Digital Systems and Microprocessors

Project 5: Ultrasonic Sensor

Enrique Almazán Sánchez
Víctor Miguel Álvarez Camarero

Table of Contents

Objective

The overall objective of this project is to demonstrate the use and skill using the Arduino System, and the handling of an ultrasonic sensor to measure distances.

Description

In this project, an RGB LED, a buzzer and the HC-SR04 ultrasonic sensor will be used as elements for the assembly of an electronic distance measurement system. Connect the components to the following pins on the Arduino development board as follows:

- **RGB LEDs with respective resistors to 22, 23 and 24.**
- **Buzzer, connected to pins 26 and GND.**
- **HC-SR04, connected to pins 36 (Echo pin) and 37 (Trigger pin).**

Requirements

For this setup, we will require the following components:

- Arduino Mega 2560.
- Cables for connections.
- 3 resistors with a resistance of 220 Ohms each.
- Protoboard for circuit assembly.
- RGB Led.
- PWM Piezoelectric.
- **HC-SR04 ultrasonic sensor**, which measures the distance by sending a sound wave at a specific frequency determined by the characteristics of the transducers used. The wave travels through the medium and, upon encountering the obstacle or object, part of the signal sent is reflected and transmitted back to the ultrasonic sensor.
  The HC-SR04 is not going to measure distances lower than 15mm because if not we would have an object very close to the transmitter and it would not reach the receiver.

Exercise 1

Write a code in which the distance of an obstacle in front of the ultrasonic sensor is measured. The distance ranges to be measured and the actions to be taken are as follows:

- **Distance between 20 and 100 mm, range 0, red LED on, other LEDs off, buzzer off.**
- **Distance greater than 100 mm and less than or equal to 200 mm, range 1, blue led on, other leds off, buzzer of.**
- **Distance greater than 200 mm and less than or equal to 300 mm, range 2, green led on, other leds off, buzzer off**
- **Distance over 300 mm, range 3, LEDs off, buzzer on and frequency to generate 440 Hz.**

Both the measured distance and the range should also be displayed on the computer.

The code presented for this exercise is designed to measure the distance of an obstacle using an ultrasonic sensor (HC-SR04) and adjust an RGB LED and a buzzer based on the measured distance.

First of all, some variables are initialized, as well as the baud rate, the speed at which data bits are going to be transmitted, at 9600.

- "lowestPin" (constant integer): representing the pin number to which the RED LED is connected, in this case 2.
- "mediumPin" (constant integer): representing the pin number to which the GREEN LED is connected, in this case 3.
- "highestPin" (constant integer): representing the pin number to which the BLUE LED is connected, in this case 4.
- "brightness" (integer): representing the brightness of the RGB LED, set to 255.
- "tempo" (integer): defines the speed of the musical notes being played by the buzzer. It is set to 200 milliseconds.
- "tones" (array): defines the frequencies (in Hz) corresponding to the buzzer, set at 440Hz.
- "duration" (integer): defines the duration of each musical note in seconds, set to 3 seconds.
- "buzzerPin": defines the pin number to which the buzzer is connected. In this case, it is set to pin 22.
- "range": stores the detected range based on the measured distance, set to 3.

```
const int baudrate = 9600; // Baudrate is the speed at which data bits
are going to be transmitted
const int lowestPin = 2;  // Red Pin
const int mediumPin = 3;  // Green Pin
const int highestPin = 4; // Blue Pin
int brightness = 255; // variable defined for brightness

// The following variables will be used for the musical notes of the
buzzer
int tempo = 200; // tempo used
// char notesName[] = {'b', 'c'};
int tones[]= {440}; // variable to define the tones
int duration = 3; // variable to define the duration
int buzzerPin = 22; // Pin at which the buzzer is connected
int range = 0; // variable to define the range
```

Regarding the ultrasonic sensor setup, the “EchoPin” and “TriggerPin” are defined at 36 and 37 ports respectively, using a define statement (useful for saving resources since the system memory is not been used for storing information). In addition, the “SoundSpeed” is also stated, computed with the formulas given in class (depends on the temperature having 25´3 ºC).

```
#define EchoPin 36
#define TriggerPin 37

const long SoundSpeed = 173; // variable to define de sound speed
long EchoTime, Distance;
```

Moreover, in the setup function, begins serial communication with the specified baud rate (“*Serial.begin*”). Also, all the pins are configured to output, except “EchoPin” which is an input.

```
// STEP 2: SETUP FUNCTION. This routine runs once when reset is pressed
void setup() {

  // initialize serial communication at 9600 bits per second:
  Serial.begin(baudrate);

  // All the pins used are set as outputs except the EchoPin
  pinMode(buzzerPin, OUTPUT);
  pinMode(lowestPin, OUTPUT);
  pinMode(mediumPin, OUTPUT);
  pinMode(highestPin, OUTPUT);
  pinMode(EchoPin, INPUT);
  pinMode(TriggerPin, OUTPUT);
  digitalWrite(TriggerPin, 0);

  delay(200); // delay of 200ms
}
```

Furthermore, in the loop function the following has been performed:

```
// STEP 3: LOOP FUNCTION
void loop() {
  resetled(); // Reset the LED, helping to tern them off in each
iteration

  // print out the value you read:
  digitalWrite(TriggerPin, 1);
  delayMicroseconds(10);
  digitalWrite(TriggerPin, 0);
  EchoTime = pulseIn (EchoPin, 1);
  Distance = SoundSpeed*EchoTime/1000;
  Serial.print("Echo Time(us) = ");
  Serial.print(EchoTime);
  Serial.print("  - Distance(mm) = ");
  Serial.println(Distance);
  delay(500);

  // Determine the range based on the sensor value (distance) in the
following if-else body

  {if-else body shown below}

  Serial.println("The range is: ");
  Serial.print(range);
  delay(1000);  // delay in between distance reads for stability
}
```

- The “*resetled()*” function is called to turn off all LEDs at the beginning of each iteration by setting their brightness to 0.

```
// STEP 4: reset LED color from measurement to measurement
void resetled(){
  int brightness = 0;
  analogWrite(highestPin, brightness);
  analogWrite(mediumPin, brightness);
  analogWrite(lowestPin, brightness);
}
```

- The value of the distance that is being measured by the US sensor is computed, stored in the variable “Distance”, through the following operation:

$$Distance\ =\ SoundSpeed * EchoTime/1000$$

- Moreover, this distance value will be printed in the serial monitor console in order to correctly watch the functioning of the circuit (“Serial.print” or “Serial.println”).
- Then a delay of 500ms is implemented before going with the conditional clauses, capturing a set of distances and thus, having different outcomes. These clauses can be seen in the code below:

```
if ((Distance>=20)&&(Distance<100)){
    range = 0;
    analogWrite(lowestPin, brightness);
    noTone(buzzerPin);
  }
  else if ((Distance>=100)&&(Distance<200)){
    range = 1;
    analogWrite(highestPin, brightness);
    noTone(buzzerPin);
  }
  else if ((Distance>=200)&&(Distance<=300)){
    range = 2;
    analogWrite(mediumPin, brightness);
    noTone(buzzerPin);// stop the sound
  }
  else{
    range = 3;
    tone(buzzerPin, tones[0], duration * tempo);
  }
```

The assembly circuit can be seen in the below figures for different distances and outcomes, while the entire implementation in the video 'Exercise_1'.

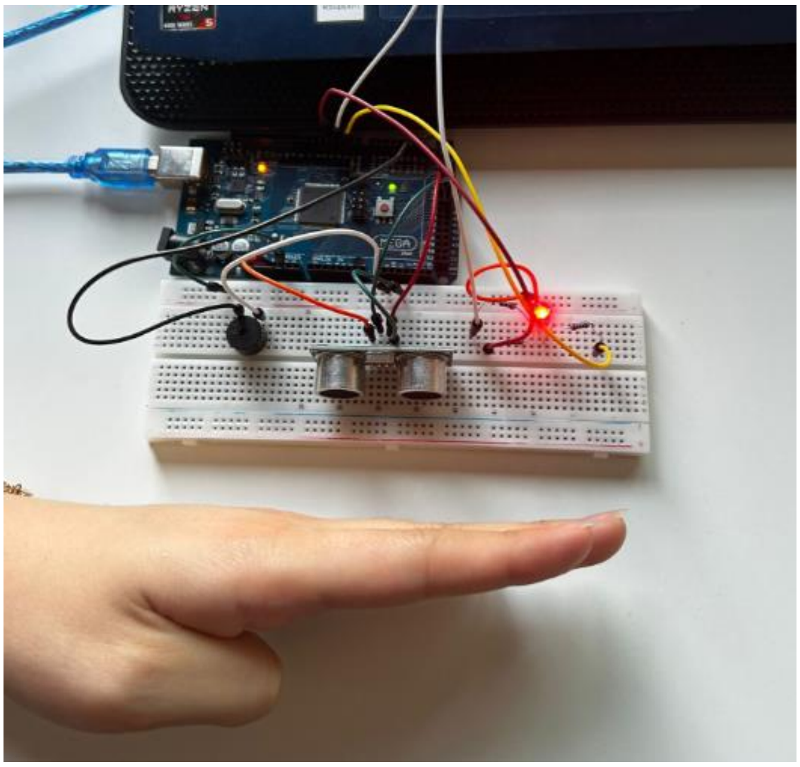


Figure 1: Assembly circuit for small distance.

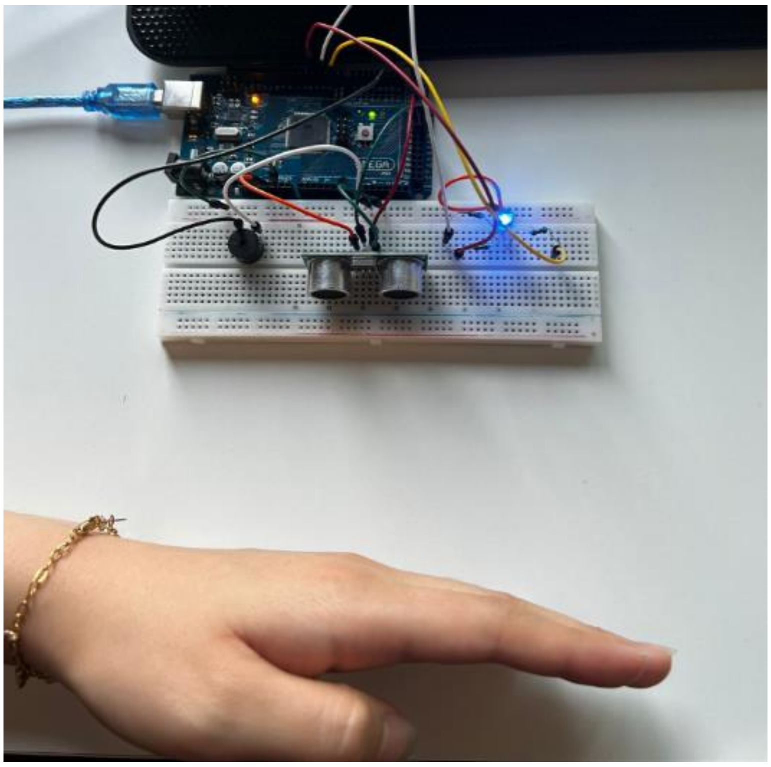


Figure 2: Assembly circuit for medium distance.

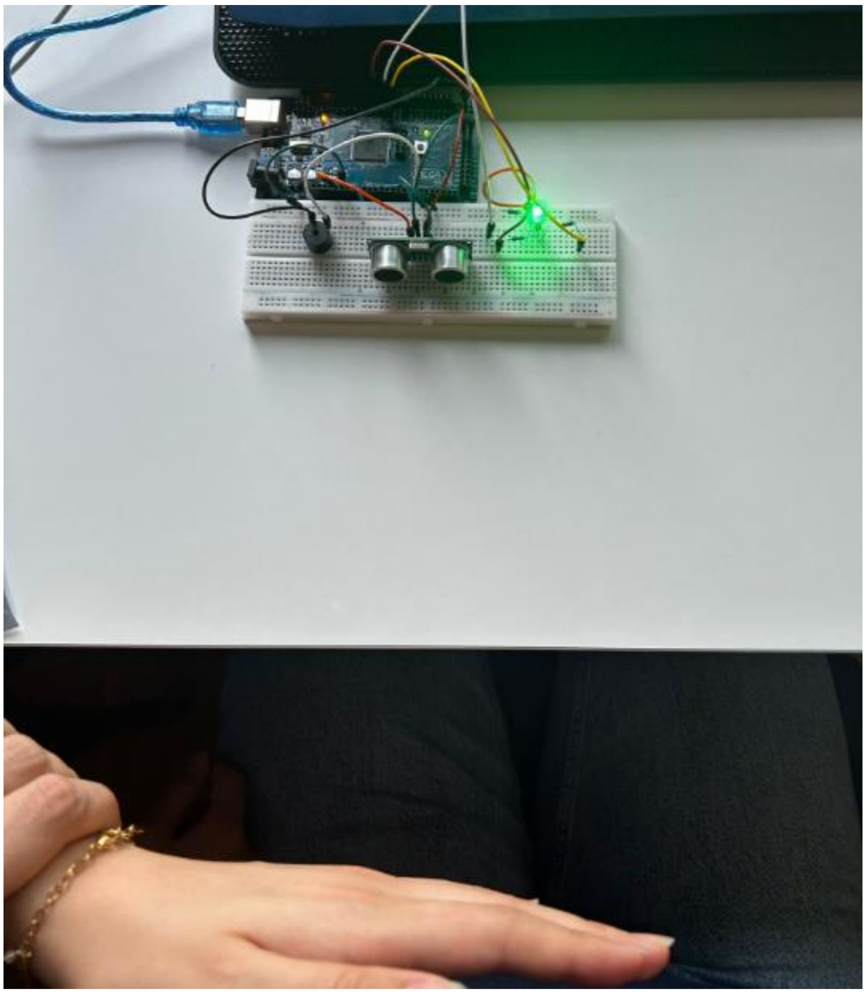
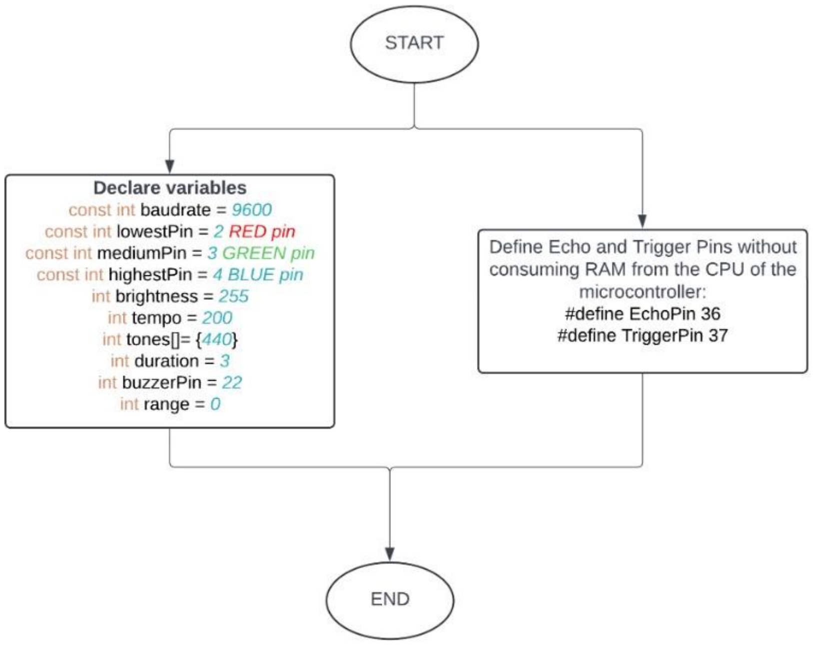


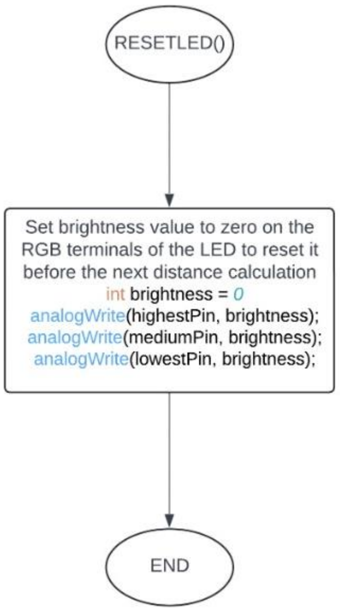
Figure 3: Assembly circuit for long distance.

Flowcharts

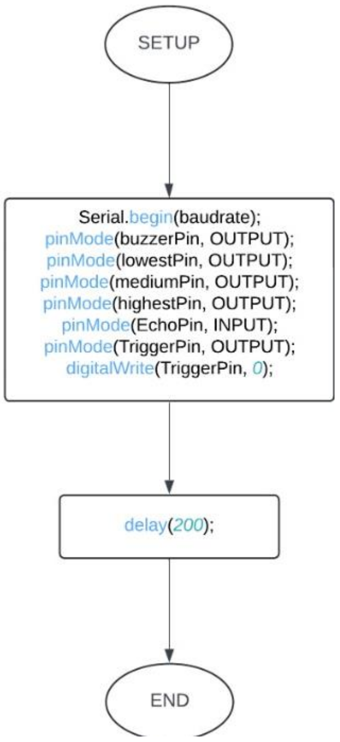

START
Declare variables
Define Echo and Trigger Pins without consuming RAM from the CPU of the microcontroller:
END

```
const int baudrate = 9600
const int lowestPin = 2 RED pin
const int mediumPin = 3 GREEN pin
const int highestPin = 4 BLUE pin
int brightness = 255
int tempo = 200
int tones[]= {440}
int duration = 3
int buzzerPin = 22
int range = 0
```

```
#define EchoPin 36
#define TriggerPin 37
```

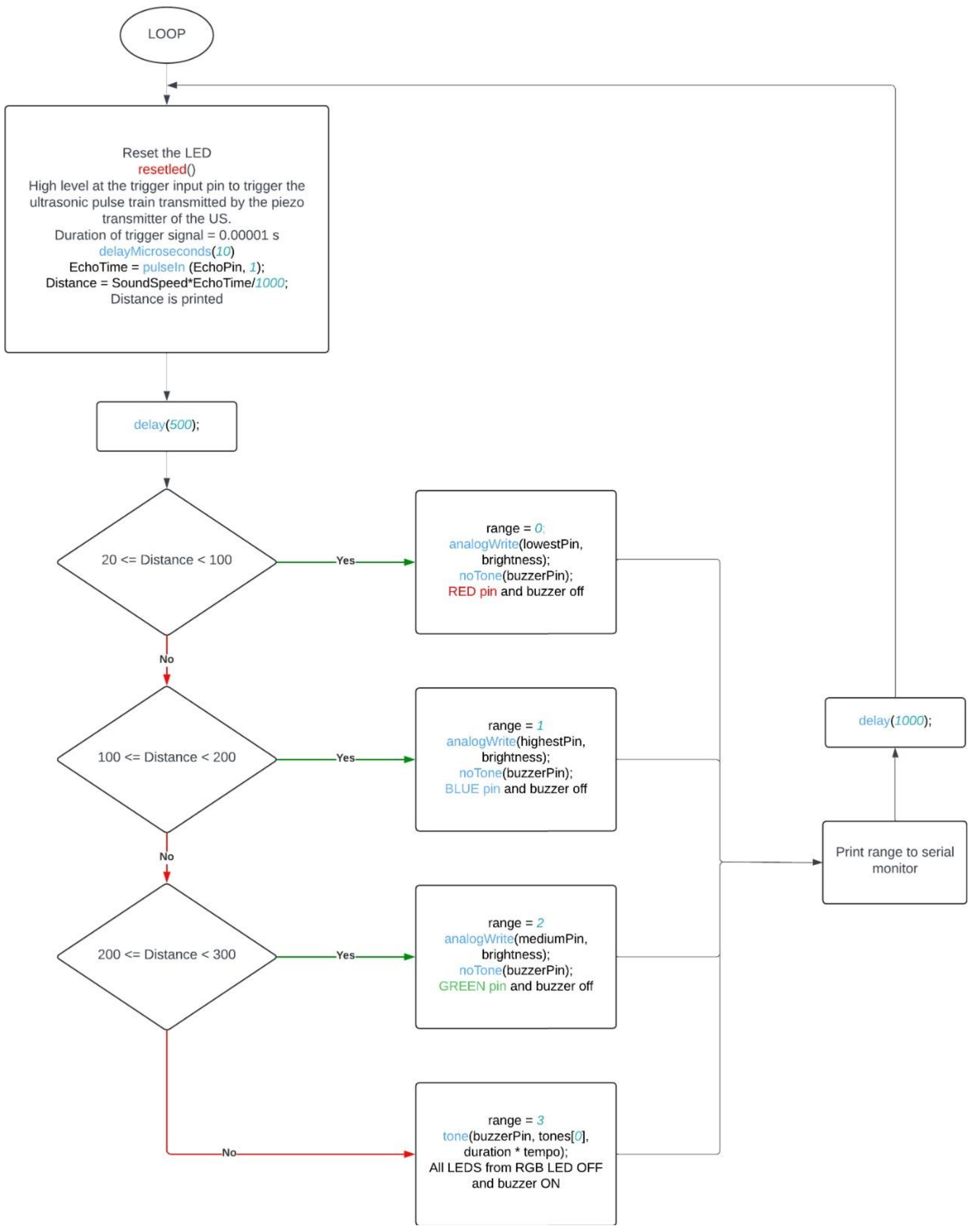

RESETLED()
Set brightness value to zero on the RGB terminals of the LED to reset it before the next distance calculation
END

```
int brightness = 0
analogWrite(highestPin, brightness);
analogWrite(mediumPin, brightness);
analogWrite(lowestPin, brightness);
```



```
Serial.begin(baudrate);
pinMode(buzzerPin, OUTPUT);
pinMode(lowestPin, OUTPUT);
pinMode(mediumPin, OUTPUT);
pinMode(highestPin, OUTPUT);
pinMode(EchoPin, INPUT);
pinMode(TriggerPin, OUTPUT);
digitalWrite(TriggerPin, 0);
```

```
delay(200);
```


LOOP
Reset the LED
resetled()
High level at the trigger input pin to trigger the ultrasonic pulse train transmitted by the piezo transmitter of the US.
Duration of trigger signal = 0.00001 s
delayMicroseconds(10)
EchoTime = pulseIn (EchoPin, 1);
Distance = SoundSpeed*EchoTime/1000;
Distance is printed
delay(500);
20 <= Distance < 100
Yes
range = 0;
analogWrite(lowestPin, brightness);
noTone(buzzerPin);
RED pin and buzzer off
No
100 <= Distance < 200
Yes
range = 1
analogWrite(highestPin, brightness);
noTone(buzzerPin);
BLUE pin and buzzer off
No
200 <= Distance < 300
Yes
range = 2
analogWrite(mediumPin, brightness);
noTone(buzzerPin);
GREEN pin and buzzer off
No
range = 3
tone(buzzerPin, tones[0], duration * tempo);
All LEDS from RGB LED OFF and buzzer ON
Print range to serial monitor
delay(1000);