

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: Segunda Etapa.

Integrantes:

Sara Castro Sáenz 2014085332

Ericka Céspedes Moya 2017239557

José David Martínez Garay 2018160104

Fecha de Entrega: 26 de noviembre

II Semestre, 2018

Introducción.

El proyecto realizado se centró en el desarrollo de la propuesta hecha por cada grupo, que constó de diseñar un sistema aplicado a un ambiente real basado en una arquitectura de computador utilizando el lenguaje ensamblador. La idea implementada tiene como propósito 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 funciona 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 gire, cuando el sensor ultrasónico ya no detecte al tren cerca y se apagará el semáforo.

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.

La documentación de esta consta de cuatro partes referentes al segundo 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 y todas las especificaciones técnicas del de la implementación en lenguaje ensamblador, a su vez, se obtuvieron conclusiones y recomendaciones de acuerdo a estos. En el apéndice se adjunta el diseño final del proyecto (maqueta y circuito funcional).

De acuerdo con De Coi et al. (2012), los LEDS, en inglés Light Emitting Diodes, son un tipo de diodo que se ilumina cuando la electricidad pasa a través de él. La electricidad solo fluye en una dirección a través de diodos. El ánodo (+), que normalmente se conecta al poder,

suele ser la pierna más larga, y el cátodo (-) es la pierna más corta. Según Gerber (2016), Los LED están polarizados, tienen una ventaja positiva y una ventaja negativa. La corriente solo puede fluir a través de una dirección, y cuando se conectan de la manera correcta, emiten luz. Una resistencia limitadora de corriente debe incluirse en el circuito para evitar dañar el LED. Según De Coi et al. (2012), los resistores resisten el flujo de energía eléctrica en un circuito, cambiando el voltaje y la corriente como resultado. Los valores del resistor se miden en ohmios y las rayas de colores en el resistor indican su valor.

De acuerdo con Santos (2013), el sensor ultrasónico HC-SR04 usa *sonar*, que es un método de localización acústica, para determinar la distancia a un objeto. Este sensor detecta de 2 a 400 cm o de 1 a 13 pies y su funcionamiento no se ve afectado por la luz solar, la oscuridad, o el material negro. Sin embargo, los materiales acústicamente suaves como la tela pueden ser difíciles de detectar. También, este aparato tiene un transmisor ultrasónico y módulo receptor. El sensor cuenta con dos pines: el *trigger* y el *echo*. El trigger, el transmisor, envía una señal que es un sonido de alta frecuencia. Cuando esta señal encuentra un objeto, esta es reflejada y el *echo* la recibe. El tiempo entre la transmisión y la recepción de la señal nos permite conocer la distancia a un objeto. Esto es posible porque sabemos la velocidad del sonido en el aire.

Según De Coi et al. (2012), un servomotor es un tipo de motor reductor que puede girar 180 grados. Este es controlado por las señales eléctricas en formato de pulsos que son enviadas desde la tarjeta Arduino; estos pulsos le dicen al motor a qué posición se debe de mover. Los impulsos siempre tienen los mismos intervalos, pero el ancho de estos impulsos puede variar entre 1000 y 2000 microsegundos. Además, los servomotores son un tipo especial de motores que no giran alrededor de un círculo continuamente, sino se mueven a una posición específica y permanecen en ella hasta que se les diga que se muevan de nuevo.

Se realizó la investigación del funcionamiento y configuración del sensor ultrasónico y el Servo motor en Ensamblador AVR. Al realizarse la implementación con un lenguaje de bajo nivel como lo es ensamblador, se consultó el Datasheet de cada dispositivo para saber el flujo de corriente que tenía que ser enviado mediante señales digitales por una cierta cantidad de tiempo, según el delay establecido por el fabricante.

En el caso de los servos, se necesitó averiguar la frecuencia del reloj del Arduino (20MHz), y de esta forma lograr realizar la rotación de 90° para ello fue necesario enviar corriente al PORTB (pin 9) durante 1.5ms, ya que el servo realiza rotaciones mediante pulsos eléctricos durante un determinado tiempo.

La configuración del sensor ultrasónico se realiza mediante la constante interacción del *ECHO* y el *TRIGGER* de este. Mediante delays y el input del Trigger se puede saber si un objeto está en el rango de distancia que detecta el sensor ultrasónico, es importante recalcar que este sensor trabaja con 15° de visión, por ende, objetos muy grandes estarán fuera de su rango en algunos casos. Con respecto a los leds, se puede utilizar el puerto 13, 1 o 0 para realizar conexiones inmediatas ya que estos poseen corriente directa.

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.

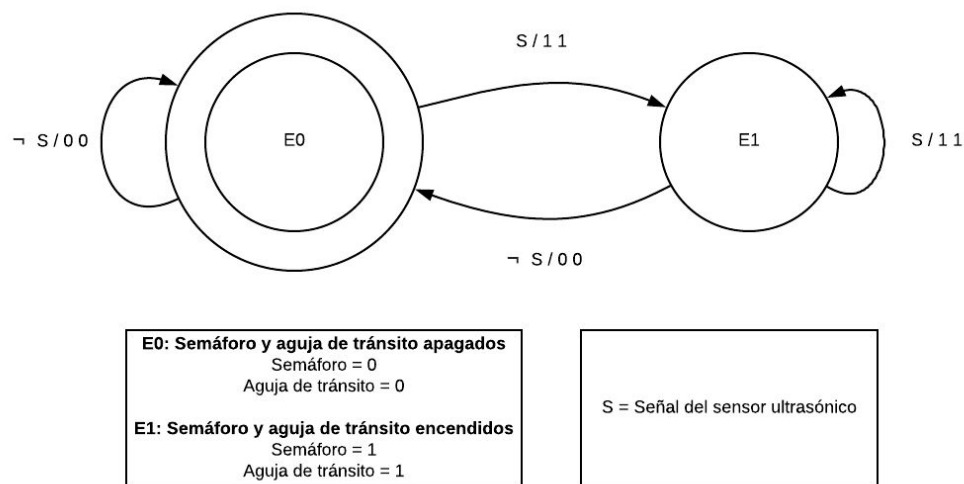


Figura 1. Máquina de estados del proyecto elegido.

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 encendido 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		Siguiete 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 está segunda parte del proyecto se logró implementar el proyecto propuesto utilizando ensamblador AVR. El lenguaje de programación AVR posee diversas prestaciones para modificar directamente el comportamiento del hardware, aspecto que se dificulta en lenguajes de alto nivel, por ejemplo, la cantidad de corriente eléctrica enviada, delays del dispositivo, cambiar las frecuencias del reloj del Arduino (F_{CPU}), etc.

Se investigó el funcionamiento del microcontrolador ATMEGA328p, cada salida y bloques de puertos, estos deberán ser inicializados y configurado su comportamiento en el código según la configuración de cada sensor/dispositivo externo dependiente al Arduino.

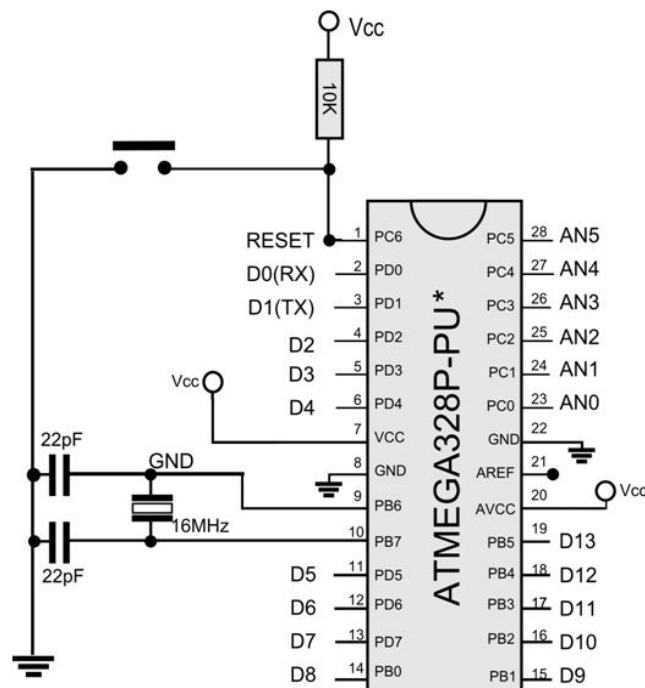
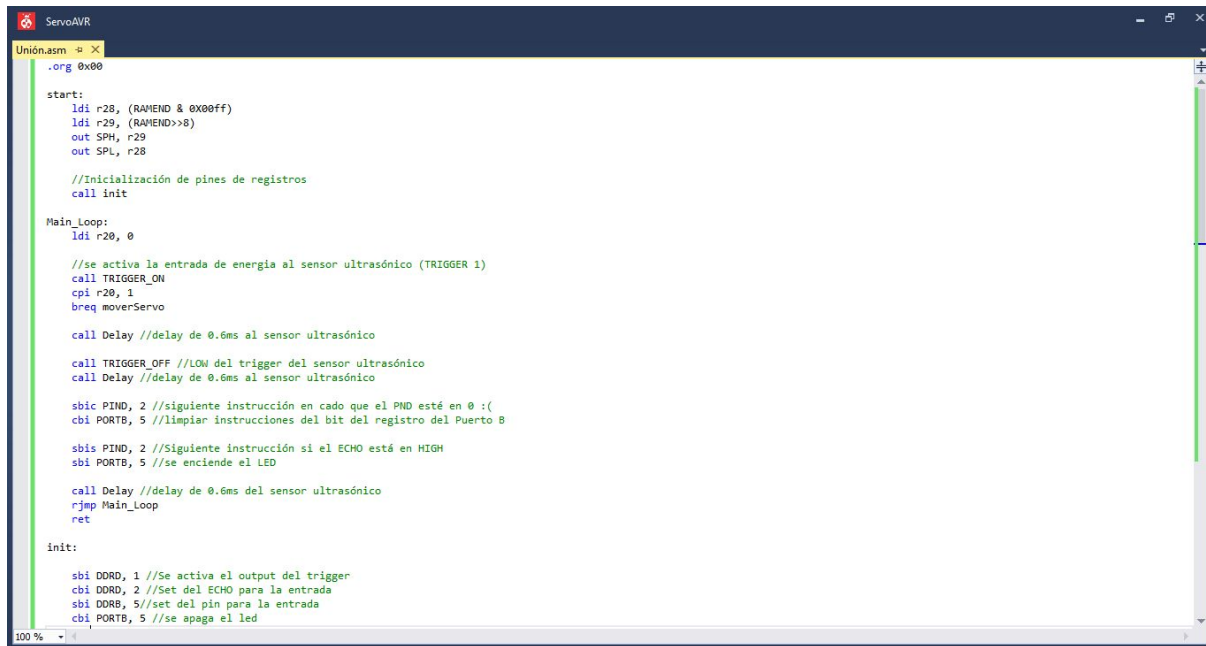


Figura 3. Arquitectura del Atmega328p.

El primer dispositivo auxiliar conectado al Arduino en inicializarse es el sensor ultrasónico. Este va a estar en un loop continuo detectando el movimiento de objetos a una distancia aproximada de 4-12cm. Con respecto a la información que envíe el sensor acerca del comportamiento de un objeto cercano, se desplegarán las demás operaciones del circuito (movimiento del servo o interacción de los leds).

The image shows a screenshot of the ServoAVR IDE. The main window displays an assembly file named 'Unión.asm'. The code is written in assembly language and includes comments in Spanish. It starts with a memory origin of 0x00. The 'start' section initializes the stack pointer (SPH) and the status register (SPL). It then calls an 'init' function. The 'Main_Loop' section initializes register r20 to 0 and enters a loop. Inside the loop, it calls 'TRIGGER_ON', delays for 0.6ms, calls 'TRIGGER_OFF', delays for 0.6ms, checks if pin 2 is low (PIND bit 2), and if so, clears bit 5 of PORTB. It then checks if pin 2 is high (PIND bit 2), and if so, sets bit 5 of PORTB. It delays for 0.6ms and jumps back to the start of the loop. The 'init' function sets DDRD bit 1, sets DDRD bit 2, sets DDRB bit 5, and clears PORTB bit 5. The status bar at the bottom shows '100 %' zoom.

```
.org 0x00

start:
    ldi r28, (RAMEND & 0X00FF)
    ldi r29, (RAMEND>>8)
    out SPH, r29
    out SPL, r28

    //Inicialización de pines de registros
    call init

Main_Loop:
    ldi r20, 0

    //se activa la entrada de energia al sensor ultrasónico (TRIGGER 1)
    call TRIGGER_ON
    cpi r20, 1
    brq moverServo

    call Delay //delay de 0.6ms al sensor ultrasónico

    call TRIGGER_OFF //LOW del trigger del sensor ultrasónico
    call Delay //delay de 0.6ms al sensor ultrasónico

    sbic PIND, 2 //siguiente instrucción en caso que el PIND esté en 0 :(
    cbi PORTB, 5 //limpiar instrucciones del bit del registro del Puerto B

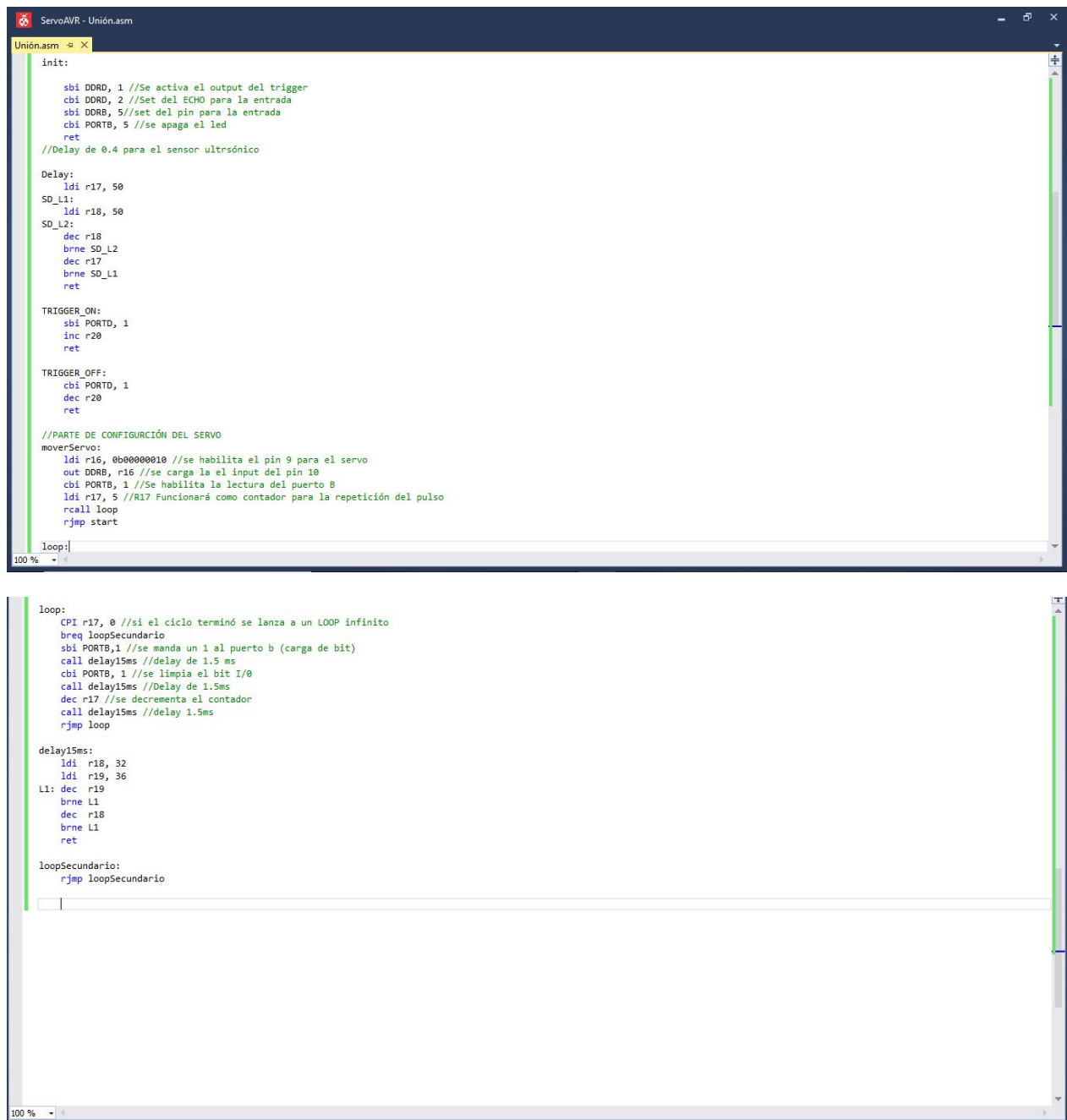
    sbis PIND, 2 //Siguiete instrucción si el ECHO está en HIGH
    sbi PORTB, 5 //se enciende el LED

    call Delay //delay de 0.6ms del sensor ultrasónico
    rjmp Main_Loop
    ret

init:
    sbi DDRD, 1 //Se activa el output del trigger
    cbi DDRD, 2 //Set del ECHO para la entrada
    sbi DDRB, 5 //set del pin para la entrada
    cbi PORTB, 5 //se apaga el led
```

Figura 4. Segmento del código para la configuración del sensor ultrasónico.

Como se puede apreciar en la *Figura 3*, se coloca el stack en el tope de espacio de memoria ram disponible. Luego de esto se llama a la función *Init* que se encarga de inicializar todos los pins y registros a ocupar durante el funcionamiento del sensor ultrasónico. Luego se habilita el *Trigger* y según la entrada de este se añade 1 al registro *r20*, este va a funcionar como una especie de “bandera” para que el servo gire. Posteriormente se inhabilita el *Trigger* y se habilita el *ECHO*, se recibe la información de él y se enciende el LED del arduino (L).



The image consists of two screenshots of an AVR assembly editor window titled 'ServoAVR - Unión.asm'. The top screenshot shows the 'init' section of the code, which includes setting up the trigger output, configuring delays for the ultrasonic sensor, and initializing the servo motor. The bottom screenshot shows the 'loop' section, which contains the main logic for sending pulses to the servo and handling delays.

```
init:
    sbi DDRD, 1 //Se activa el output del trigger
    cbi DDRD, 2 //Set del ECHO para la entrada
    sbi DDRB, 5 //set del pin para la entrada
    cbi PORTB, 5 //se apaga el led
    ret
//Delay de 0.4 para el sensor ultrasónico

Delay:
    ldi r17, 50
SD_L1:
    ldi r18, 50
SD_L2:
    dec r18
    brne SD_L2
    dec r17
    brne SD_L1
    ret

TRIGGER_ON:
    sbi PORTD, 1
    inc r20
    ret

TRIGGER_OFF:
    cbi PORTD, 1
    dec r20
    ret

//PARTE DE CONFIGURACIÓN DEL SERVO
moverServo:
    ldi r16, 0b00000010 //se habilita el pin 9 para el servo
    out DDRB, r16 //se carga la el input del pin 10
    cbi PORTB, 1 //Se habilita la lectura del puerto B
    ldi r17, 5 //R17 Funcionará como contador para la repetición del pulso
    rcall loop
    rjmp start

loop:
    CPI r17, 0 //si el ciclo terminó se lanza a un LOOP infinito
    breq loopSecundario
    sbi PORTB, 1 //se manda un 1 al puerto b (carga de bit)
    call delay15ms //delay de 1.5 ms
    cbi PORTB, 1 //se limpia el bit I/O
    call delay15ms //Delay de 1.5ms
    dec r17 //se decreuenta el contador
    call delay15ms //delay 1.5ms
    rjmp loop

delay15ms:
    ldi r18, 32
    ldi r19, 36
L1: dec r19
    brne L1
    dec r18
    brne L1
    ret

loopSecundario:
    rjmp loopSecundario
```

Figura 5. Configuración de delays del sensor ultrasónico y la inicialización del servo.

Como se puede observar en la figura anterior, se da la implementación del *delay* de 0.4ms del sensor ultrasónico para detectar una menor distancia en comparación a la que trae por default (20cm). A su vez, se inicializa el puerto 9 del Arduino para establecer la conexión con el Servo Motor. Una vez realizada la inicialización del servo, se manda corriente al servo mediante el pin 9 durante 1.5ms para realizar un giro de 90°, esta rotación se realizará cuando el registro r20 tenga un valor de 1, según la cercanía del objeto al sensor ultrasónico.

Registros y puertos utilizados.	
Registro.	Propósito.
R17	Contador de ciclos que debe hacer el servo para rotar 90°.
R20	Funciona como bandera para indicar al servo si debe girar o no (1=girar, 0=estático).
R28, R29	Inicialización del puntero del stack y se reserva una palabra de 1 byte (RAMEND).
R17, R18	Registros temporales para los diversos delays.
PORTB	Se habilita el bloque B de puertos para la lectura del PIN9 (servo).
PIND	Se habilita la lectura de los pines 1 y 2 (RX, TX) para recibir información del sensor ultrasónico.

Cuadro 1. Registros y puertos utilizados durante la implementación.

Diseño del circuito.

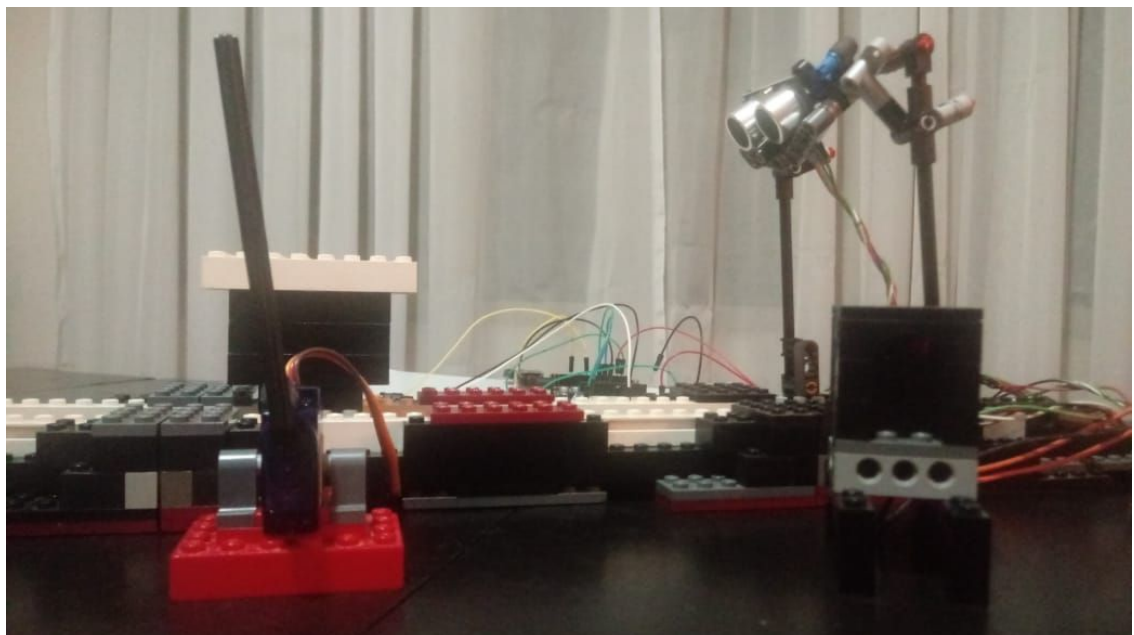


Figura 6. Maqueta realizada con base en el proyecto propuesto.

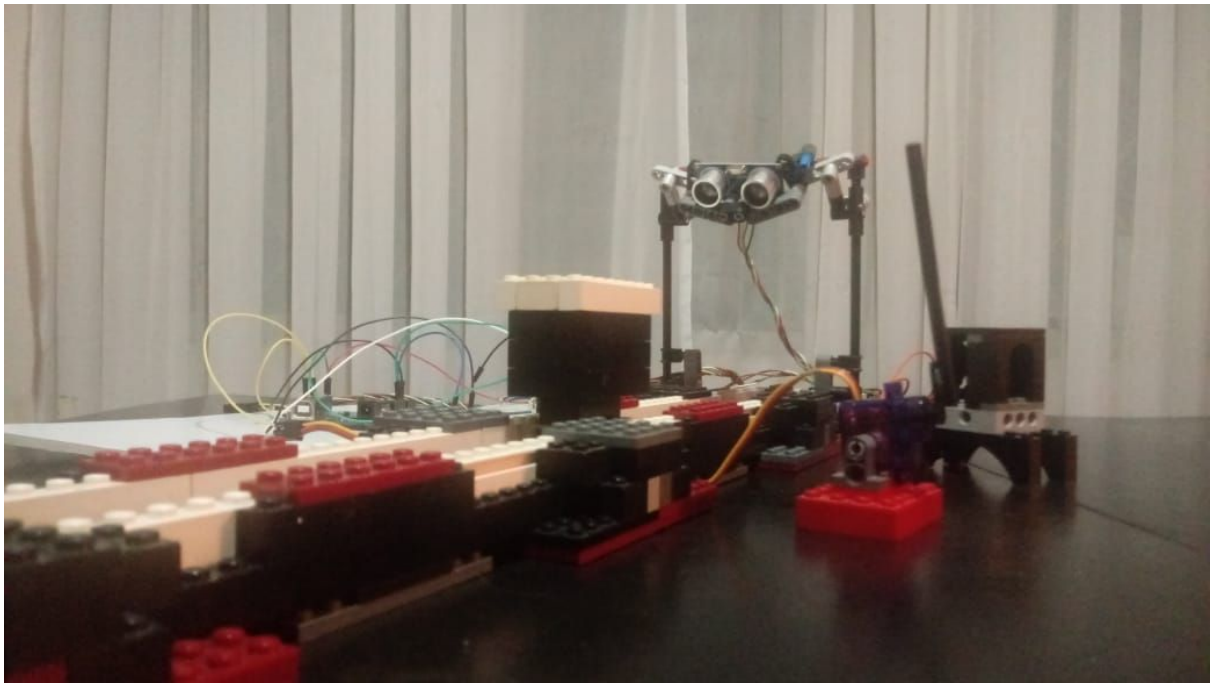


Figura 7. Maqueta realizada con base en el proyecto propuesto.

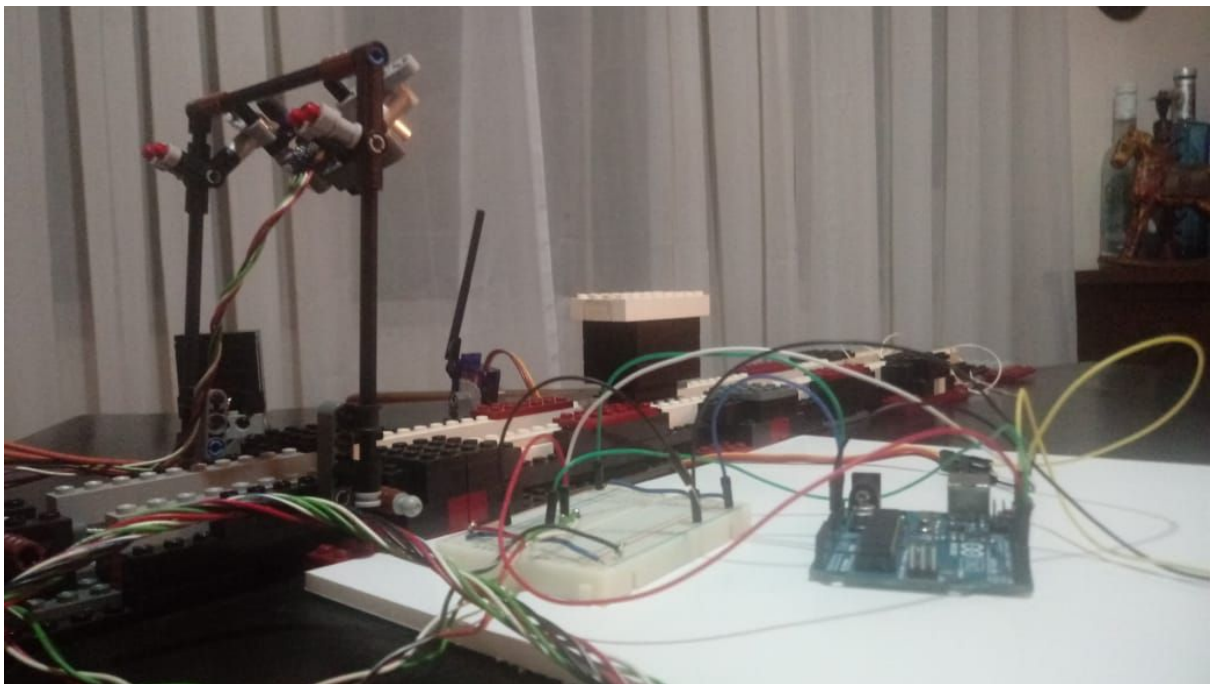


Figura 8. Maqueta realizada con base en el proyecto propuesto.

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 220 ohmios funcionaron como sustitutos de los de 330 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.
- La búsqueda de información acerca de ensamblador AVR resulta exhaustiva debido a que gran parte decide utilizar el lenguaje C para la programación a bajo nivel que interactuar directamente con los registros y el microcontrolador.
- Resulta complejo buscar el Datasheet de cada dispositivo (en este caso el servo) ya que solo aparecen las referencias de versiones anteriores.
- Se tuvieron que realizar cambios con respecto al plan piloto presentado en la primera fase del proyecto, aspectos como la implementación de los leds y la rotación a la posición 0 del servo sufrieron modificaciones en comparación con su desarrollo de alto nivel debido al diseño descendente que se fue realizando.
- Se dio una correcta configuración del sensor ultrasónico y la modificación de distancia que este puede rastrear.
- Se dio la utilización de corriente externa (CA) para la alimentación del Arduino y de esta forma lograr mantener activo todo el circuito.

Recomendaciones.

- Realizar primero el proyecto en lenguaje de alto nivel 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.

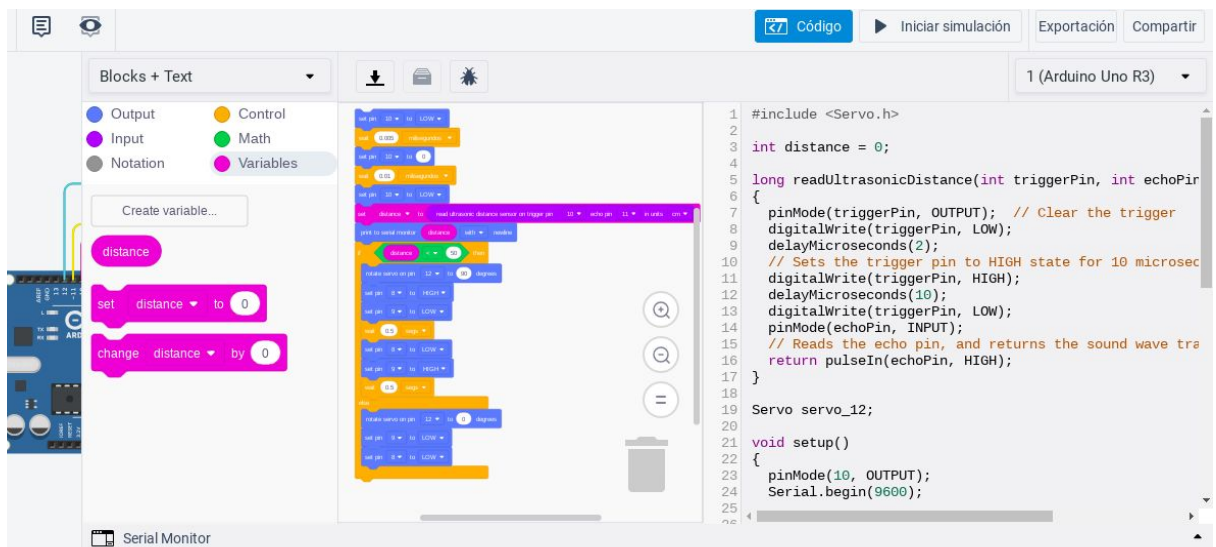


Figura 8. Código de la simulación en Tinkercad

- 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 más fáciles de manejar.
- Tenga cuidado con el manejo de los delays, una incorrecta implementación puede generar una sobrecarga en el circuito y provocar que algún otro dispositivo se queme.
- Procure realizar diagramas de flujo, diagrama de estados, modelado UML, etc. Esto facilita la abstracción del código en ensamblador.
- Trate de investigar constantemente los mnemonics de AVR y su forma de ejecutarlos para evitar pérdida de tiempo encontrando errores minúsculos.

Literatura consultada

Arduino - Ultrasonic Sensor with LEDs and buzzer. (n.d.). Recuperado el 9 de noviembre del 2018 de <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

De Coi, K., Bassi, E., Banzi, M., Martino, G., Shiloh, M., Nebiolo, M., . . . Mellis, D. (2012). *Arduino Projects Book* (S. Fitzgerald, M. Shiloh, & T. Igoe, Eds.). Torino, Italy.

Recuperado el 9 de noviembre del 2018 de

<https://bastiaanvanhengel.files.wordpress.com/2016/06/arduino-projects-book.pdf>

Santos, R. (2013). Complete Guide for Ultrasonic Sensor HC - SR04. Recuperado el 9 de noviembre del 2018 de

<https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>

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

Microchip (2018). *AVR Assembler Instruction Mnemonics* [Sitio web]. Recuperado de

https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_instructions.Branch_Instructions.html

Arduino. (2018). *Port Registers* [Sitio web]. Recuperado de

<https://www.arduino.cc/en/Reference/PortManipulation>

TowerPro. (s.f). *SG90 ServoMotor Datasheet*. Recuperado de

http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

Christoph Redecker. (s.f). *Assembler Tutorial Jumps, Calls and the Stack*. Recuperado de

http://www.avrbeginners.net/new/wp-content/uploads/2011/08/avrbeginners_04_Jumps_Calls_and_the_Stack_1.0.1.pdf

(2008). *Generating PWM signals using Timers in the ATMega chip*. Recuperado de

<https://mil.ufl.edu/5666/handouts/ATMPWM.pdf>

Atmel Corporation. (2016). *Atmel AVR Instruction Set Manual*. San José, CA, EEUU:

Recuperado de

<http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf>

Schuster, G. (2013). *AVR assembler* [diapositiva de PowerPoint]. Recuperado de

<http://www.uni-obuda.hu/users/schuster.gyorgy/avrasm.pdf>

Apéndice: Diseño Modular

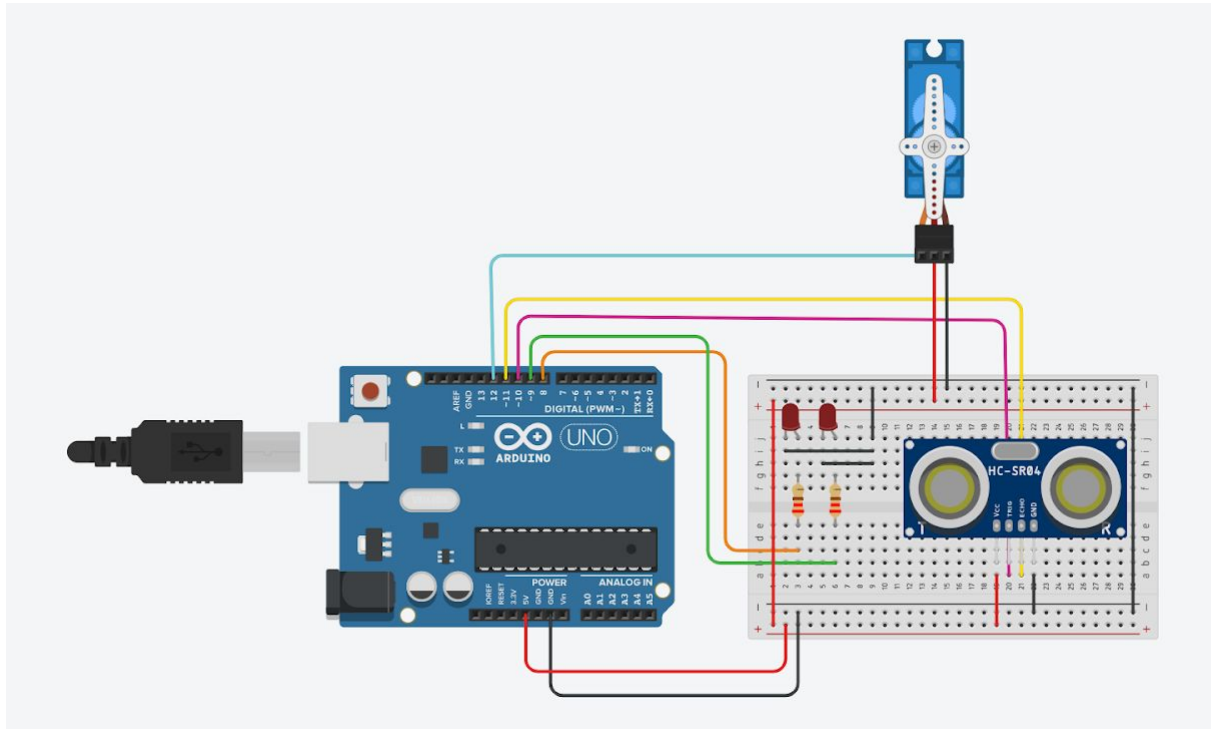


Figura 1. Diagrama realizado en Tinkercad del circuito que se implementó

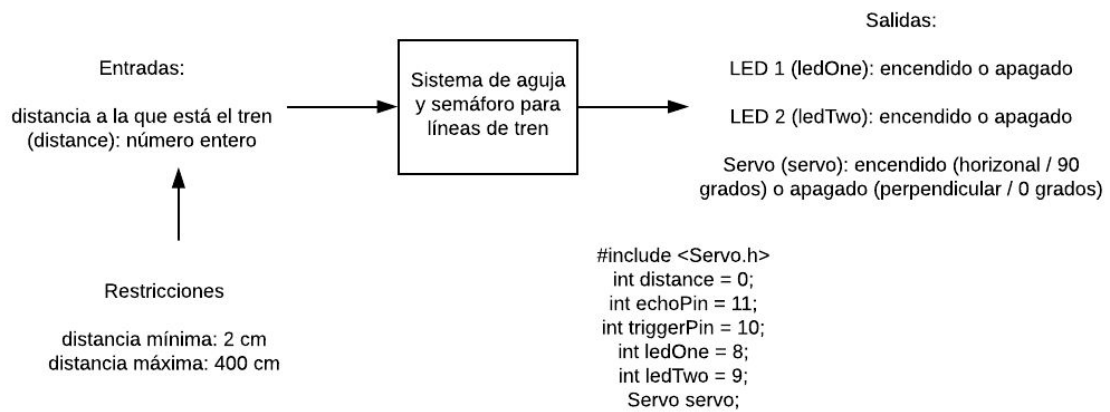


Figura 2. Diagrama de caja negra con entradas, salidas y restricciones.

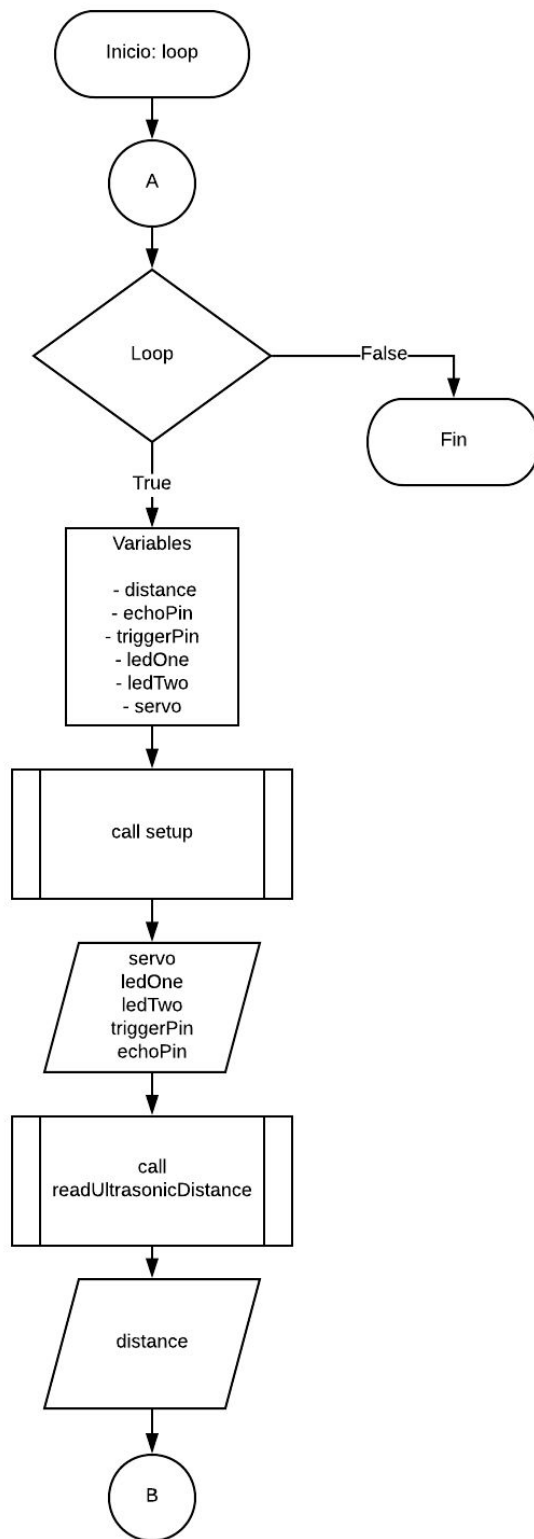


Figura 3. Primera parte del diagrama de flujo de la función *loop*.

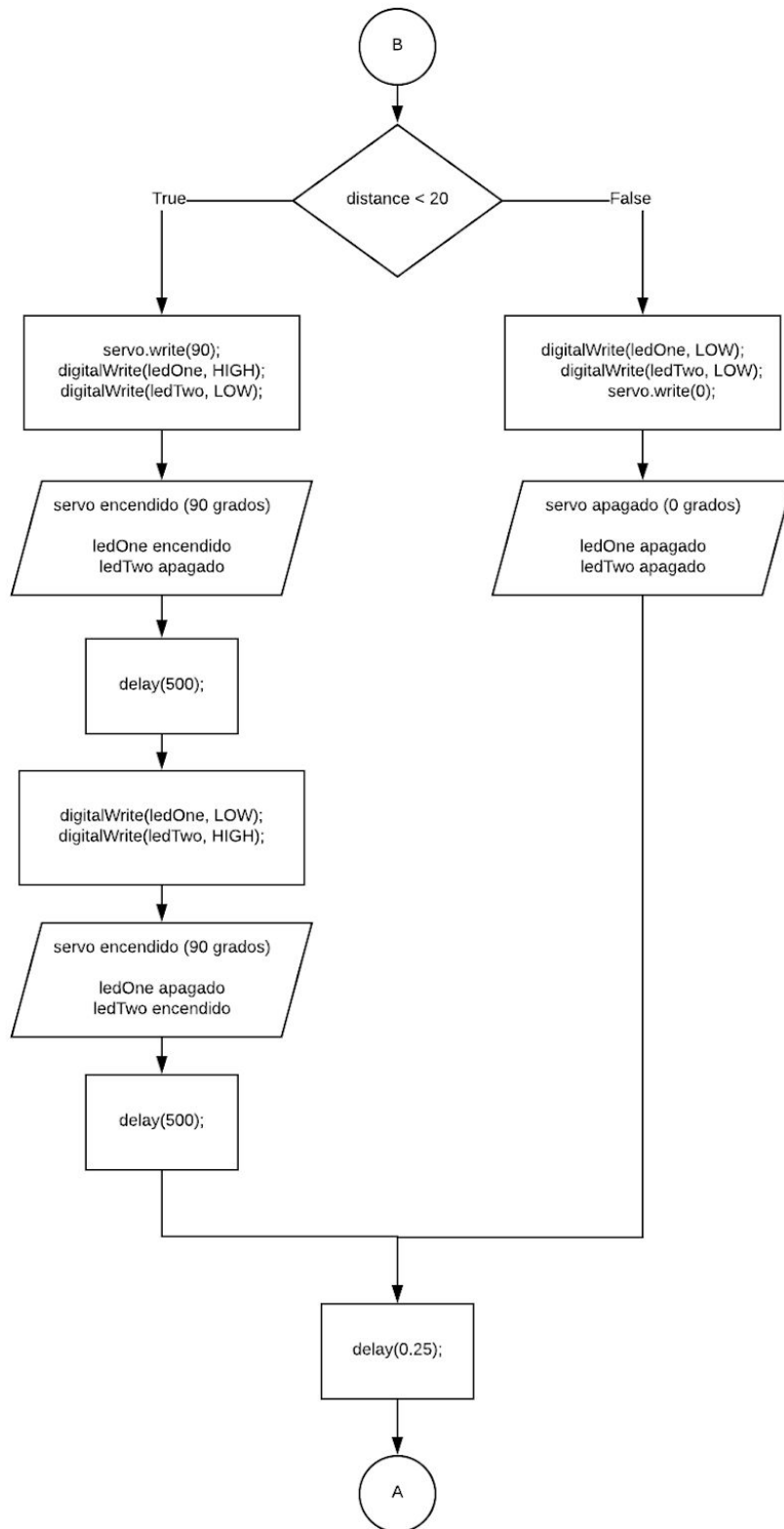


Figura 4. Segunda parte del diagrama de flujo de la función *loop*.

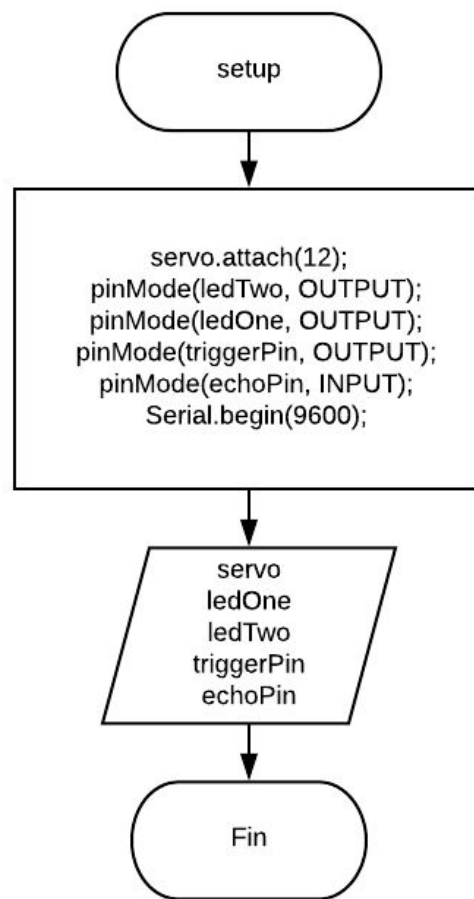


Figura 4. Diagrama de flujo de la función *setup*.

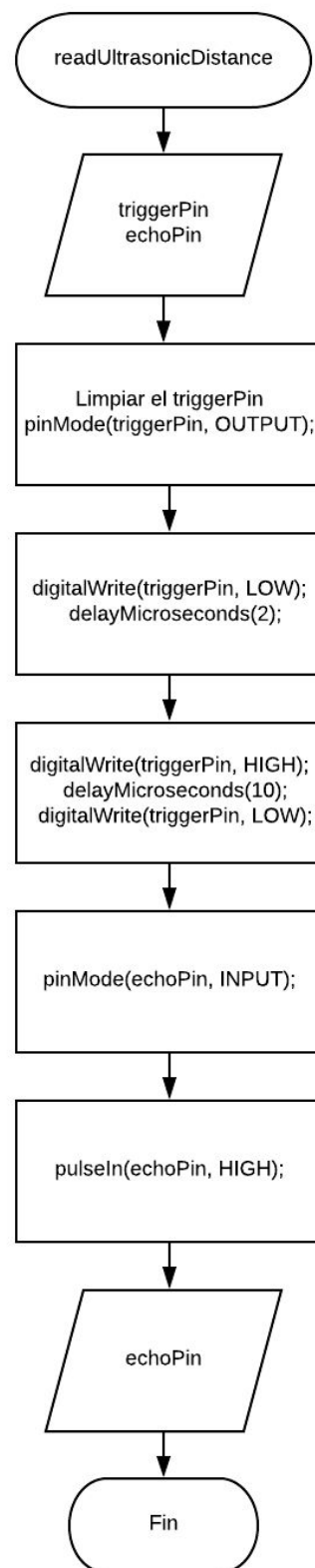


Figura 5. Diagrama de flujo de la función *readUltrasonicDistance*.

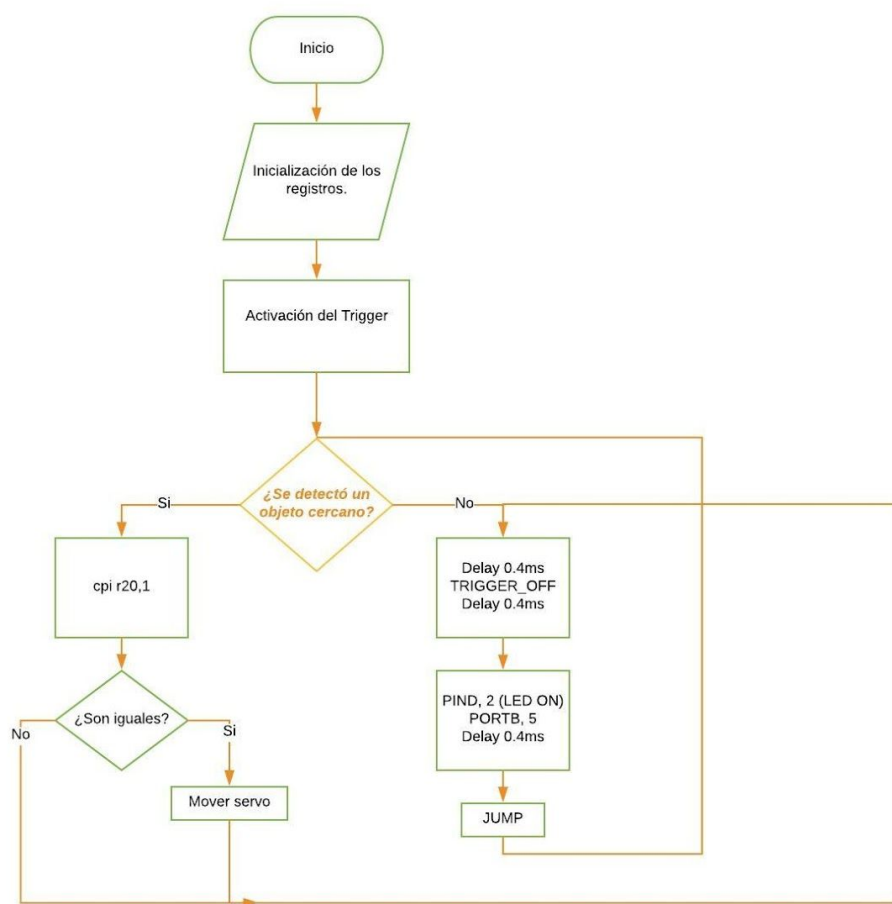


Figura 6. Diagrama de flujo del funcionamiento del circuito en bajo nivel.