Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bachillerato en Ingeniería en Computación

IC-3101 Arquitectura de Computadoras

Profesor: Ing. Eduardo A. Canessa Montero, M.Sc.

Proyecto Final: Primer Etapa

Integrantes:

Sara Castro Sáenz 2014085332

Ericka Céspedes Moya 2017239557

José David Martínez Garay 2018160104

Fecha de Entrega: 16 de noviembre

II Semestre, 2018

Introducción

El proyecto actual se trata del desarrollo de la propuesta hecha por cada grupo, que consta de diseñar un sistema aplicado a un ambiente real basado en una arquitectura de computador utilizando el lenguaje ensamblador. La idea elegida se trata de automatizar una aguja de tránsito y un semáforo de acuerdo a la distancia de un tren. Este tipo de tecnología es utilizado en la regulación de tránsito y es muy útil en un país como Costa Rica donde hay choques contra el tren frecuentemente. En general, el sistema planteado funcionará de la siguiente manera: un tren se acercará a cierta distancia detectada por el sensor ultrasónico, esta señal causará que el semáforo se encienda y que la aguja de tránsito baje, cuando el sensor ultrasónico ya no detecte al tren cerca, se apagará el semáforo y se subirá la aguja de tránsito.

Los materiales ocupados para el proyecto serán en un principio un Arduino UNO R3, un servo SG92R para hacer la aguja de tránsito, LEDs rojos para el semáforo, resistores 330 ohm, y un sensor ultrasónico HC-SR04 para detectar la distancia a la que viene el tren. El sensor HC-SR04 que usa tecnología sonar para detectar objetos dentro de un rango con gran precisión, es capaz de enviar señales y recibir, además no se ve afectado por la luz por lo cual es ideal para la implementación del proyecto. El servo utilizado tiene un rango de 90 grados en cada dirección, lo cual es suficiente para realizar la aguja, la cual no necesita moverse más de 90 grados para funcionar.

El trabajo escrito consta de cuatro partes referentes al primer avance del proyecto: los resultados y análisis, las conclusiones, las recomendaciones, y el apéndice con el diseño modular. Este primer avance se enfocó mayoritariamente en el diseño del proyecto, no de su implementación. En los resultados y análisis se documenta el diseño propuesto para el sistema que se desea realizar, mientras que se obtuvieron conclusiones y recomendaciones de acuerdo a estos. En el apéndice se adjunta el diseño modular del proyecto en Arduino utilizando Tinkercard y la descripción de cada módulo con un diagrama de flujo.

Resultados y análisis

La siguiente máquina de estados muestra el funcionamiento general del proyecto. En el Estado 0 el semáforo y la aguja están apagados y en el Estado 1 el semáforo y la aguja estarán encendidos. La aguja de tránsito se considera apagada cuando está en posición perpendicular y encendida cuando está en posición horizontal. El cambio del Estado 0 al Estado 1 se da cuando la señal del sensor ultrasónico se enciende; en otras palabras, cuando el sensor ultrasónico detecta al tren a una determinada distancia. El cambio del Estado 1 al Estado 0 se da cuando ocurre lo contrario de lo anterior, cuando la señal del sensor ultrasónico se apaga; en otras palabras, cuando el sensor ultrasónico no detecta al tren. El Estado 0 se mantiene en el Estado 0 cuando la señal del sensor está apagada y el Estado 1 se mantiene en el Estado 1 cuando la señal del sensor está encendida. En otros términos, mientras se detecte al tren, el semáforo permanecerá encendido y la aguja de tránsito seguirá perpendicular. Además, mientras no se detecte al tren, el semáforo se mantendrá apagado y la aguja de tránsito permanecerá horizontal.

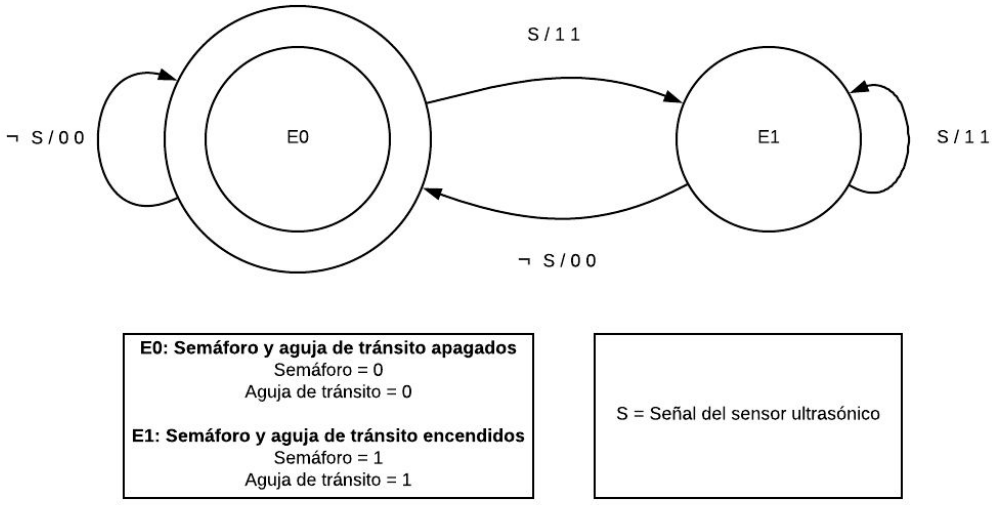

Figura 1. Máquina de estados del proyecto elegido

En el diagrama de la máquina de estados se muestra la letra S como la señal del sensor ultrasónico y las salidas 00 o 11 como el estado encedido o apagado del semáforo y la aguja de

tránsito. A continuación se muestra la tabla de transición correspondiente a la máquina de estados propuesta.

Estado Actual	Entradas	Salidas		Siguiente Estado
	Señal del sensor (S)	Semáforo	Aguja de tránsito	
E0	0	0	0	E0
	1	1	1	E1
E1	0	0	0	E0
	1	1	1	E1

Figura 2. Tabla de transición de la máquina de estados

En este primer avance del proyecto se logró implementar el proyecto propuesto en el lenguaje de programación de Arduino usando el IDE de Arduino. El lenguaje de programación Arduino se puede dividir en tres partes principales: estructura, valores (variables y constantes) y funciones. Entre las variables escritas en el código se encuentran: la distancia (distance), el echo pin (echoPin), el trig pin (triggerPin), los dos LEDs (ledOne y ledTwo), y un objeto servo de tipo Servo (para esto se incluyó la biblioteca Servo.h). El triggerPin es un pin que manda la señal del sensor y el echoPin es el que recibe la señal cuando rebota. El tiempo que hay entre lo que se transmite y se recibe es lo que se utiliza para calcular la distancia entre el sensor y un objeto. El transmisor (trig pin) envía una señal, un sonido de alta frecuencia, y por esto el sensor ultrasónico es muy susceptible a los sonidos.

El código en lenguaje de programación de Arduino se divide en tres funciones: la primera función lee la distancia con el sensor ultrasónico utilizando los pines (trigger y echo), la segunda inicializa los pines y el servo, y la tercera es el ciclio del programa. A continuación se muestran las imágenes del código escrito en tres partes.

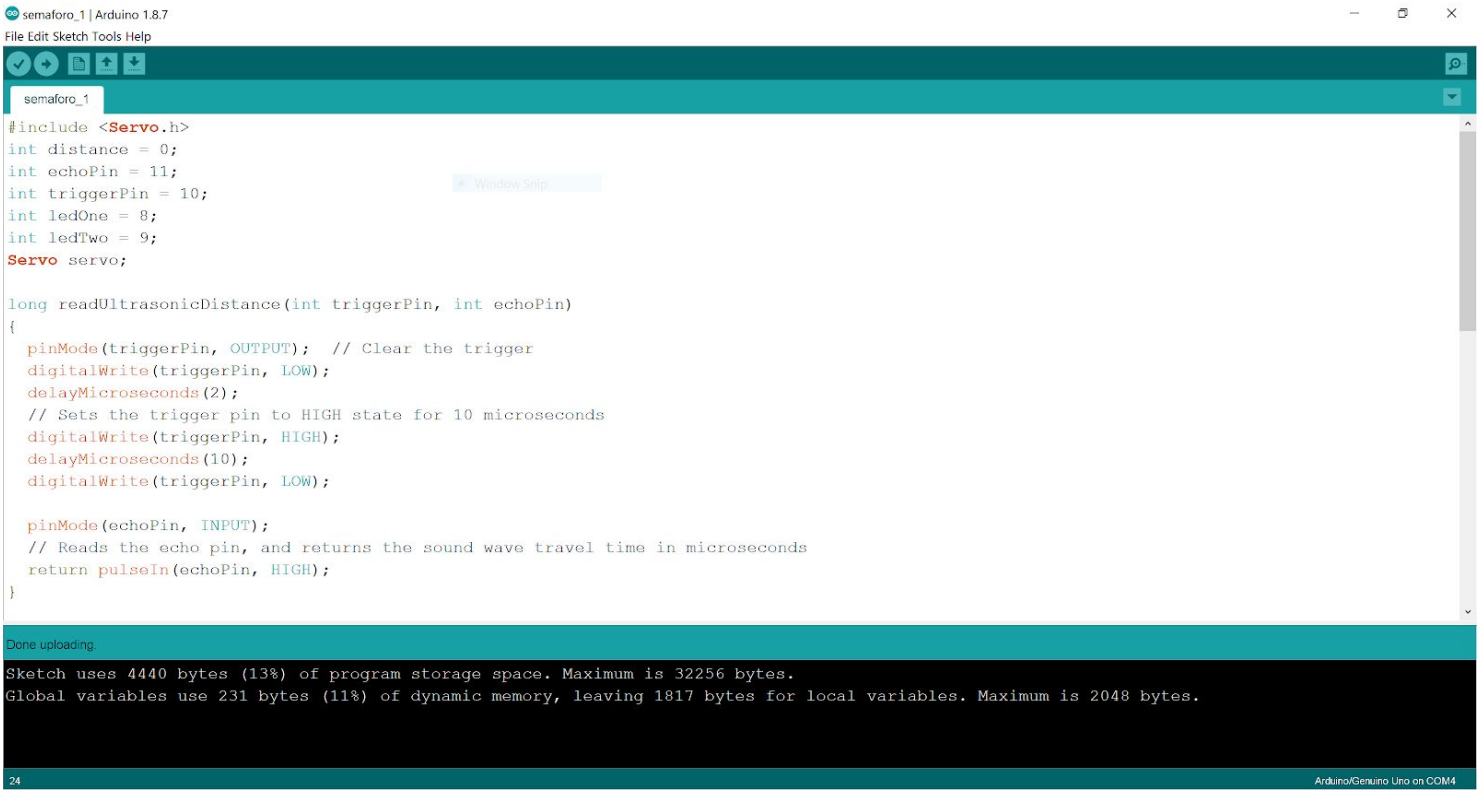

Figura 3. Primera parte del código en lenguaje de programación de Arduino

La función *readUltrasonicDistance(int triggerPin, echoPin)* lee la señal del sensor ultrasónico recibiendo las señales del los pines y devuelve la distancia. La función *setup()* inicializa los pines y el servo. La función *loop()* contiene el ciclo del programa que recibe constantemente la distancia del sensor ultrasónico de la función *readUltrasonicDistance(int triggerPin, echoPin)*, y consecuentemente mueve la hélice del servo y enciende los LEDs intermitentemente.

```
void setup()
{
  servo.attach(12);
  pinMode(ledTwo, OUTPUT);
  pinMode(ledOne, OUTPUT);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);

}
```

Figura 4. Segunda parte del código en lenguaje de programación de Arduino

La función *loop()* recibe la distancia de la función *readUltrasonicDistance(int triggerPin, echoPin)* y la multiplica por 0.01723 para convertirla a centímetros. Luego, si la distancia es menor a 20 centímetros, el servo se moverá 90 grados y los LEDs se encenderán intermitentemente cada 500 milisegundos. De lo contrario, si la distancia es mayor o igual a 20 centímetros, los LEDs estarán apagados y el servo estará en 0 grados. Esta función cuenta con un delay de 0.25 milisegundos dentro del ciclo.

```
void loop()
{

    distance = 0.01723 * readUltrasonicDistance(triggerPin, echoPin);
    Serial.println(distance);
    if (distance < 20) {
      servo.write(90);
      digitalWrite(ledOne, HIGH);
      digitalWrite(ledTwo, LOW);
      delay(500); // Wait for 500 millisecond(s)
      digitalWrite(ledOne, LOW);
      digitalWrite(ledTwo, HIGH);
      delay(500); // Wait for 500 millisecond(s)
    } else {
          digitalWrite(ledOne, LOW);
        digitalWrite(ledTwo, LOW);
        servo.write(0);
   }
    delay(0.25); // Wait for 0.25 millisecond(s)
}
```

Figura 5. Tercera parte del código en lenguaje de programación de Arduino

En un principio se usó Tinkercad para el diseño del proyecto en Arduino que permite diseñar el sistema en Arduino y simular su funcionamiento. De esta manera, como funcionaba correctamente la simulación, se implementó el mismo diseño en físico. A continuación se muestra la implementación física en Arduino derivada de la simulación hecha Tinkercad.

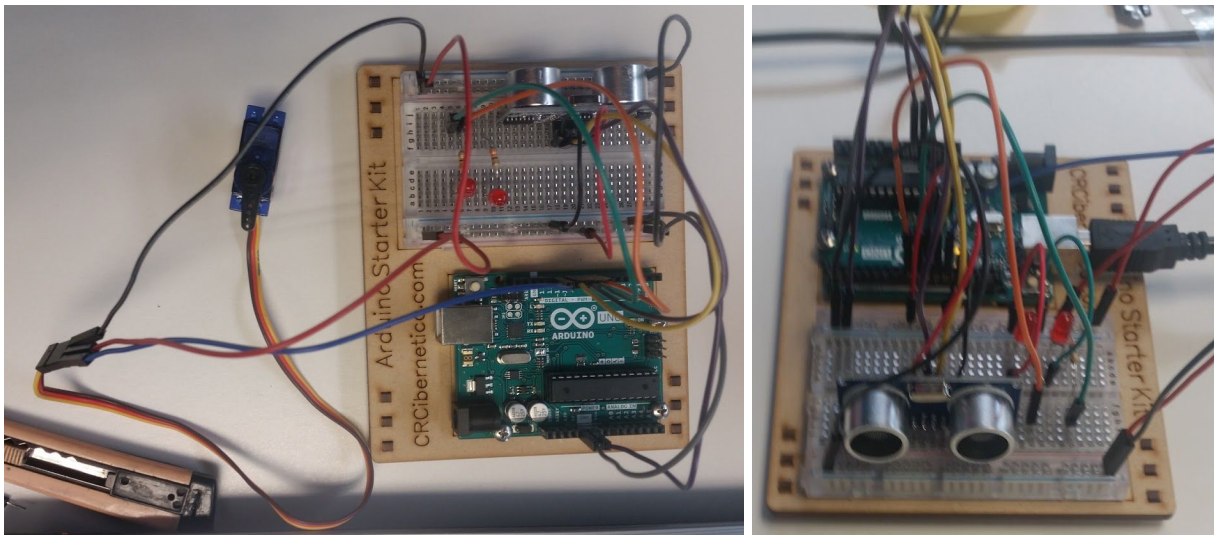

Figura 6. Implementación física en Arduino

En la figura anterior se muestran los LEDs apagados cuando el sensor no detecta un objeto a 20 centímetros de distancia. En la siguiente figura se muestra el correcto funcionamiento del sensor ultrasónico cuando se coloca un objeto a 20 centímetros de distancia. El resultado de detectar el objeto enciende y apaga los LEDs periódicamente en intervalos regulares.

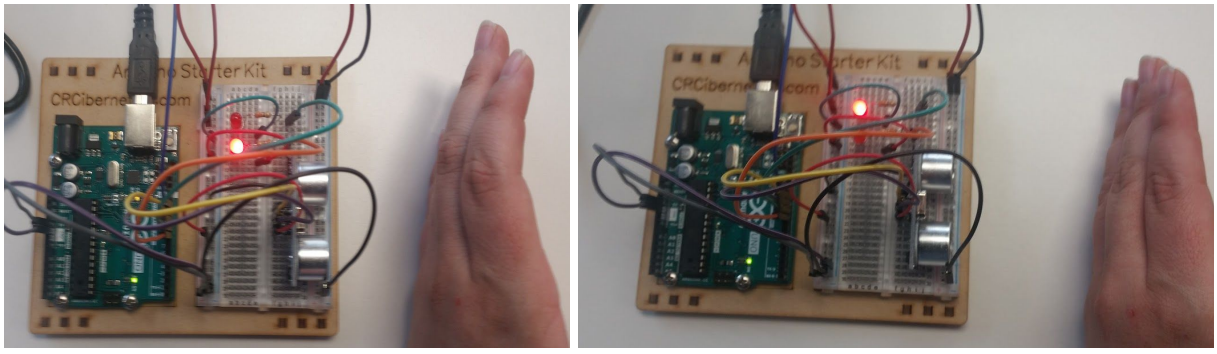

Figura 7. Demostración del funcionamiento del sensor ultrasónico

Conclusiones

- El uso de Tinkercad facilita el diseño del sistema en Arduino y logra simular el funcionamiento de un Arduino real.
- Se pudo implementar el sistema y comprobar su correcto funcionamiento rápidamente por hacerlo usando el lenguaje de programación en Arduino.
- El sensor ultrasónico funciona correctamente y detecta un objeto a 20 centímetros de distancia.
- El sensor ultrasónico puede detectar un objeto de 2 a 400 cm de distancia y no se ve afectado por la luz solar o el material negro, como el sensor infrarrojo, aunque los materiales acústicamente suaves como la tela pueden ser difíciles de detectar.
- El sistema actúa inmediatamente después de que detecta un objeto a 20 centímetros de distancia, pero tarda un par de segundos al pasar del Estado 1 al Estado 0 luego de que no detecta el objeto.
- Los resistores de 330 ohmios funcionaron como sustitutos de los de 220 ohmios planeados inicialmente en el diseño modular.
- Los LEDs rojos son más baratos y se queman más fácilmente que los azules. Por esto, son los ideales para hacer pruebas.
- El servo SG92R funcionó en el sistema propuesto porque puede girar 90 grados, lo necesario para simular el funcionamiento de una aguja de tránsito.

Recomendaciones

- Realizar primero el proyecto en alto lenguaje para verificar que el diseño funciona correctamente.
- Usar Tinkercad, página en la cual se puede diseñar y realizar simulaciones con Arduino, permite descargar el diagrama y el código realizado en la simulación para utilizarlo con el Arduino.

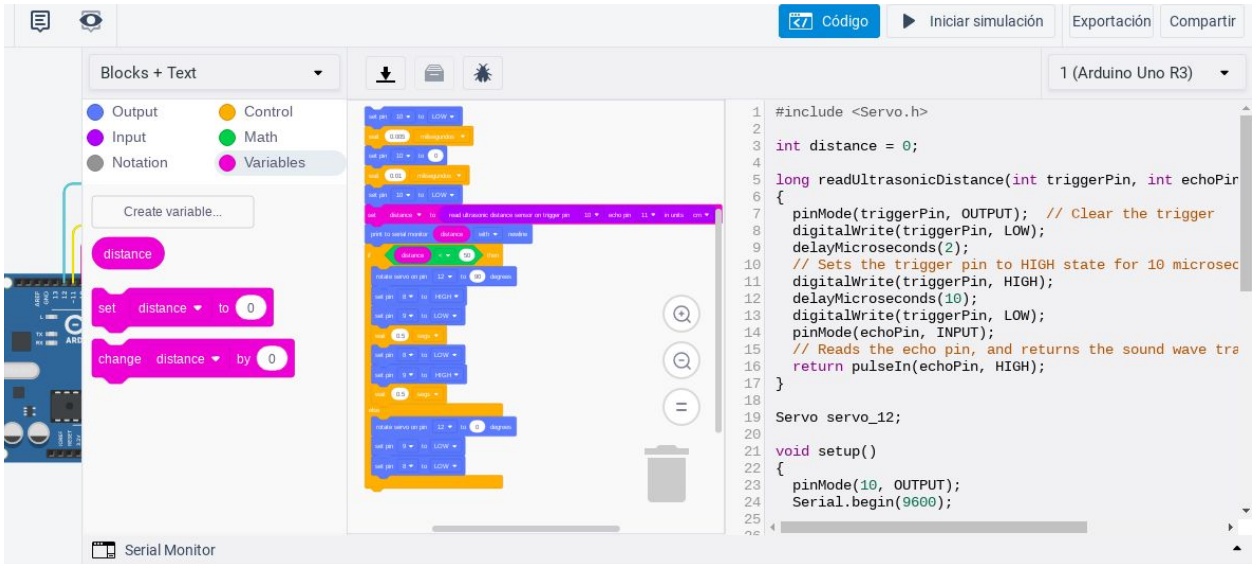

Figura 8. Código de la simulación en Tinkercad

- Utilizar un sensor ultrasónico para medir la distancia, ya que es bastante preciso en diferentes tipos de ambiente mientras que otros sensores como el infrarrojo pueden ser imprecisos al aire libre.
- Se recomienda convertir los datos adquiridos del sensor ultrasónico a centímetros o pulgadas para que sean mas fáciles de manejar.

Literatura consultada

Arduino - Ultrasonic Sensor with LEDs and buzzer. (n.d.). Retrieved November 9, 2018, from https://randomnerdtutorials.com/arduino-ultrasonic-sensor-with-leds-and-buzzer/

Barrett, S. F., & Pack, D. J. (2008). *Atmel AVR Microcontroller Primer: Programming and Interfacing*. Morgan & Claypool. doi:10.2200/S00100ED1V01Y200712DCS015

Complete Guide for Ultrasonic Sensor HC - SR04. (n.d.). Retrieved November 9, 2018, from https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

Tinkercad. (2018). Recuperado de https://www.tinkercad.com/

Apéndice: Diseño Modular

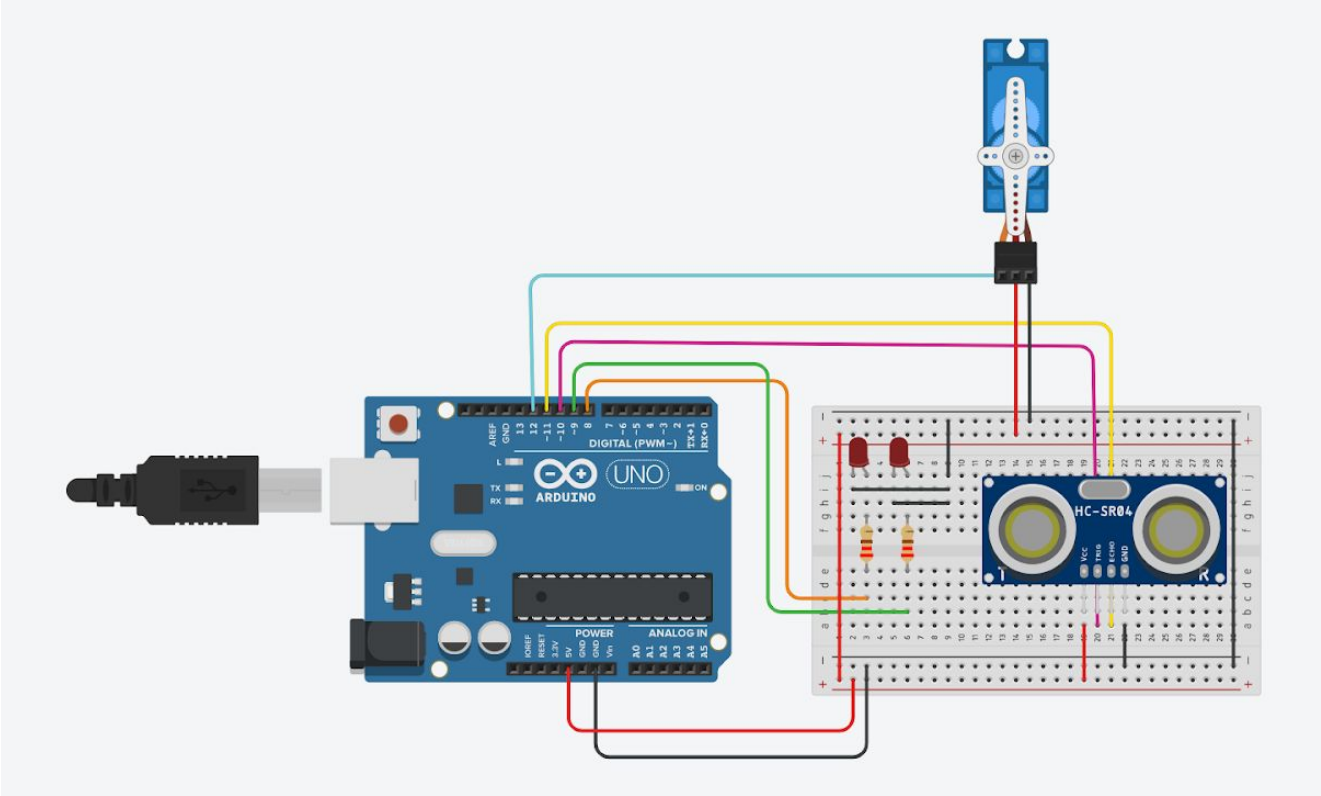

Figura 1. Diagrama realizado en Tinkercad del circuito que se implementó

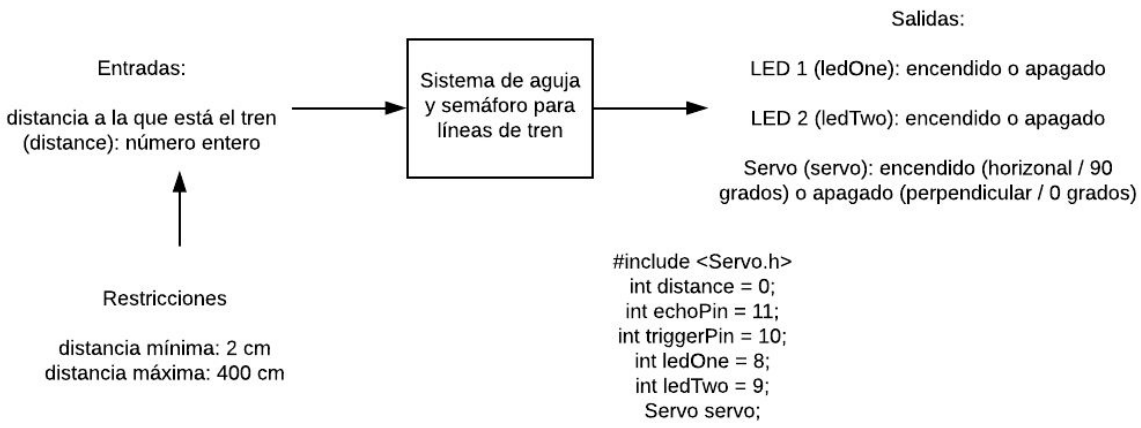

Figura 2. Diagrama de caja negra con entradas, salidas y restricciones

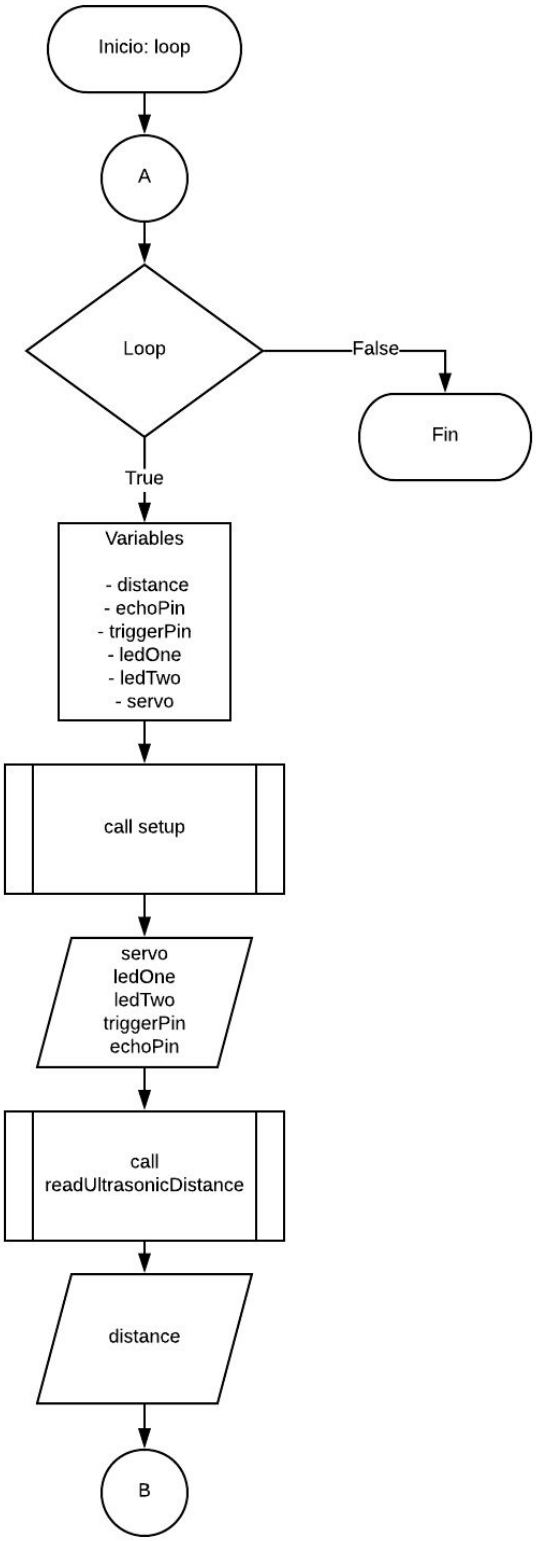

Figura 3. Primera parte del diagrama de flujo de la función *loop*

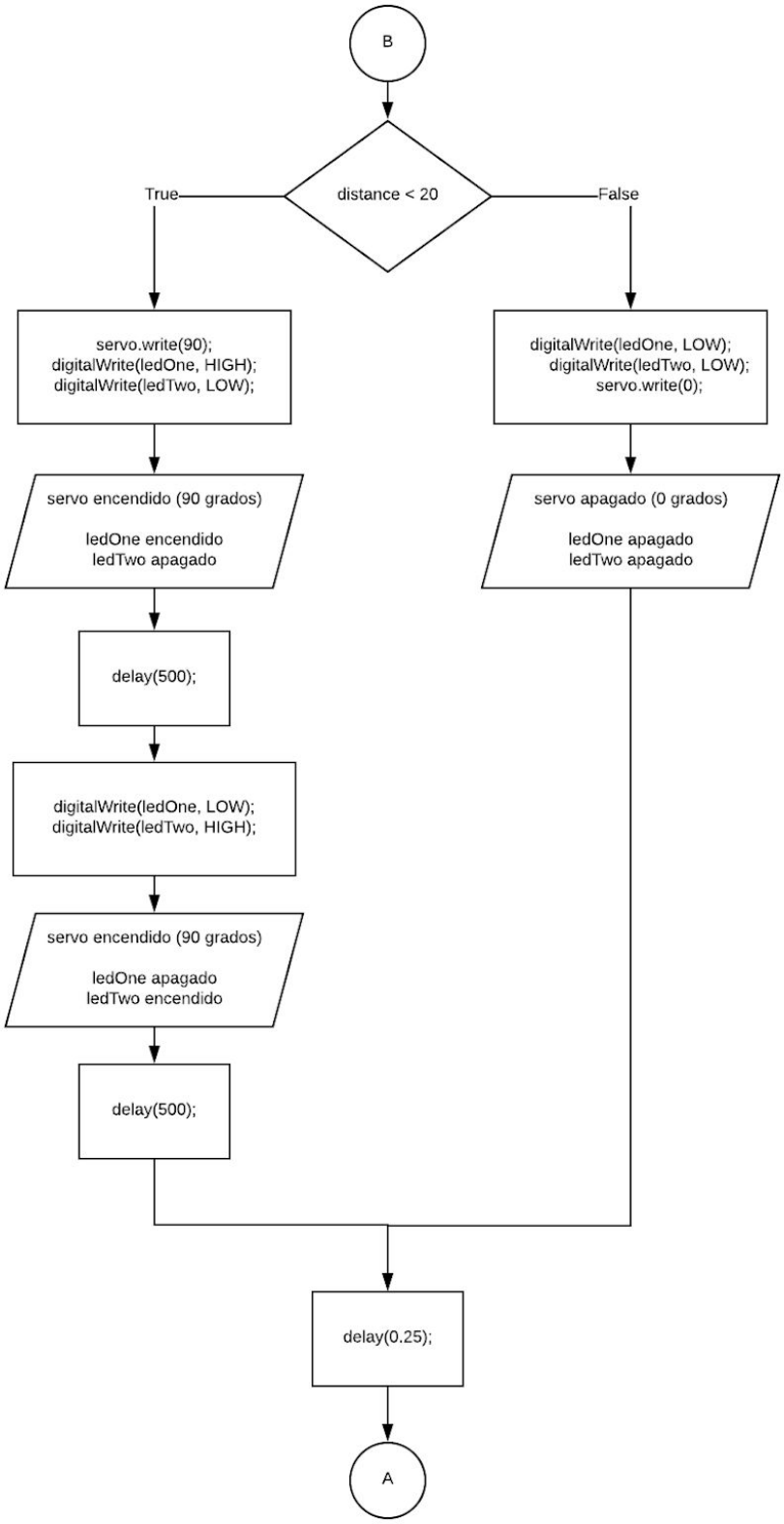

Figura 4. Segunda parte del diagrama de flujo de la función *loop*

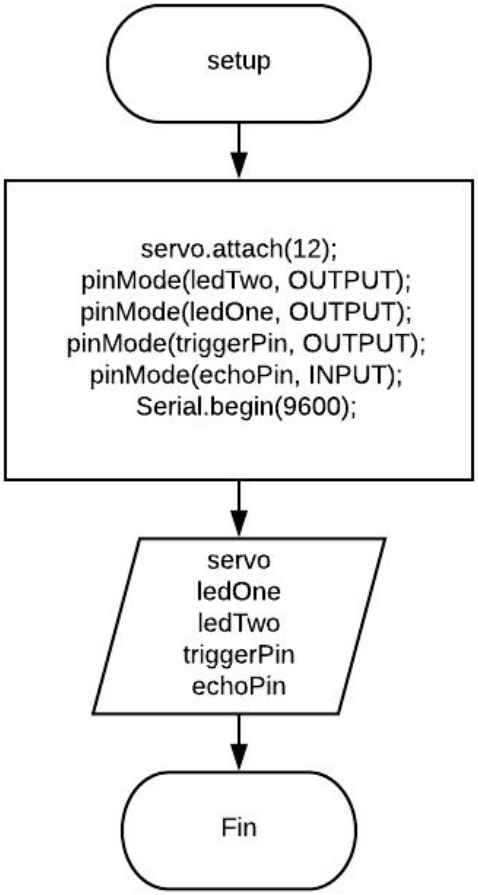

Figura 4. Diagrama de flujo de la función *setup*

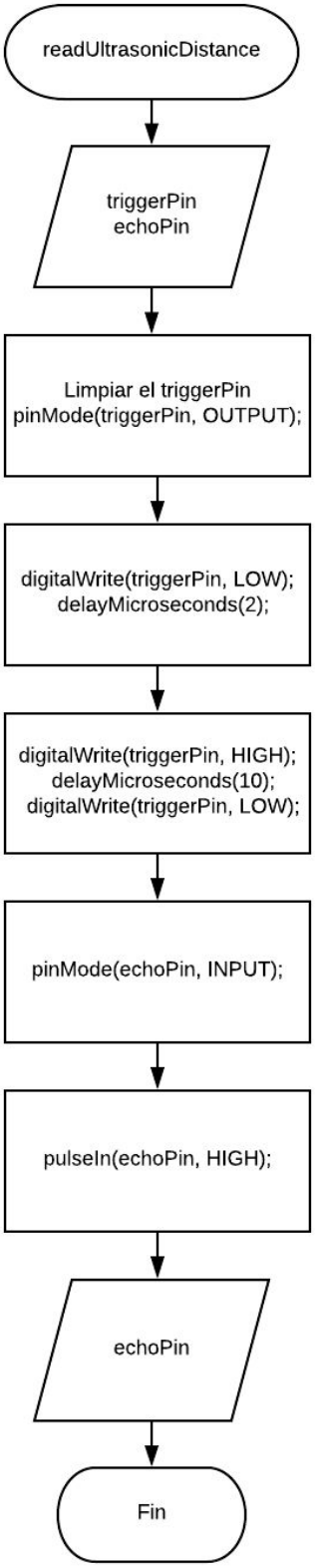

Figura 5. Diagrama de flujo de la función *readUltrasonicDistance*