TRABAJO FINAL

Circuitos y Mediciones Electrónicas

Alumnos: Sofía Bacich, Agustín Iván López, Iván Morán

UNDAV – 1°C 2022

Docente: Pablo Luciano Lannes

Introducción

El objetivo de este trabajo final es el diseño y la implementación de un robot inteligente conceptualmente similar a robots como Wall-E o Cortocircuito, que pueda desplazarse de manera autónoma esquivando obstáculos mediante un análisis de su entorno, controlado por una plaqueta Arduino MEGA.

Esto será posible gracias a un sensor ultrasónico que estará implementado en la “cabeza” del robot, la cual tendrá la posibilidad de rotar hacia un lado y hacia otro para que el sensor pueda buscar una ruta posible que permita a nuestro robot desplazarse hacia adelante. El sensor ultrasónico funciona emitiendo un impulso de onda ultrasónica, la cual se verá reflejada y le permitirá medir la distancia que debe avanzar.

Para poder moverse, utilizaremos un Motor Shield Drive L293D que permitirá al Arduino controlar dos motores de DC y un servomotor. El servomotor se utilizará para poder rotar la “cabeza” del robot, la cual recordemos tiene el sensor ultrasónico; mientras que los dos motores de DC van a permitir que nuestro robot pueda avanzar, retroceder y girar hacia ambos lados. El principio de funcionamiento es que tenemos cada uno de los motores “conectados” a las ruedas de cada lado del robot, las cuales vienen incluidas con los motores.

Sin más preámbulos, comencemos con el desarrollo de este robot inteligente.

Desarrollo

Para empezar con el diseño del robot inteligente, determinamos a partir de nuestra investigación que sería necesario utilizar un L293D, el cual tiene puentes H integrados, como se puede ver en la Figura 1, para poder cambiar la dirección de la corriente aplicada a los motores y de esta manera cambiar la dirección en que los motores giran. Además, para controlar el voltaje promedio que se aplica al motor, y por extensión su velocidad, los puentes H integrados usan la modulación por ancho de pulso o PWM, mediante la cual se utiliza una fuente de voltaje de cd constante y se selecciona un voltaje para que varíe su valor promedio. El L293D en específico puede proveer una tensión de potencia para los motores de entre 4,5-24V y una corriente de 600mA por canal.

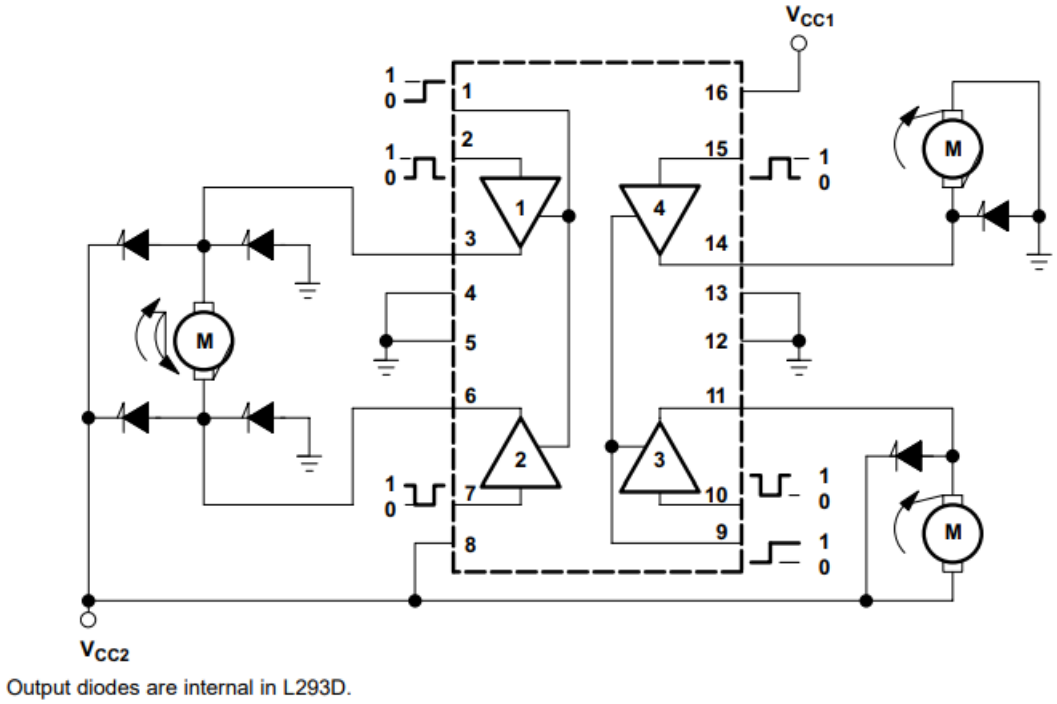

Figura 1

A partir de esto, desarrollamos el siguiente diagrama de circuito, como se puede ver en la Figura 2; con los motores de DC, el servomotor y el sensor ultrasónico conectados al L293D (y el mismo conectado al Arduino), además de la alimentación para los motores de DC. Para alimentar los motores, usamos cuatro baterías AA con características de 1,2V y 2000mAh. Esto se encargará de alimentar a los dos motores de DC, que requieren un voltaje de entre 3-6V con una corriente de carga usual de 70mA, máximo 250mA. De esta manera, nuestros 4,8V totales son suficientes para poder alimentar a los motores, y con un consumo máximo de los motores de DC tendremos una autonomía de:

$$A = \frac{2000mAh}{250mA * 2} = \frac{2000mAh}{500mA} \rightarrow \boxed{A = 4h}$$

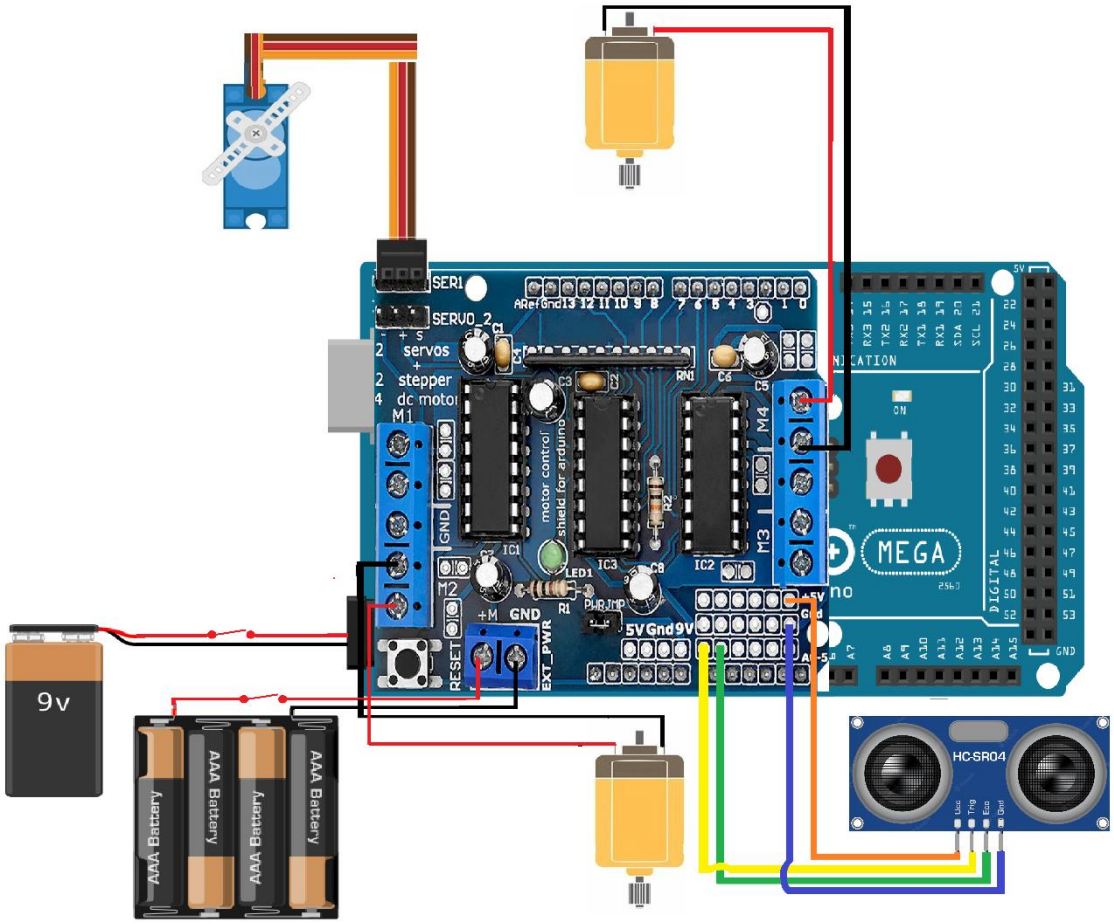

Figura 2

En cambio, para alimentar el Arduino, como el L293D está compuesto por transistores bipolares que puedan drenar el voltaje aplicado para los motores, lo tuvimos que alimentar de forma separada con una batería con características de 9V y 600mAh. El Arduino Mega puede operar a 5V, pero se recomienda usar entre 7-12V de tensión, por lo que la batería de 9V es más que adecuada. El Arduino además consume 20mA por cada pin; el L293D requiere 5V y 60mA para su control; el servomotor requiere un voltaje entre 4-7,2V con una corriente de carga de 650mA; y el sensor requiere un voltaje de 5V con una corriente de operación de 15mA. De esta manera, nuestros 9V son suficientes para poder alimentar todos estos componentes, ya que el Arduino les proveerá de 5V a todos ellos. Sin embargo, la autonomía de este robot con una sola batería de 9V y 600mAh será de:

$$A = \frac{600mAh}{60mA + 15mA + 650mA + 20mA * 4} = \frac{600mAh}{805mA} \rightarrow \boxed{A \cong 0{,}745h \cong 45m}$$

A partir del diagrama de circuito de la Figura 2, comenzamos a montar la base de nuestro robot, con los dos motores de DC atornillados a cada lado en la parte trasera de la base, cada uno manejando una rueda, y con la ruedita de giro atornillada en la parte delantera de la base. El circuito equivalente de un motor de DC se puede ver en la Figura 3 (es importante mencionar que este circuito es válido para poder mostrar cómo funciona el motor, pero que en la práctica se debe agregar un diodo en paralelo al motor para evitar que la f.e.m. inducida pueda dañar el transistor que lo controla; incluso se podría usar un optoacoplador para evitar que se dañe el microcontrolador); su principio de funcionamiento es una espiral de alambre que gira de manera libre en medio del campo de un imán permanente. Cuando por el

devanado pasa una corriente, las fuerzas resultantes ejercidas en sus lados y en ángulo recto al campo provocan fuerzas que actúan a cada lado produciendo una rotación. Sin embargo, para que este continúe, cuando el devanado pasa por la posición vertical se debe invertir la dirección de la corriente.

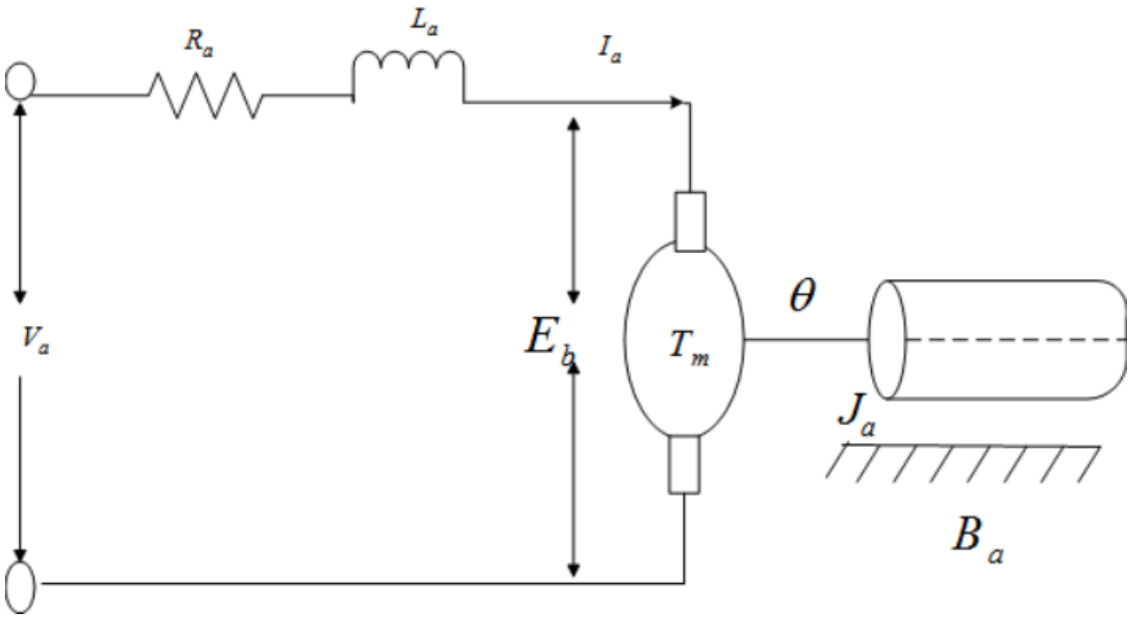

Figura 3

En un motor de DC convencional, como se puede ver en la Figura 4, los devanados de alambre se montan en las ranuras de cilindro de material magnético conocido como armadura, la cual está montada en cojinetes (en el campo magnético producido por los polos de campo, que pueden ser imanes permanentes o electroimanes) y puede girar. Conforme la armadura gira, se debe invertir la corriente de cada uno de los devanados al desplazarse por los polos de campo, para así poder asegurar que las fuerzas que actúan en el devanado sigan actuando en la misma dirección y la rotación continúe. La dirección del motor de DC se invierte al invertir la corriente de armadura o la corriente de campo.

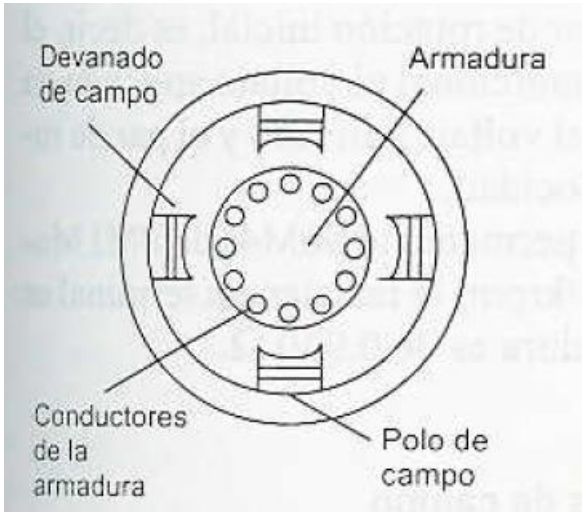

Figura 4

Luego, realizamos las conexiones con el Arduino, de acuerdo a lo explicitado en el diagrama del circuito que se puede ver en la Figura 2. Y finalmente, pegamos el servomotor sobre la

parte delantera de la base y le pegamos sobre su hélice el sensor ultrasónico, lo cual estéticamente hace parecer que el servomotor es el “cuello” del robot y que el sensor es su “cabeza”. En la Figura 5 se puede ver el circuito de un servomotor, el cual permite tener un control preciso de la posición lineal o angular, de su velocidad y de su aceleración. Sin embargo, a diferencia del motor de DC, los servomotores sólo pueden rotar 180°. Así como los puentes H, los servomotores realizan este control mediante un PWM, donde un pulso mínimo alinea la posición del servo a la izquierda, es decir, a 0°; un pulso máximo alinea la posición del servo hacia la derecha, es decir, a 180°; y los distintos valores entre el pulso mínimo y el máximo determinarán la posición angular que tendrá el servo entre 0° y 180°.

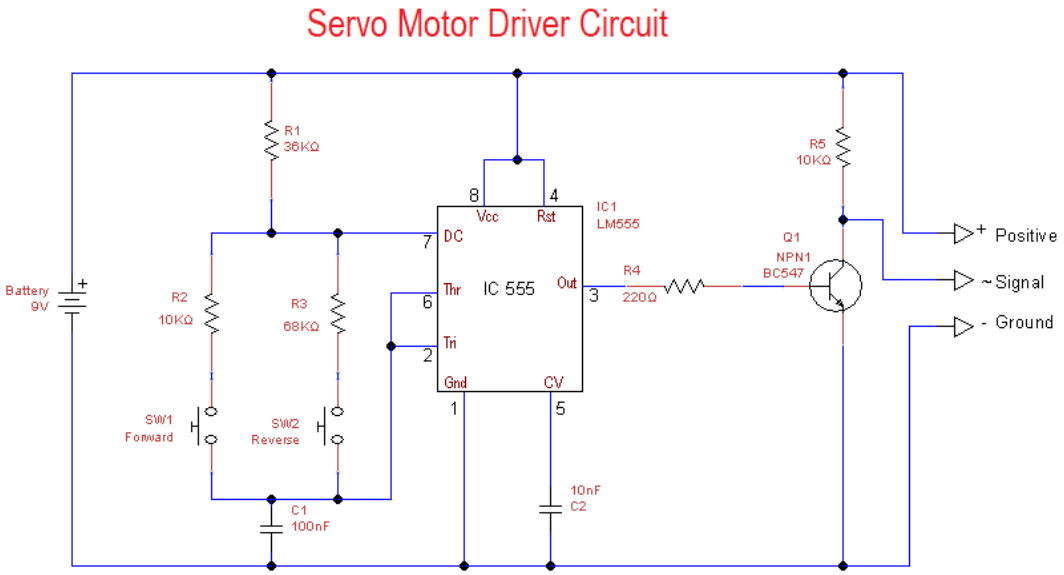

Figura 5

El sensor ultrasónico HC-SR04, mientras tanto, es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto en un rango de 2 a 450 cm. El sensor posee dos transductores: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación. El funcionamiento del sensor es el siguiente: el emisor piezoeléctrico emite 8 pulsos de ultrasonido(40KHz) luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebotan al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a Alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y así se puede calcular la distancia al objeto. El funcionamiento del sensor no se ve afectado por la luz solar o material de color negro (aunque los materiales blandos acústicamente como tela o lana pueden llegar a ser difíciles de detectar). La distancia medida por el sensor se puede calcular utilizando la siguiente fórmula:

$$\boxed{d = \frac{t * v_s}{2}}$$

Donde d es la distancia en metros, t es el tiempo del pulso ECO y v_s es la velocidad del sonido, equivalente a $340m/s$. Vale la pena mencionar que nosotros usamos una librería de Arduino que nos resuelve el problema de tener que realizar este cálculo, pero si quisiéramos determinar la distancia en centímetros a partir de pulsos que duran microsegundos, deberíamos tomar la siguiente fórmula:

$$\frac{2d}{t} = v_s = \frac{340m}{s} * \frac{1s}{1000000\mu s} * \frac{100cm}{1m} \cong \frac{2cm}{59\mu s} \rightarrow \boxed{d(cm) = \frac{t(\mu s)}{59}}$$

Una vez terminado el diseño del robot, es hora de abrir el Arduino IDE y empezar a codear el programa que va a manejar el comportamiento de nuestro robot inteligente. Primero que nada, como podemos ver en la Figura 6, necesitaremos instalar e importar tres librarías: AFMotor de Adafruit Motor Shield V1, la cual permite controlar el L293D; NewPing, una librería muy útil para manejar sensores ultrasónicos como el HC-SR04; y Servo, para poder manejar el servomotor. Luego, definimos una serie de constantes globales que hacen referencia a los pines del sensor ultrasónico, a la distancia máxima que va a ser medida, y a la velocidad máxima a la que pueden girar los motores. A continuación, definimos las variables que hacen referencia a los componentes electrónicos que vamos a usar: el sensor ultrasónico, los motores (uno para la izquierda y el otro para la derecha) y el servomotor. También definimos tres variables de control: el booleano goesForward que nos dirá si el robot está actualmente explorando hacia adelante y permitirá controlar su movimiento; el entero distance, que nos dirá la distancia leída por el sensor ultrasónico; y el entero speedSet que se usará para controlar la velocidad de los motores.

```
#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>
#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200
#define MAX_SPEED 190

NewPing sensor(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

AF_DCMotor motorL(2, MOTOR12_1KHZ); //motor Left
AF_DCMotor motorR(4, MOTOR34_1KHZ); //motor Right
Servo servo;

boolean goesForward=false;
int distance = 100;
int speedSet = 0;

void setup() {

  servo.attach(10);
  servo.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}
```

Figura 6

Finalmente, en el setup(), le asignamos un pin al servo y le definimos que inicialmente va a estar en su posición media a 115°. Luego, vamos a hacer una serie de lecturas con el sensor ultrasónico usando la función readPing(), la cual se puede ver definida en la Figura 7. Esta función simplemente realiza una lectura de la distancia en centímetros con el sensor y controla que, si el valor devuelto es 0, es decir que no hay ningún objeto enfrente en el que hayan rebotado los pulsos ultrasónicos, entonces la distancia que puede avanzar el robot es de 250cm. También en la Figura 7 podemos encontrar definidas las funciones lookRight() y lookLeft(), las cuales permiten que el robot gire su "cabeza", a la derecha o a la izquierda respectivamente, para medir la distancia en esas direcciones. Ambas funcionan prácticamente iguales: el servo gira a su nueva posición angular para que el sensor pueda realizar su lectura y luego vuelve a su posición original, finalmente devolviendo la distancia medida. En el caso de lookRight(), el servo va a cambiar su posición a 50°; mientras que para lookLeft() va a ir a 170°.

```
int lookRight()
{
    servo.write(50);
    delay(500);
    int distance = readPing();
    delay(100);
    servo.write(115);
    return distance;
}

int lookLeft()
{
    servo.write(170);
    delay(500);
    int distance = readPing();
    delay(100);
    servo.write(115);
    return distance;
    delay(100);
}

int readPing() {
  delay(70);
  int cm = sensor.ping_cm();
  if(cm==0)
  {
    cm = 250;
  }
  return cm;
}
```

Figura 7

Luego tenemos las funciones de movimiento para controlar los motores, como se puede ver en la Figura 8, con moveStop() haciendo que el robot se detenga; moveForward() haciendo que se mueva hacia delante; y moveBackward() hacia que se mueva hacia atrás. Acá podemos ver que se utiliza la función run del motor, la cual tiene tres parámetros: RELEASE para parar, FORWARD para avanzar y BACKWARD para retroceder. En moveForward() y moveBackward(), además, vamos a ir aumentando progresivamente la velocidad mediante un bucle for.

```
void moveStop() {
  motorL.run(RELEASE);
  motorR.run(RELEASE);
  goesForward = false;
  }

void moveForward() {

 if(!goesForward)
  {
    goesForward=true;
    motorL.run(FORWARD);
    motorR.run(FORWARD);
   for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
   {
    motorL.setSpeed(speedSet);
    motorR.setSpeed(speedSet);
    delay(5);
   }
  }
}

void moveBackward() {
    goesForward=false;
    motorL.run(BACKWARD);
    motorR.run(BACKWARD);
  for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
  {
    motorL.setSpeed(speedSet);
    motorR.setSpeed(speedSet);
    delay(5);
  }
}
```

Figura 8

Por último tenemos las funciones de giro para nuestro robot, como se puede ver en la Figura 9, con turnRight() permitiendo que el robot gire a la derecha; turnLeft() permitiéndole girar a la izquierda; y turnAround() permitiéndole hacer un giro de 180° para darse vuelta. En los tres casos, uno de los motores hace girar su rueda hacia delante y el otro hacia atrás durante un cierto período de tiempo hasta que el robot se detiene y se mueve hacia delante.

```
void turnRight() {
  goesForward = false;
  motorL.run(FORWARD);
  motorR.run(BACKWARD);
  delay(500);
  moveStop();
  moveForward();
}

void turnLeft() {
  goesForward = false;
  motorL.run(BACKWARD);
  motorR.run(FORWARD);
  delay(500);
  moveStop();
  moveForward();
}

void turnAround() {
  goesForward = false;
  motorL.run(FORWARD);
  motorR.run(BACKWARD);
  delay(1000);
  moveStop();
  moveForward();
}
```

Figura 9

Con todas nuestras funciones definidas, ahora sí podemos mirar el loop(), donde tendremos el bucle que controla al robot, como se puede ver en la Figura 10. El principio de funcionamiento es muy sencillo: si la distancia registrada por el sensor ultrasónico es mayor a 30cm, entonces el robot puede avanzar; pero si es menor o igual, entonces el robot tiene que girar en otra dirección. Para eso se definen los enteros distanceR y distanceL, que hacen referencia a la distancia medida hacia la derecha y la izquierda respectivamente. El robot entonces va a detenerse, retroceder hacia atrás muy brevemente, y mirar hacia la derecha y la izquierda para medir la distancia en esas direcciones. Luego, si la distancia en ambas direcciones es menor o igual a 30cm, el robot va a darse vuelta completamente; caso contrario, va a girar hacia la dirección que tenga más distancia. Finalmente, antes de volver a comenzar el bucle, vamos a volver a medir la distancia que hay en frente del robot. Esto permite que el robot pueda desplazarse de manera inteligente y autónoma como un explorador de su entorno.

```
void loop() {
 int distanceR = 0;
 int distanceL = 0;
 delay(40);
 if(distance<=30){
  moveStop();
  delay(100);
  moveBackward();
  delay(300);
  moveStop();
  delay(200);
  distanceR = lookRight();
  delay(200);
  distanceL = lookLeft();
  delay(200);
  if(distanceR <= 30 && distanceL <= 30){
    turnAround();
    moveStop();
  } else {
    if(distanceR>=distanceL){
      turnRight();
      moveStop();
    }else{
      turnLeft();
      moveStop();
    }
  }
 }else{
  moveForward();
 }
 distance = readPing();
}
```

Figura 10

Conclusión

Como conclusión, nos gustaría decir que esta ha sido una gran experiencia, ya que algunos de nosotros no teníamos tanta experiencia en proyectos de electrónica o robótica. El robot ha salido muy bien y estamos muy contentos con la forma en que puede moverse de manera autónoma por su entorno; pero, además, hemos podido poner en práctica un montón de conceptos que fueron aprendidos a lo largo del cuatrimestre.

Eso es muy valioso para nosotros como estudiantes, ya que nos sirve como una muy buena práctica profesional si quisiéramos seguir por este camino. Sin embargo, queremos notar que, si hubiéramos tenido más tiempo y recursos, podríamos haber implementado las siguientes mejoras:

- Mejorar la autonomía del robot, especialmente para la alimentación del Arduino.
- Agregar un sistema de recarga para las baterías, mediante un cable USB.
- Mejora estética del robot para así poder ocultar el circuito interno del Arduino.
- Reemplazar el sistema de dos ruedas por una tractor oruga para que el robot pueda desplazarse más fácilmente sobre terrenos irregulares.

Además, el diseño de este robot podría ser cambiado para poder permitir un montón de funcionalidades nuevas, por ejemplo:

- Incorporarle una cámara fotográfica para poder grabar el recorrido del robot, haciendo que, en efecto, este sea un androide explorador. Los vídeos podrían ser guardados en la nube, por lo que necesitaríamos algún módulo para que el Arduino pueda conectarse mediante wifi; o bien, un dispositivo de almacenamiento interno. Para eso podríamos también incorporarle una linterna y un sensor de luz que, al detectar oscuridad, active la linterna.
- Incorporarle un sistema de aspiración y un sensor de suciedad/polvo para hacer que nuestro robot sea esencialmente una aspiradora inteligente. Obviamente también deberíamos agregarle una bolsa para guardar lo aspirado y algún tipo de sistema de sensor + alarma que avise cuando la bolsa esté llena y tenga que ser cambiada.
- Agregarle un sensor de sonido y reconocimiento de voz para poder darle instrucciones al robot.