[[1]](#footnote-1)

Robot Seguidor de Línea con ATMega328 arduino

Ciro Cristian Ferney1, Sepúlveda Mario Alejandro2.

*Abstract*—la siguiente guía, muestra la construcción de un robot seguidor de línea implementando el microcontrolador de 8 bits de arduino ATMega328 con implementación de un control PID (proporcional, integral y derivativo).

*Index Terms*—control PID, proporcional, derivativo, integral, microcontrolador, bits, etc.

# INTRODUCTION

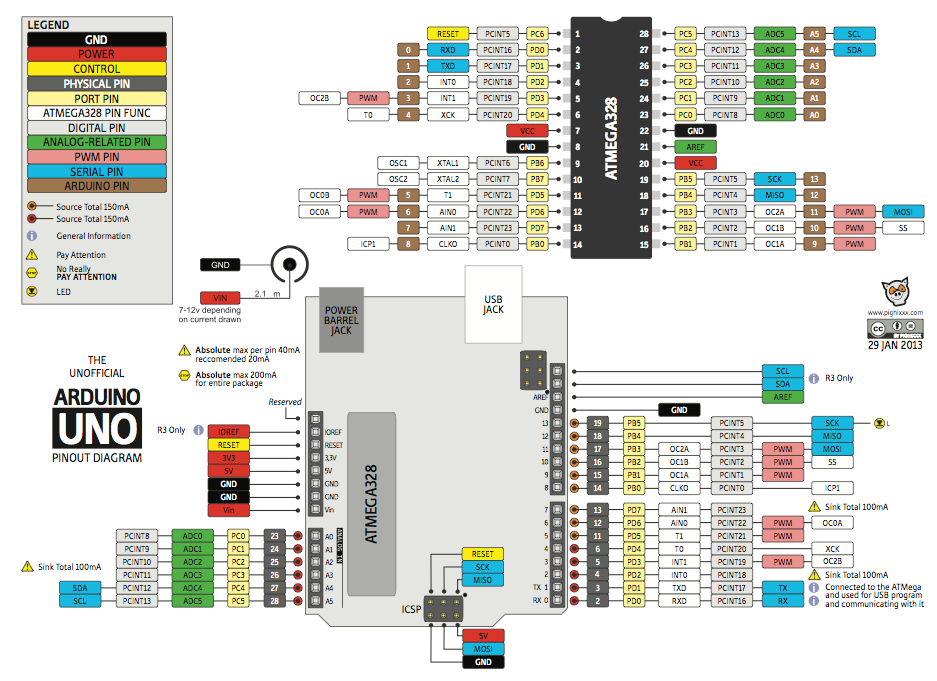
El microcontrolador ATMega328 posee una CPU AVR-8 bits, está basado en la arquitectura RISC, cuenta con una memoria Flash de 32Kb, con capacidad de leer y escribir al mismo tiempo, la memoria EEPROM de 1Kb, 2 Kb de SRAM, 23 líneas de E/S de propósito general, 32 registros de proceso general, tres temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interface serial orientada a byte de 2 cables, SPI puerto serial, 6-canales 10-bit Conversor A/D (8-chanales en TQFP y QFN/MLF packages), "watchdog timer" programable con oscilador interno, y cinco modos de ahorro de energía seleccionables por software. El dispositivo opera entre 1.8 y 5.5 voltios. Por medio de la ejecución de poderosas instrucciones en un solo ciclo de reloj, el dispositivo alcanza una respuesta de 1 MIPS, balanceando consumo de energía y velocidad de proceso.

# materiales

Los materiales usados en la construcción del robot seguidor de línea se eligieron de forma tal que haya un equilibrio entre velocidad, fuerza y flexibilidad.

## Arduino uno (ATMega328p)

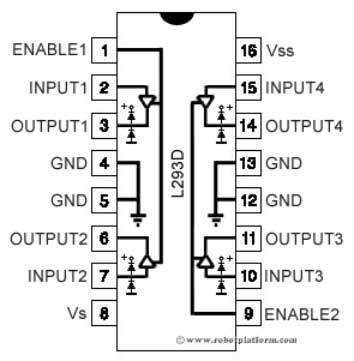
Es una tarjeta basada en el ATMega328p, el cual posee 14 pines digitales que se pueden usar como entrada o salida, 6 entradas análogas, 6 salidas PWM, trabaja con un reloj de 16MHertz y su alimentación esta entre 7V hasta 12V.

fig. 1: diagrama del arduino uno.

## L293D

Es un integrado el cual posee 4 amplificadores en modo comparador. Su función básicamente es la de controlar el sentido y la velocidad del motor.

Este integrado cuenta con la capacidad de conectar 2 motores, enseguida se mostrara su respectivo diagrama.

fig. 2: distribución de pines del L2293D

Los pines 1 y 2 van conectados a las salidas PWM del arduino uno, son los encargados de controlar la velocidad de los motores.

En los pines 3, 6, 11 y 14 conectaremos los motores los cuales deben ir de la siguiente forma: un motor se conecta en los pines 3 y 6, el segundo motor se conecta en los pines 11 y 14, se pueden conectar de cualquier forma ya que el motor no presenta polaridad.

Los pines 2, 7, 10 y 15 son los encargados de controlar el movimiento de los dos motores, asi los pines 2 y 7 controlan el movimiento de un motor y los otros dos pines el 10 y el 15 controlan el movimiento del otro motor. Cabe resaltar que cada par de pines (2, 7) y (10,15) tienen que estar de la siguiente forma para su correcto funcionamiento:

Uno el alto (1 lógico o 5V) y otro en bajo (0 lógico o 0V) para dirigir el motor en un sentido y viceversa para el otro sentido.

Los dos en bajo o en alto para frenar el motor.

Los pines marcados con GND son los que deben conectarse a tierra o a 0V, el pin Vs y Vss van a 5V.

Para el proyecto solo fue necesario implementar un L293D

## CNY70

Son sensores infrarrojos que cuentan con un LED emisor (capsula azul) y un fototransistor receptor (capsula blanca o en otros caso oscura), el diagrama de su conexión se muestra a continuación.

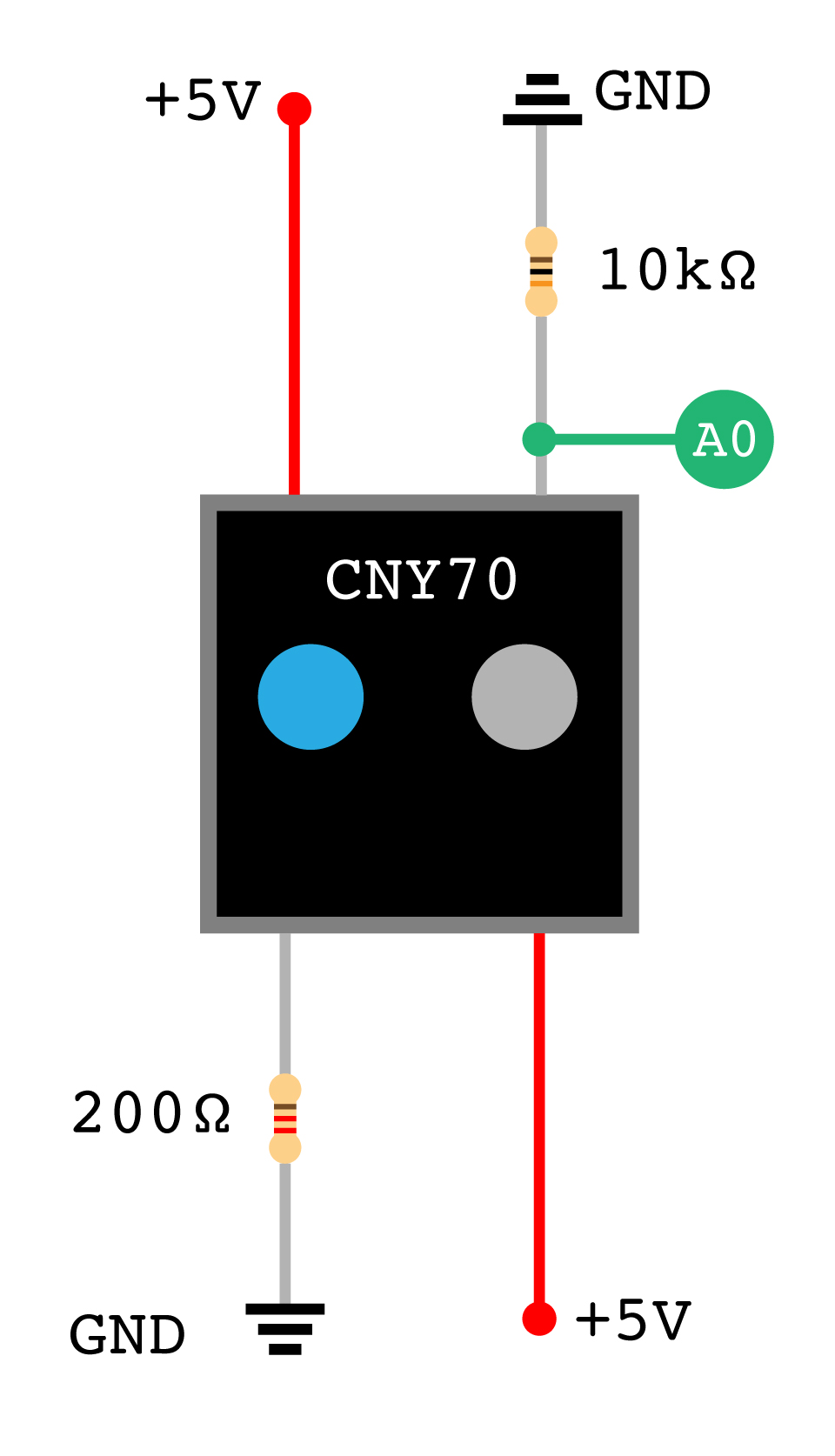
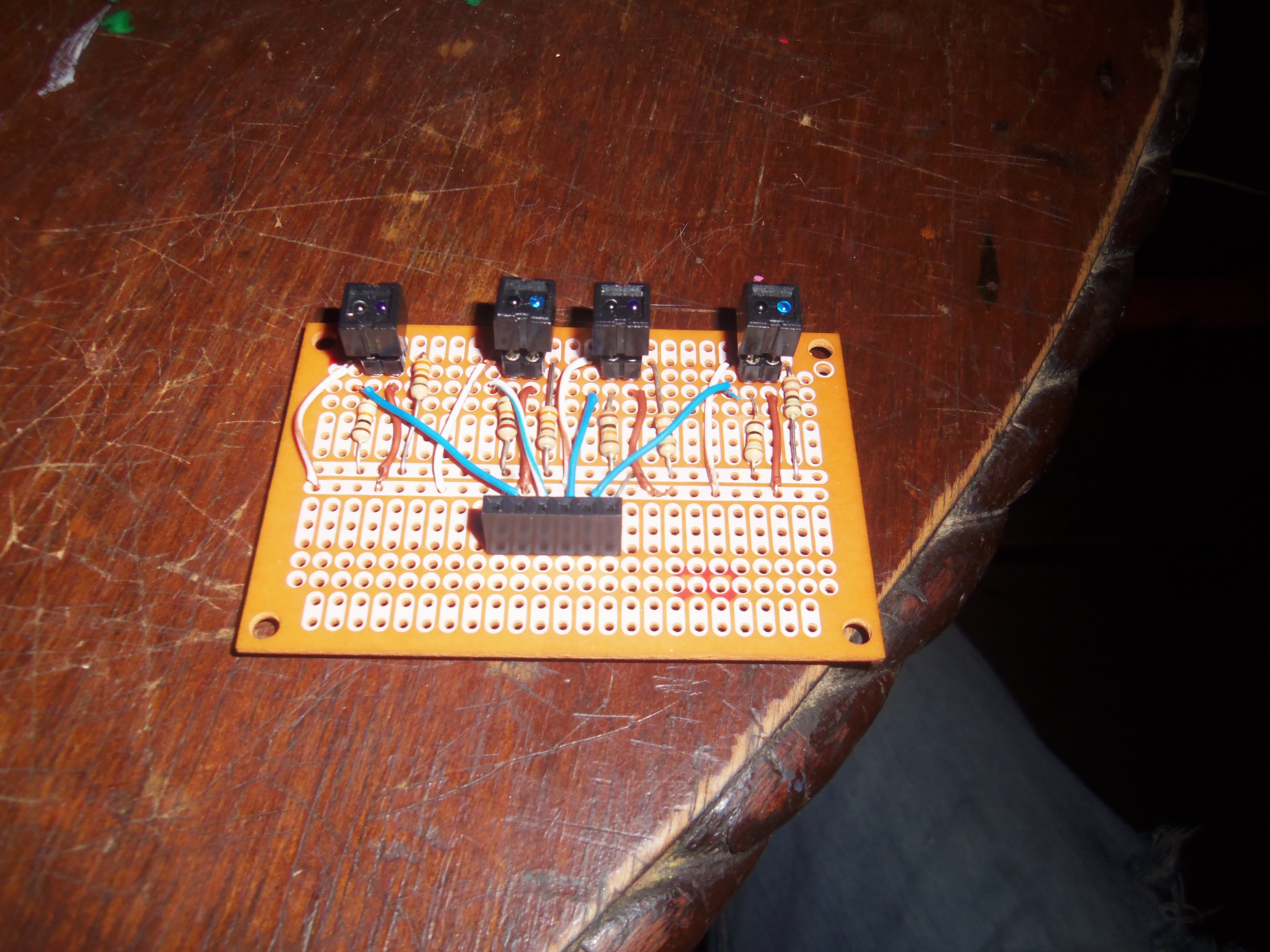


fig. 3: conexión de los sensores infrarrojos CNY70

La parte demarcada de color verde (A0) va conectado, en nuestro caso a las entradas análogas del arduino.

Para nuestro proyecto se implementaron 4 sensores infrarrojos CNY70.

fig. 4: sensores CNY70 implementados en el robot seguidor de línea

## MOTORES

Se implementaron 4 motores tipo pololu 10:1, los cuales presentan una caja reductora metálica, que genera un torque de 0.3Kg/cm, una velocidad de 3000 rpm a 6V y una corriente de consumo de 120mA sin carga, y con carga puede llegar a consumir 1600mA



fig. 5: motores pololu 10;1 usados en el robot

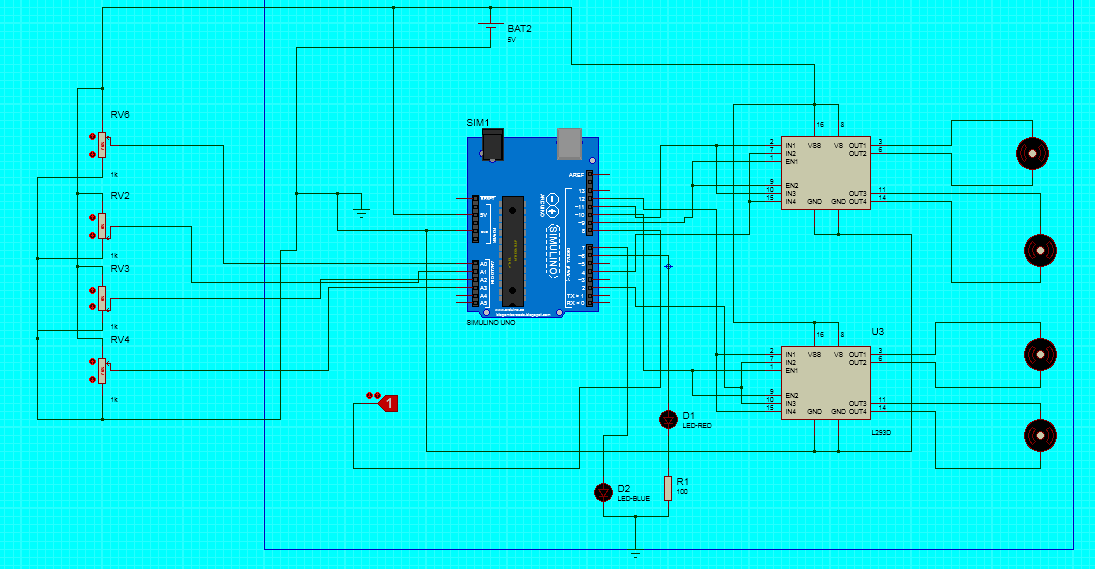
## Chasis metálico

Para la construcción de la base donde se implementarán las distintas partes fue diseñado con partes metálicas lo cual le da al robot firmeza y un buen ajuste ya que se puede desarmar y volver armar.



fig. 6: lego usado para el chasis del robot

# esquema de conexión en proteus

fig. 7: conexión en proteus

Como se puede apreciar en el diagrama anterior hemos sustituido los 4 sensores CNY70 por 4 potenciómetros los cuales presentan condiciones similares a los sensores.

# codigo

La programación usada en el robot seguidor de línea fue basada en la implementación del control PID (proporcional, integral y derivativo) el cual es un algoritmo encargado de reducir el error en la lectura de los sensores y seguir un valor de referencia.

A continuación se muestra el código implementado con sus respectivos comentarios.

// Definimos los pines digitales que usaremos

int boton = 8;

int motor\_derecha\_ad = 11; // Con este pin, dirigimos el motor derecho hacia adelante

int motor\_izquierda\_ad = 12; // Con este pin, dirigimos el motor izquierdo hacia adelante

int motor\_derecha\_at = 4; // Con este pin, dirigimos el motor derecho hacia atras

int motor\_izquierda\_at = 2; // Con este pin, dirigimos el motor izquierdo hacia atras

int pwm\_iz = 10; // pin de manejo de velocidad del motor izquierdo

int pwm\_der = 9; // Pin de manejo de la velocidad del motor derecho

int ledOn = 6;

int ledPlay = 7;

// Pines analogos

int sensor1 = 0;

int sensor2 = 1;

int sensor3 = 2;

int sensor4 = 3;

// Variables iniciadas en cero

float proporcional = 0;

float proporcional\_pasado = 0;

int posicion = 0;

int ultima\_posicion = 0;

float integral = 0;

float derivativo = 0;

float valor\_pwm = 0;

// Constantes del PID

float kp=0.20; //2,

float kd=12;

float ki=0.001;

void setup() {

delay(2000); // Hacemos una espera de 2 segundos

Serial.begin(9600); // iniciamos comunicación serial con 9600 bouds

// Configuramos los pines como entrada o salida

pinMode(boton,INPUT\_PULLUP);

pinMode(motor\_derecha\_ad,OUTPUT);

pinMode(motor\_izquierda\_ad,OUTPUT);

pinMode(motor\_derecha\_at,OUTPUT);

pinMode(motor\_izquierda\_at,OUTPUT);

pinMode(ledOn,OUTPUT);

pinMode(ledPlay,OUTPUT);

pinMode(pwm\_iz,OUTPUT);

pinMode(pwm\_der,OUTPUT);

int b = digitalRead(boton); // Leemos el valor en el pin boton (8)

// Ciclo de espera del pulso del boton de inicio

while(b==1){

b = digitalRead(boton);

digitalWrite(ledOn,255);

delay(300);

digitalWrite(ledOn,0);

delay(300);

}

}

void loop() {

digitalWrite(ledPlay,HIGH); // Led que indica inicio de lectura de sensores

// Lectura de los sensores en las entradas analogas

int s1 = analogRead(sensor1);

int s2 = analogRead(sensor2);

int s3 = analogRead(sensor3);

int s4 = analogRead(sensor4);

int s11 = 0; // Variables usadas para la conversión

int s12 = 0;

int s13 = 0;

int s14 = 0;

// Calibración de sensores para convertirlos a binario

if(s1>150){

s11=1;

}

if(s2>180){

s12=1;

}

if(s3>200){

s13=1;

}

if(s4>200){

s14=1;

}

posicion = (1\*s11)+(2\*s12)+(4\*s13)+(8\*s14); // Convertimos los números binarios obtenidos de los sensores en decimales

if(posicion <= 14){

ultima\_posicion = posicion; // Almacenamos las posiciones leidas

}

Serial.println(posicion); // Mostramos en pantalla el valor de la posicion leida por los sensores

// Calculamos los valores proporcional, integral y derivativo

proporcional = (posicion - 9);

integral = (integral + proporcional\_pasado);

derivativo = (proporcional - proporcional\_pasado);

// Valor obtenidos del PID

valor\_pwm = (kp\*proporcional)+(ki\*integral)+(kd\*derivativo);

//Serial.println(valor\_pwm); // Mostramos el valor del PID obtenido

// Limitamos el valor de la integral

if(integral>200){

integral=200;

}

if(integral<-200){

integral=-200;

}

// Almacenamos el ultimo valor de proporcional

proporcional\_pasado = proporcional;

// Condiciones de la lectura de los sensores

if(posicion == 9){ // Si está en línea, avanza

digitalWrite(motor\_derecha\_ad,HIGH);

digitalWrite(motor\_derecha\_at,LOW);

analogWrite(pwm\_der,180);

digitalWrite(motor\_izquierda\_ad,HIGH);

digitalWrite(motor\_izquierda\_at,LOW);

analogWrite(pwm\_iz,180);

}

if(posicion == 8){

digitalWrite(motor\_derecha\_ad,HIGH);

digitalWrite(motor\_derecha\_at,LOW);

analogWrite(pwm\_der,250);

digitalWrite(motor\_izquierda\_ad,LOW);

digitalWrite(motor\_izquierda\_at,HIGH);

analogWrite(pwm\_iz,120+valor\_pwm);

}

if((posicion > 11 && posicion < 13)||(posicion > 13 && posicion < 15) || posicion < 8){ // Si los sensores estan en otra posicion

if(valor\_pwm < 0){ // Gira para la derecha

digitalWrite(motor\_derecha\_ad,LOW);

digitalWrite(motor\_derecha\_at,HIGH);

analogWrite(pwm\_der,120+valor\_pwm);

digitalWrite(motor\_izquierda\_ad,HIGH);

digitalWrite(motor\_izquierda\_at,LOW);

analogWrite(pwm\_iz,250);

}

if(valor\_pwm > 0){ // Gira para la izquierda

digitalWrite(motor\_derecha\_ad,HIGH);

digitalWrite(motor\_derecha\_at,LOW);

analogWrite(pwm\_der,250);

digitalWrite(motor\_izquierda\_ad,LOW);

digitalWrite(motor\_izquierda\_at,HIGH);

analogWrite(pwm\_iz,120-valor\_pwm);

}

if(posicion == 0){ // si todos los sensores estan sobre la linea negra, el carro frena.

digitalWrite(motor\_derecha\_ad,LOW);

digitalWrite(motor\_derecha\_at,LOW);

analogWrite(pwm\_der,0);

digitalWrite(motor\_izquierda\_ad,LOW);

digitalWrite(motor\_izquierda\_at,LOW);

analogWrite(pwm\_iz,0);

}

}

if(posicion == 15){ // Si los sensores se salieron de la linea negra, se lee el ultimo valor de la posicion leida por estos

if(ultima\_posicion < 9){ // Gira para la derecha

digitalWrite(motor\_derecha\_ad,LOW);

digitalWrite(motor\_derecha\_at,HIGH);

analogWrite(pwm\_der,120-valor\_pwm);

digitalWrite(motor\_izquierda\_ad,HIGH);

digitalWrite(motor\_izquierda\_at,LOW);

analogWrite(pwm\_iz,250);

}

if(ultima\_posicion > 9){ // Gira para le izquierda

digitalWrite(motor\_derecha\_ad,HIGH);

digitalWrite(motor\_derecha\_at,LOW);

analogWrite(pwm\_der,250);

digitalWrite(motor\_izquierda\_ad,LOW);

digitalWrite(motor\_izquierda\_at,HIGH);

analogWrite(pwm\_iz,120-valor\_pwm);

}

}

}

# resultados

Al momento de la prueba, el robot seguidor de línea presentaba oscilaciones en su recorrido lo cual le reducía velocidad. La solución se dio en el tanteo de las constantes kp, ki y kd en donde definimos los valores que menos oscilaciones presentaba.

# Conclusiones

1. Comprendimos el funcionamiento y la importancia del control PID.
2. Probamos distintos métodos de controlar el robot y llegamos a comprobar que el control PID era el mas eficiente.

References

1. https://es.wikipedia.org/wiki/Controlador\_PID
2. <http://aprendiendofacilelectronica.blogspot.com.co/2014/12/robot-velocista-de-competencia_4.html>
3. <http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf>
4. https://www.arduino.cc/en/main/arduinoBoardUno

1. Ola que mas bn bueno [↑](#footnote-ref-1)