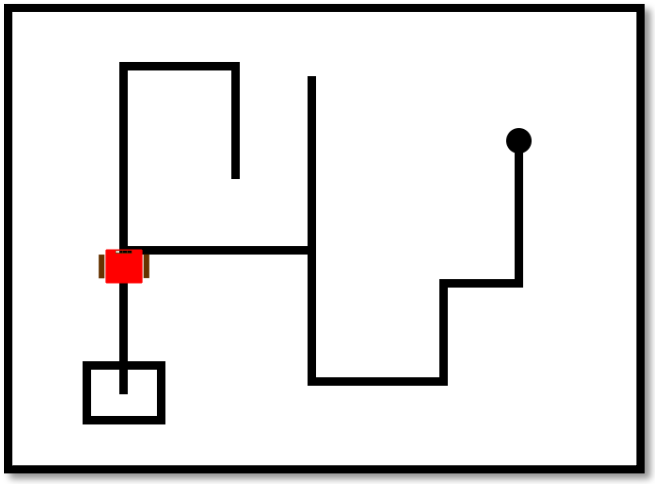
Para resolver un laberinto debemos usar un método muy eficaz llamado regla de la mano izquierda (o de la mano derecha), este método consta de ciertas reglas obligatorias en donde dice que el robot siempre girara a la izquierda cuando pueda (en cruces de líneas) el cual garantiza que sin importar cuanto se demore, siempre llegara al final del laberinto, pero para que no sea tan sencillo, el robot deberá memorizar por cual camino paso para así retornar al principio por el camino más corto.



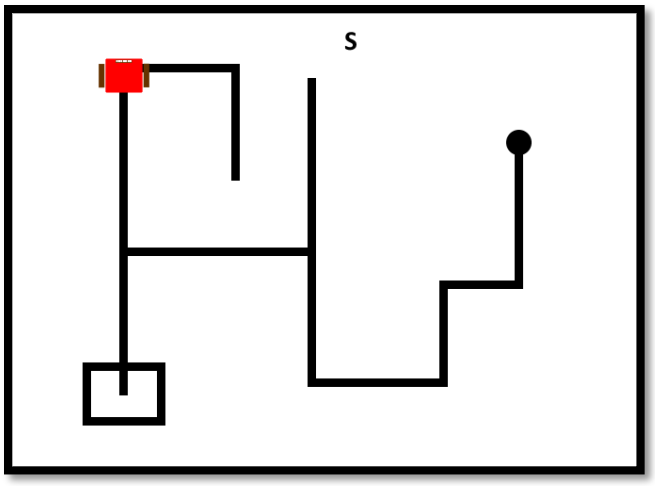
Como ya se vio en el capítulo anterior el como el robot puede seguir líneas negras, nos saltaremos ese paso.



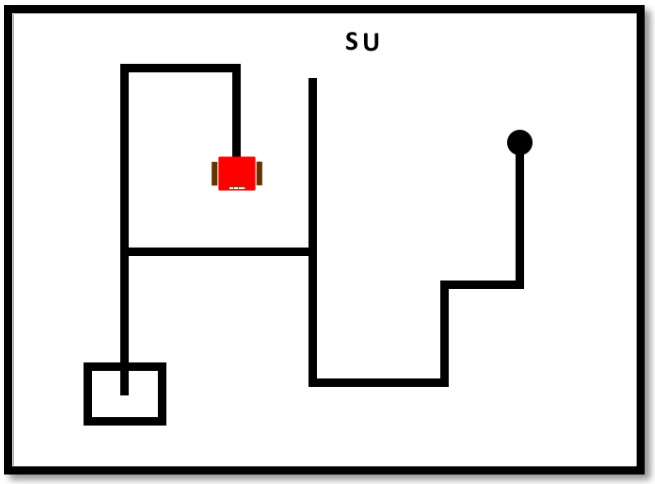
Al partir el robot debe seguir la línea como siempre hasta que llegue a un cruce, o un giro obligado o un fin de línea, es ese punto evalúa con los sensores si existen líneas adyacentes o no hay, y toma una decisión priorizando la regla de la mano izquierda.



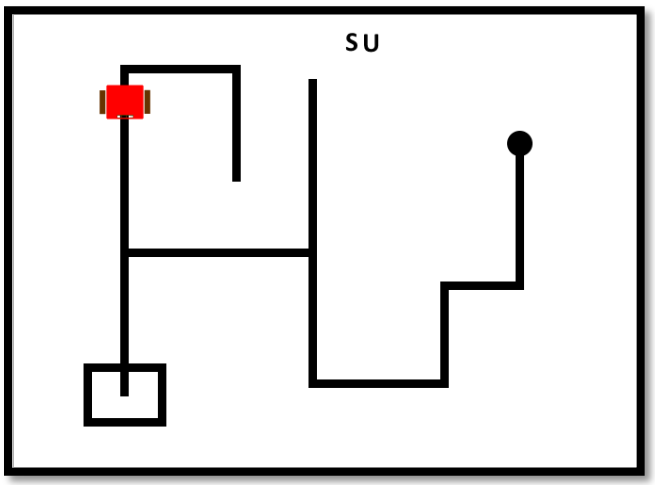
Como no puede girar a la izquierda, el robot guarda en la memoria la letra S que significa cruce pero sin giro y el robot continua en línea recta hasta que llega a un giro obligado a la derecha y gira en esa dirección sin mayor problema, como después nuevamente llega a un giro obligado a la derecha, repite la acción.



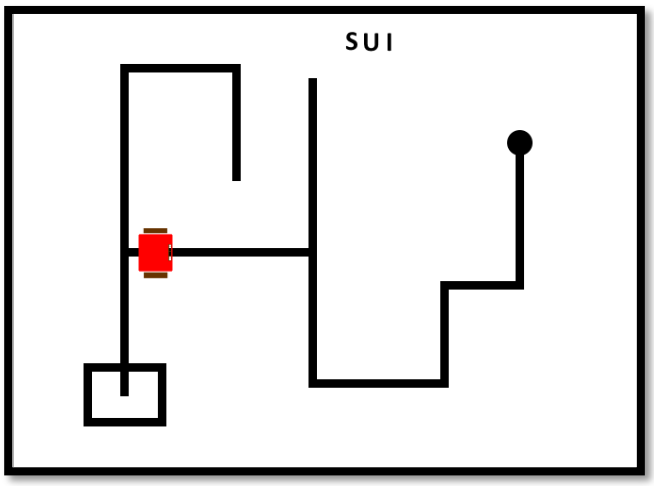
Al llegar a una línea sin fin, el robot guarda en la memoria una U que significa giro 180 grados y realiza la acción.



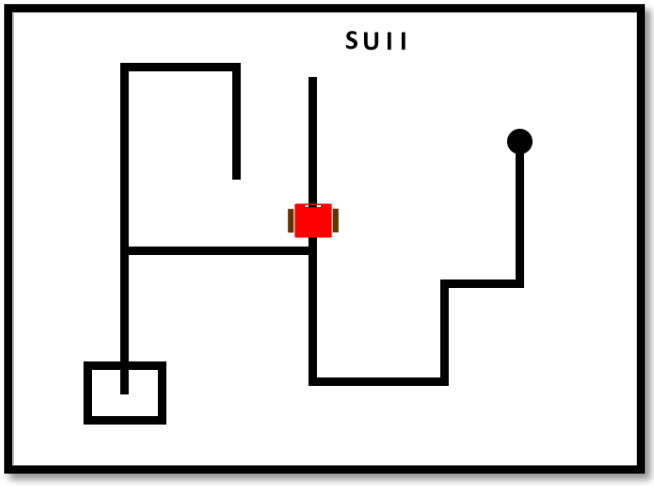
Como se puede ver en la siguiente figura, se encuentra con dos giros obligados a la izquierda, la cual realiza la acción sin mayo complejidad.



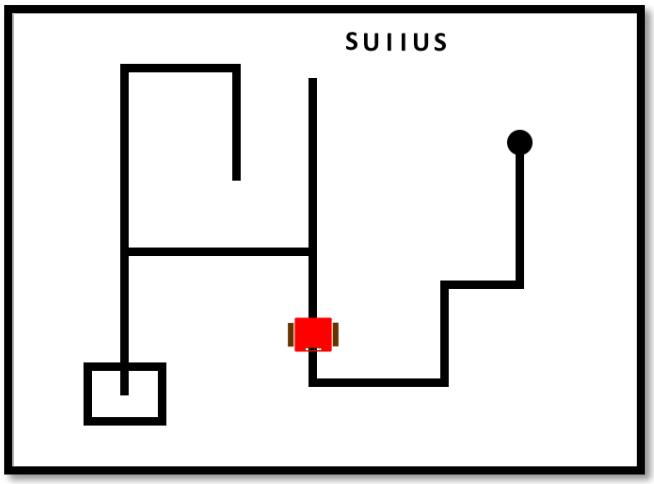
Cuando llega al cruce, el robot se da cuenta que si puede girar a la izquierda, por lo tanto realiza la acción y como es un cruce guarda en la memoria la letra I que significa giro izquierda en cruce.



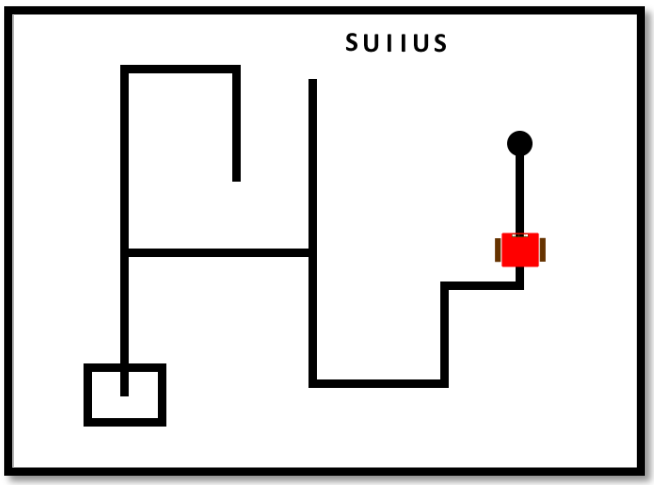
Al llegar al próximo cruce, puede girar a la izquierda y guarda en memoria otra I y realiza la acción.



Posteriormente gira 180 y guarda en memoria una U y sigue avanzando hasta llegar al cruce sin poder girar a la izquierda, sigue avanzando y guarda una S.



Las siguientes acciones son 2 giros obligados a la izquierda, un giro obligado a la derecha y luego otro giro obligado a la izquierda sin guardad nada en la memoria.



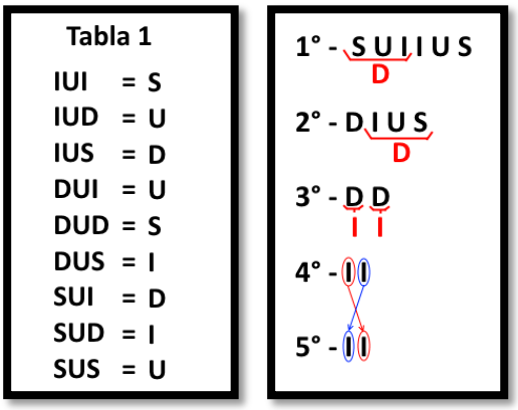
Cuando llega al final del laberinto, gira 180 sin guardar nada en la memoria y empieza a reducir los datos guardados siguiendo la tabla 1.

El primer paso es buscar de izquierda a derecha la primera U, si existe reemplaza su valor anterior con dicha U y su valor posterior por una D como lo señala la tabla; quedando más reducido los datos.

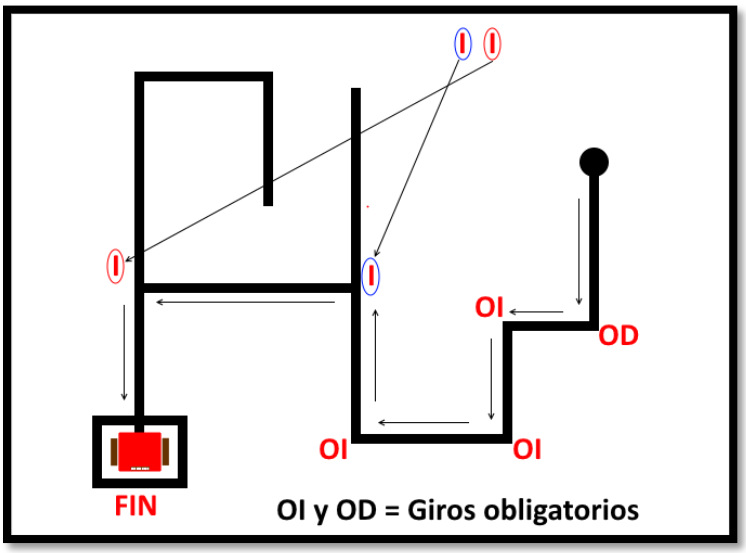
Luego se hace lo mismo con otra U que venga después y se reemplaza por una D.

Después se reemplaza las I por D y las D por I

Y finalmente se ordena inversamente los datos teniendo como resultado valores sin U y muy reducido.



Cuando tengamos listo los cambios necesarios el robot comenzara a seguir esas instrucciones sin usar la regla de la mano izquierda, pero respetando los giros obligatorios; asi podrá llegar al punto donde comenzó pero por el camino más corto.



8.2 Código laberinto (Laberinto.ino).

```
#include <Arduino.h>
#include <EEPROMex.h>
int n;

int valor_bateria;
long sensors_average;
int sensors_sum;
long sensors[] = {0, 0, 0, 0, 0, 0, 0};
float position;
float proportional;
float integral;
float derivative;
float last_proportional;
int Set_point=3000;
float error_value;
float Kp=0.40;
float Ki=0;
float Kd=4.30;
float right_speed;
float left_speed;
//unsigned char dir;
unsigned char path[300];
unsigned int path_length=0;
int dato[300];
int dato2[300];

int max_speed=180; //90 es perfect
int vel_giro=150;

void setup(){
//  while(digitalRead(2));
  Serial.begin(115200);
  Serial.setTimeout(50);
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(2,INPUT);
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  Serial.println(bateria());
  while(digitalRead(2));
  delay(500);
}



int bateria(){
  valor_bateria=(map(analogRead(7),0,1023,0,500));
  if(valor_bateria<310){
    delay(200);
    if(valor_bateria<310){
      motores(0,0);
      while(1);
    }
  }
  return valor_bateria;
}

void turn(int dir){
  switch(dir){
    case 'I': // Turn left.
                        //Serial.print("I");
                        motores(-180,-180);
                        delay(100);
                        while(analogRead(0)>500){
                        motores(-180,180);
                        }
                        motores(180,-180);
                        delay(50);
                        while(analogRead(3)>500){
                        motores(80,-80);
                        }
                        estabilizar();
                        motores(0,0);
                        //while(1);


    break;
    case 'D': // Turn right.
                        //Serial.print("D");
                        motores(-180,-180);
                        delay(100);
```

```
                    while(analogRead(6)>500){
                    motores(180,-180);
                    }
                    motores(-180,180);
                    delay(50);
                    while(analogRead(3)>500){
                    motores(-80,80);
                    }
                    estabilizar();
                    motores(0,0);
                    //while(1);
    break;
    case 'U': // Turn around.
                      //Serial.print("U");
                    motores(-180,-180);
                    delay(100);
                    while(analogRead(6)>500){
                    motores(180,-180);
                    }
                    motores(-180,180);
                    delay(100);
                    while(analogRead(3)>500){
                    motores(-80,80);
                    }
                    estabilizar();
                    motores(0,0);
                    //while(1);

    break;
    case 'S': // Don't do anything!
      //Serial.print("S");

    break;
  }
}


int select_turn( int found_left,int found_straight, int found_right){
  if(found_left)return 'I';
  else if(found_straight)return 'S';
  else if(found_right)return 'D';
  else return 'U';
}

void leer_sensores(){
  sensors_average = 0;
  sensors_sum = 0;
  for (int i = 0; i < 7; i++){
    sensors[i] = analogRead(i);
    sensors_average += sensors[i] * i * 1000;
    sensors_sum += int(sensors[i]);
  }
}

void motores(int pwm1, int pwm2){
  int freno=1;
  if(pwm1>255){pwm1=255;}
  if(pwm1<-255){pwm1=-255;}
  if(pwm2>255){pwm2=255;}
  if(pwm2<-255){pwm2=-255;}
  if(pwm1>0){
    digitalWrite(9,LOW);
    analogWrite(6,pwm1);
  }else if(pwm1<0){
    digitalWrite(6,LOW);
    analogWrite(9,-pwm1);
  }else{
    digitalWrite(6,freno);
    digitalWrite(9,freno);
  }
  if(pwm2>0){
    digitalWrite(3,LOW);
    analogWrite(5,pwm2);
  }else if(pwm2<0){
    digitalWrite(5,LOW);
    analogWrite(3,-pwm2);
  }else{
    digitalWrite(5,freno);
    digitalWrite(3,freno);
  }
}

void estabilizar(){
      int tiempo=millis();
  while(1){
```

```
    leer_sensores();
    position = int(sensors_average / sensors_sum);
    proportional = position-Set_point;
    integral = integral + proportional;
    derivative = proportional - last_proportional;
    last_proportional = proportional;
    error_value = int(proportional * 0.2 + ((integral * Ki)/100) + derivative * 0.2);
    if (error_value< -255){
      error_value = -255;
    }
    if (error_value> 255){
      error_value = 255;
    }
    if (error_value < 0){
      right_speed = -error_value;
      left_speed = error_value;
    }else{
      right_speed = -error_value;
      left_speed = error_value;
    }
    motores(right_speed, left_speed);
                  if((millis()-tiempo)>200){
                   break;
                  }

  }
}

void sigue_lineas(){
  while(1){
    leer_sensores();
    position = int(sensors_average / sensors_sum);
    proportional = position-Set_point;
    integral = integral + proportional;
    derivative = proportional - last_proportional;
    last_proportional = proportional;
    error_value = int(proportional * Kp + ((integral * Ki)/100) + derivative * Kd);
    if (error_value< -255){
      error_value = -255;
    }
    if (error_value> 255){
      error_value = 255;
    }
    if (error_value < 0){
      right_speed = max_speed ;
      left_speed = max_speed + error_value;
    }else{
      right_speed = max_speed - error_value;
      left_speed = max_speed;
    }
    motores(right_speed, left_speed);
                  if(sensors[1] > 700 && sensors[3] > 700 && sensors[5] > 700){
      return;
    }
    if(sensors[0] < 350 || sensors[6] < 350){
      return;
    }


  }
}

void loop(){
      while(0){
        if ( Serial.available()) // Check to see if at least one character is available
        {int k=Serial.read();
             switch (k){
               case 'A':
                   Kp=Kp + 0.1;
               break;
               case 'a':
                   Kp=Kp + 0.01;
               break;
               case 'B':
                   Kp=Kp - 0.01;
               break;
               case 'b':
                   Kp=Kp - 0.1;
               break;
               case 'C':
                   Kd=Kd + 0.01;
               break;
               case 'c':
                   Kd=Kd + 0.1;
               break;
               case 'D':
```

```
                    Kd=Kd - 0.01;
                  break;
                  case 'd':
                    Kd=Kd - 0.1;
                  break;
                  case 'R':
                    break;
                  break;


                }
                Serial.print("Kp= ");
                Serial.println(Kp);
                Serial.print("Kd= ");
                Serial.println(Kd);

        }

        sigue_lineas();

       }
       //Serial.println("PRIMER LOOP");
      while(analogRead(6)>500){
        motores(70,70);
       }
       while(analogRead(6)<500){
         motores(90,90);
       }
       motores(0,0);
       //delay(1000);


       while(1){
     sigue_lineas();
     motores(180,180);
     //delay(15);
     int found_left=0;
     int found_straight=0;
     int found_right=0;
     leer_sensores();
     if(sensors[0] < 700){found_left=1;}
     if(sensors[6] < 700){found_right=1;}
     motores(80,80);
     delay(40);
     leer_sensores();
     if(sensors[1] < 700 || sensors[3] < 700 || sensors[5] < 700){found_straight = 1;}
     if(sensors[0] < 900 && sensors[2] < 900 && sensors[6] < 900){break;}
     int dir = select_turn(found_left, found_straight, found_right);
     turn(dir);
     path[path_length] = dir;
     path_length ++;
     //simplify_path();

   }
        motores(-180,-180);
        delay(100);
        motores(0,0);

        //Serial.println();
        //Serial.println("FIN DEL PRIMER LOOP");
        //Serial.println("FIN DEL PRIMER LOOP");
        int x=0;

        for(x=0;x<path_length;x++){
          // Serial.print(" ");
          // Serial.write(path[x]);
         //  delay(50);
        }
        //Serial.println();
        //Serial.println(x);
        //Serial.println(path_length);

        for(int r=0;r<300;r++){
          dato[r]=path[r];

        }


    int L=1;
 while(L){
   n=2;
   L=0;
   while(1){
     if((dato[n-1]=='U')){
       if((dato[n-2]=='I')&&(dato[n]=='I')){dato[n-2]='S';}
       else if((dato[n-2]=='I')&&(dato[n]=='D')){dato[n-2]='U';}
```

```
      else if((dato[n-2]=='I')&&(dato[n]=='S')){dato[n-2]='D';}
      else if((dato[n-2]=='D')&&(dato[n]=='I')){dato[n-2]='U';}
      else if((dato[n-2]=='D')&&(dato[n]=='D')){dato[n-2]='S';}
      else if((dato[n-2]=='D')&&(dato[n]=='S')){dato[n-2]='I';}
      else if((dato[n-2]=='S')&&(dato[n]=='I')){dato[n-2]='D';}
      else if((dato[n-2]=='S')&&(dato[n]=='D')){dato[n-2]='I';}
      else if((dato[n-2]=='S')&&(dato[n]=='S')){dato[n-2]='U';}
    for(int m=n;m<=path_length;m++){
      dato[m-1]=dato[m+1];
    }
    path_length=path_length-2;
    L=1;
    break;
    }
  n++;
  if(n>path_length)break;
  }
}

for(int x=0;x<=path_length;x++){
//  Serial.write(dato[x]);
//  Serial.print(" ");
 }

//  Serial.println();

 for(int x=0;x<=path_length;x++){
   if(dato[x]=='I'){
     dato2[path_length-(x+1)]='D';
  }else if(dato[x]=='D'){
     dato2[path_length-(x+1)]='I';
  }else{
    dato2[path_length-(x+1)]=dato[x];
  }
 }


for(int x=0;x<path_length;x++){
//  Serial.write(dato2[x]);
//  Serial.print(" ");
 }
// Serial.println();

 while(analogRead(6)<900){
     motores(180,-180);
 }
 turn('U');

       //while(digitalRead(2));
       //delay(1000);
       //Serial.println("SEGUNDO LOOP");
       while(1){
       int i;
       for(i=0;i<=path_length;i++){
         sigue_lineas();
         while((analogRead(0)<500)||(analogRead(6)<500)){
            motores(90,90);
         }
         motores(0,0);
         turn(dato2[i]);
         //delay(1000);

       }
       //sigue_lineas();
       break;
       }
       while(analogRead(6)<500){
       motores(70,70);
      }


      sigue_lineas();
      motores(-180,-180);
      delay(100);
      motores(0,0);
       // Serial.print("FIN SEGUNDO LOOP");
        while(1);
}
```

9 Bibliografía

Guía de usuario Pololu 3 pi:

http://www.pololu.com/file/0J137/Pololu3piRobotGuiaDeUsuario.pdf

Como leer varios sensores infrarrojos con un solo pin:

http://www.sistemasorp.es/2011/10/25/leer-multiples-sensores-de-infrarrojos-con-un-solo-pin-analogico/

Algoritmo laberinto:

http://www.pololu.com/file/0J195/line-maze-algorithm.pdf

Control PID:

http://es.wikipedia.org/wiki/Proporcional_integral_derivativo

Arduino:

http://www.arduino.cc/es/pmwiki.php?n