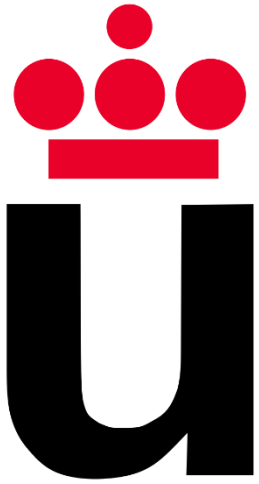


Digital Systems and Microprocessors

Project 4: Pulse Width Modulation (PWM) and Piezoelectric

Enrique Almazán Sánchez
Víctor Miguel Álvarez Camarero

Table of Contents

Objective

The general objective of this project is to demonstrate the use and skill using the Arduino System, and the handling of different types of sensors, actuators and generation of PWM signals.

Description

Connect the LDR and LM35 temperature sensor to the A0 and A1 analog inputs of the Arduino development board, respectively. An RGB LED is used, as well as a LED of another color. Also, a buzzer must be connected to pin 23 of the Arduino development board.

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.
- 1 resistor with a resistance of 1KΩ.
- Protoboard for circuit assembly.
- LDR (Light Dependent Resistor)
- LM 35
- **RGB Led**.
- **PWM Piezoelectric**

The Arduino ATMega2560, in addition to the standard digital outputs that any microcontroller has, has an output that is internally connected to a Pulse Width Modulation, or PWM, generator.

A PWM works like a switch, constantly turning on and off, regulating the amount of voltage and therefore the power delivered to the device or load to be controlled. These devices can be DC motors or DC light sources, among others.

Piezoelectrics are very useful actuators, as they are transducers capable of converting electrical energy into sound.

Exercise 1

Write a code in which light incident on the LDR is measured and displayed on an LED as the luminous intensity of the measured amount of light. The measured incident light will be divided into 6 ranges from 0 to 5, and each range will be divided into 171 units (ADC conversion intervals).

To control the intensity of the light generated by the LED, it must be controlled by a PWM signal generator, so the LED must be connected to one of the 12 pins that generate a PWM signal on the Arduino development board.

When the sensor's output voltage (code generated by the ADC) is between 0 and 171, the LED should be lighting up according to the amount of light the sensor measures, and so on for each range. The minimum value, range 0, is when the incident light on the LDR is maximum and the LED must be lighting at maximum. In the interval of range 5, maximum value, is when the incident light is minimal, that is, there is little or no light, and the LED will light very little or will be off.

The range of the light detected by the LDR should also be displayed on the serial monitor. Therefore, the following message "Measured light range:" should be displayed, after this message the numerical value of the range should be displayed.

The code presented for this exercise is designed to measure the light incident on an LDR sensor and adjust the brightness of an RGB LED (specifically the green) based on the measured light intensity. In this sense if the LDR sensor detects low light the LED will shine in a low intensity and when the LDR sensor detects an intense light the LED will shine in a high intensity.

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.

- “redpin” (constant integer): Pin for the red color of the RGB LED, set to 2.
- “greenpin” (constant integer): Pin for the green color of the RGB LED, set to 3.
- “bluepin” (constant integer): Pin for the blue color of the RGB LED, set to 4
- “analogpin” (constant integer): set to A0, which is the analog input pin connected to the LDR sensor.
- “sensorvalue” (float): initialized to 0, used to store the analog reading from the LDR sensor.
- “brightness” (integer): representing the brightness of the RGB LED, which will be controlled by PWM.
- “cont” (integer): initialized to 0, used to represent the range of light intensity detected by the LDR sensor.
-

```
const int redPin = 2;
const int greenPin = 3;
const int bluePin = 4;
const int analogpin = A0;
const int baudrate = 9600;
float sensorvalue = 0;
int brightness = 0;
int cont = 0;
```

Moreover, in the setup function, all the color pins are configured as outputs ("*pinMode()*") to control the RGB LED brightness, obtaining a purple or violet color from the combination of the three of them (red, green and blue), and begins serial communication with the specified baud rate ("*Serial.begin*"). It continues setting the analog reference pin to default ("*analogReference()*"), indicating that the reference voltage for analog inputs is the default voltage provided by the board. Finally, a delay of 100ms is added.

```
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(baudrate);
  analogReference(DEFAULT);
  delay(100);
}
```

Furthermore, the loop function reads the analog input from pin "A0", from the LDR sensor ("*analogRead()*") and stores the amount of light detected by the sensor in a variable ("*sensorvalue*"). Then, the raw ADC value with some messages, as well as the computed input voltage source based on the sensor value, are printed on the Serial Monitor (*Serial.print*).

```
void loop() {
  sensorvalue = analogRead(A0);

  Serial.print("ADC code = ");
  Serial.println(sensorvalue);
  Serial.print("Input voltage source = ");
  Serial.println(sensorvalue * 5 / 1023);
  delay(200);

      If-else body (seen below)

  analogWrite(redPin, brightness);
  analogWrite(greenPin, brightness);
  analogWrite(bluePin, brightness);

  Serial.print("MEASURED LIGHT RANGE = ");
  Serial.println(cont);
  delay(200);
}
```

As the sensor reading value varies from 0 to 5 volts, the microcontroller internally represents the analog voltage with a 10-bit number, ranging from 0 to 1023. To obtain the real-world voltage, the sensor value is scaled linearly by performing the following operation:

$$sensorvalue * 5 / 1023$$

It continues by determining the range of light intensity based on the sensor value and adjusts the brightness of the LED accordingly using PWM, having after each iteration delays of 200ms. Hence, there is a control of the RGB LED brightness, adjusting the PWM duty cycle according to the 6 different ranges divided into 171 units of sensor values shown below.

```
if ((sensorvalue >= 0) && (sensorvalue<171)){
      cont = 0;
      brightness = 255;
      delay(10);
  }
    else if ((sensorvalue >=171) && (sensorvalue < 342)){
      cont = 1;
      brightness = 210;
      delay(10);
  }
    else if ((sensorvalue >=342) && (sensorvalue < 513)){
      cont = 2;
      brightness = 160;
      delay(10);
  }
    else if ((sensorvalue >=513) && (sensorvalue < 684)){
      cont = 3;
      brightness = 105;
      delay(10);
  }
    else if ((sensorvalue >=684) && (sensorvalue <855)){
      cont = 4;
      brightness = 55;
      delay(10);
  }
    else if ((sensorvalue >=855) && (sensorvalue <= 1023)){
      cont = 5;
      brightness = 0;
      delay(10);
  }
```

The established mechanism to increase the brightness of the RGB LED in response to the amount of light detected by the sensor is based on assigning a brightness value that ranges from 0 to 255. When the sensor detects maximum light (range from 0 to 171), the LED's brightness will be set to the maximum value of 255. Conversely, when the sensor detects minimum light (range from 855 to 1023), the LED's brightness will be very low. In this case, it has been chosen to set it to 0 for clarity when observing the LED on the Arduino breadboard.

The assembly circuit can be seen in the below figure, while the entire implementation in the video ‘Exercise_1’.

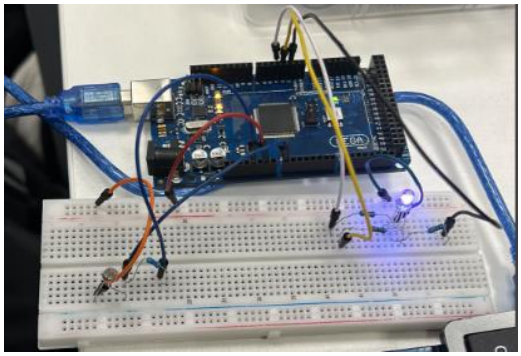


Figure 1: Assembly circuit for a LDR with a RGB LED

Exercise 2

Write a code where the temperature detected by the LM35 is measured and that temperature is displayed on the serial monitor. Two temperature ranges are selected, 25ºC and 30ºC, to establish a minimum and maximum working limit of the system.

When the temperature sensor detects a temperature below the minimum limit, the following message, "Temperature below minimum limit" appears on the serial monitor. In addition, the RGB LED will turn on the red LED and the buzzer will generate a sound corresponding to a musical note chosen by the programmer. While the system is in this state, both the LED and the buzzer should remain on.

When the temperature sensor detects a temperature above the maximum limit, the following message, "Temperature above maximum limit" will be displayed on the serial monitor. In addition, the RGB LED will turn on the blue led and the buzzer will generate a sound corresponding to a different musical note than the one chosen in the previous item. While the system is in this state, both the LED and the buzzer should remain on.

When the temperature sensor detects a temperature between the minimum and maximum limit, the following message, "Temperature normal" appears on the serial monitor. Also, the RGB LED will turn on the green LED and the buzzer will go off. While the system is in this state, the buzzer should remain off.

The code presented for this exercise is designed to monitor the temperature detected by an LM 35 sensor and display it on the serial monitor and adjust both, the RGB LED and a buzzer based on the temperature readings.

First of all, some variables are initialized, as well as the baud rate at 9600. As in the previous exercise the variables “analogpin” (set to `A0’), and “sensorvalue” (float, initialized to 0), but for the LM 35 temperature sensor. Also, the “brightness” parameter is defined (integer, set to 0), as well as the three RGB LED pins, “lowestPin”, mediumPin” and “highestPin” as for this exercise the three will be needed. On the other hand, for the buzzer the following variables are defined:

- “tempo” (integer): defines the speed of the musical notes being played by the buzzer. It is set to 200 milliseconds.
- “notesName” (array): defines the names or identifiers of the musical notes to be played. In this case, it contains the notes “b” and “c”.
- “tones” (array): defines the frequencies (in Hz) corresponding to the musical notes defined in “notesName””. The first element corresponds to the frequency of note “b” (500 Hz), and the second element corresponds to the frequency of note “c” (250 Hz).
- “duration” (integer): defines the duration of each musical note in seconds. It is set to 3 seconds.
- “buzzerPin”: defines the pin number to which the buzzer is connected. In this case, it is set to pin 23.

```
const int analogpin = A0;
const int baudrate = 9600;
float sensorvalue = 0;
const int lowestPin = 2;
const int mediumPin = 3;
const int highestPin = 4;
int brightness = 255;
```

```
// The following variables will be used for the musical notes of the
buzzer
int tempo = 200;
char notesName[] = {'b', 'c'};
int tones[]= {500, 250};
int duration = 3;
int buzzerPin = 23;
```

The setup function also has similarities with exercise 1, starting serial communication with the specified baud rate ("*Serial.begin*") and setting the analog reference pin to the power supply of the board ("*analogReference()*"). In addition, the LED and buzzer pins are initialized as outputs. A delay of 500ms is presented.

```
void setup() {
  Serial.begin(baudrate);
  analogReference(DEFAULT);
  pinMode(lowestPin, OUTPUT);
  pinMode(mediumPin, OUTPUT);
  pinMode(highestPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  delay(500);
}
```

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

```
void loop() {
  resetled();
  sensorvalue = analogRead(A0);

  Serial.print("ADC code = ");
  Serial.println(sensorvalue);
  Serial.print("Inpt voltage source = ");
  float temp = sensorvalue*5/1023;
  Serial.println(temp);
  delay(1000); // delay of 1s

  // If-else body (seen below)

  delay(1000);
}
```

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

```
void resetled(){
  int brightness = 0;
  analogWrite(highestPin, brightness);
  analogWrite(mediumPin, brightness);
  analogWrite(lowestPin, brightness);
}
```

- The analog value from pin A0 (connected to the LM 35 sensor) is read and converted to temperature in Celsius. The temperature reading and corresponding voltage are printed to the serial monitor, after performing a necessary operation:

$$sensorvalue * 5 / 1023$$

Depending on the temperature range shown in the code below, a different colour of the RGB LED will show up, having three different conditional clauses:

```
if (temp<0.25){
   Serial.println("Temperature is below 25 degrees");
   analogWrite(lowestPin, brightness); //RED PIN BRIGHTS
   tone(buzzerPin, tones[0], duration * tempo);
   delay(200);
 }
 else if ((temp>=0.25)&&(temp<=0.3)){
   Serial.println("Temperature is normal");
   analogWrite(mediumPin, brightness); //GREEN PIN BRIGHTS
   noTone(23); // stop the sound
   delay(200);
 }
 else{
   Serial.println("Temperature is above 30 degrees");
   analogWrite(highestPin, brightness);
   tone(buzzerPin, tones[1], duration * tempo); //BLUE PIN BRIGHTS
   delay(200);
 }
```

Finally, there is a delay of 1 second between each temperature measurement for stability.

The assembly circuit can be seen in the below figures, while the entire implementation in the video ‘Exercise_2’, seeing that the temperature staying stable.

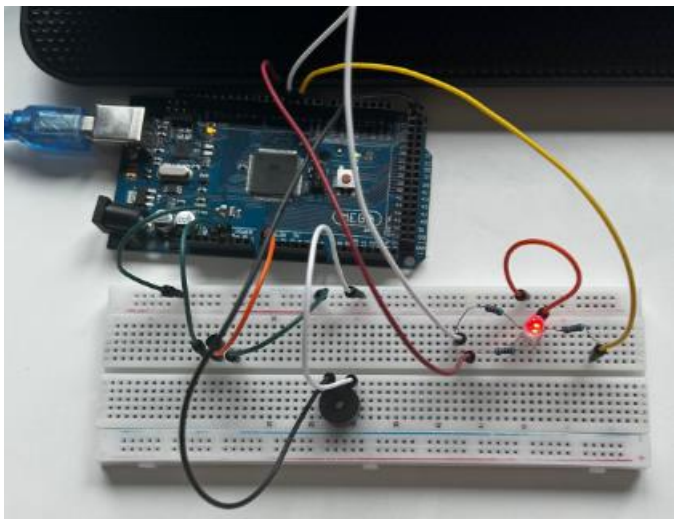


Figure 2: Assembly circuit for a LM 35 sensor with a RGB LED. Below 25 degrees, red LED.

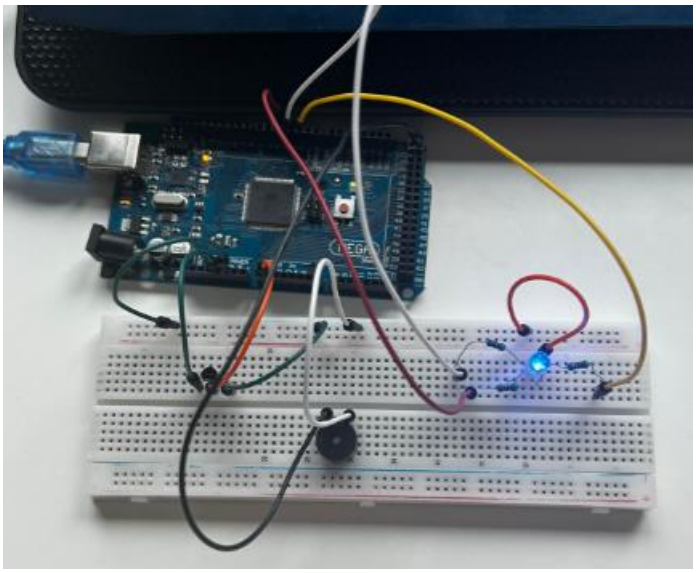
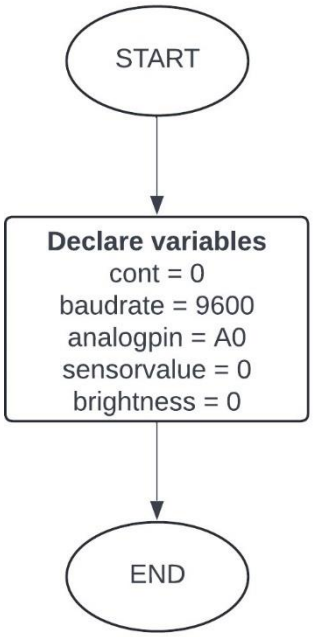
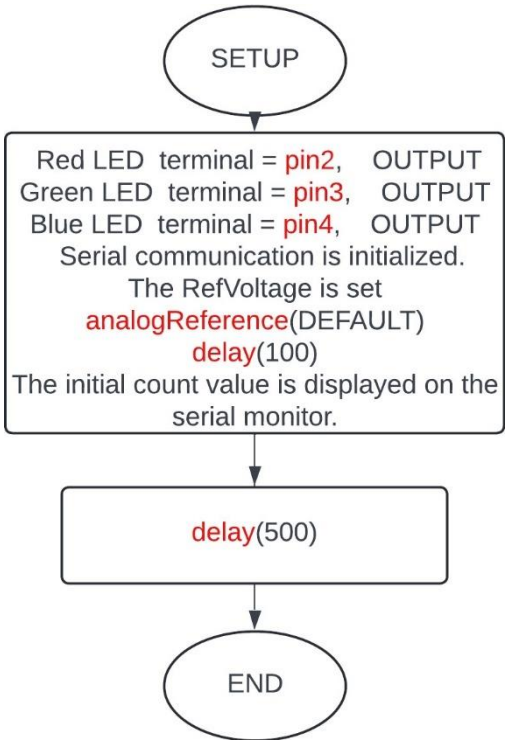


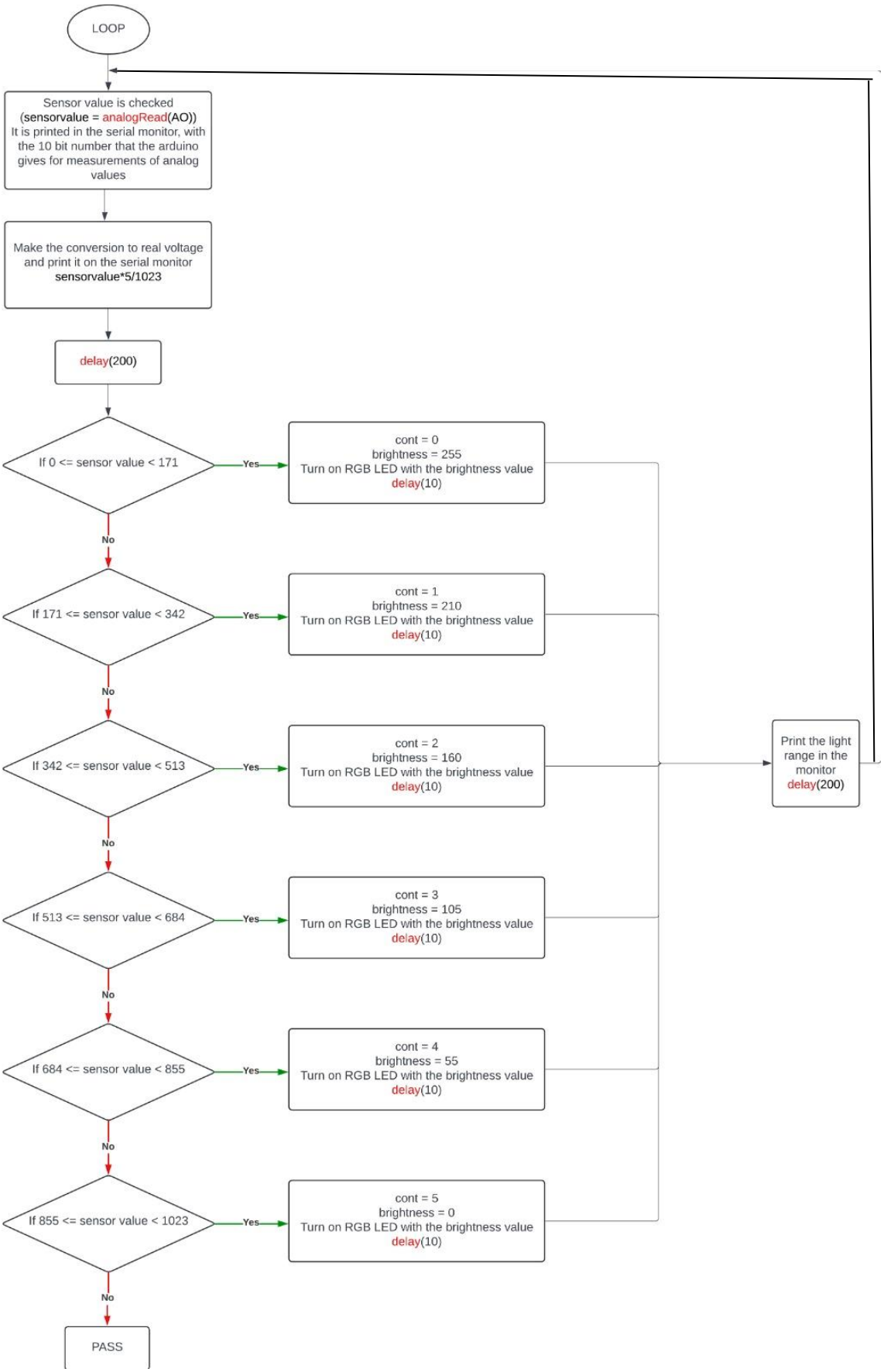
Figure 3: Assembly circuit for a LM 35 sensor with a RGB LED. Above 30 degrees, blue LED.

Flowcharts

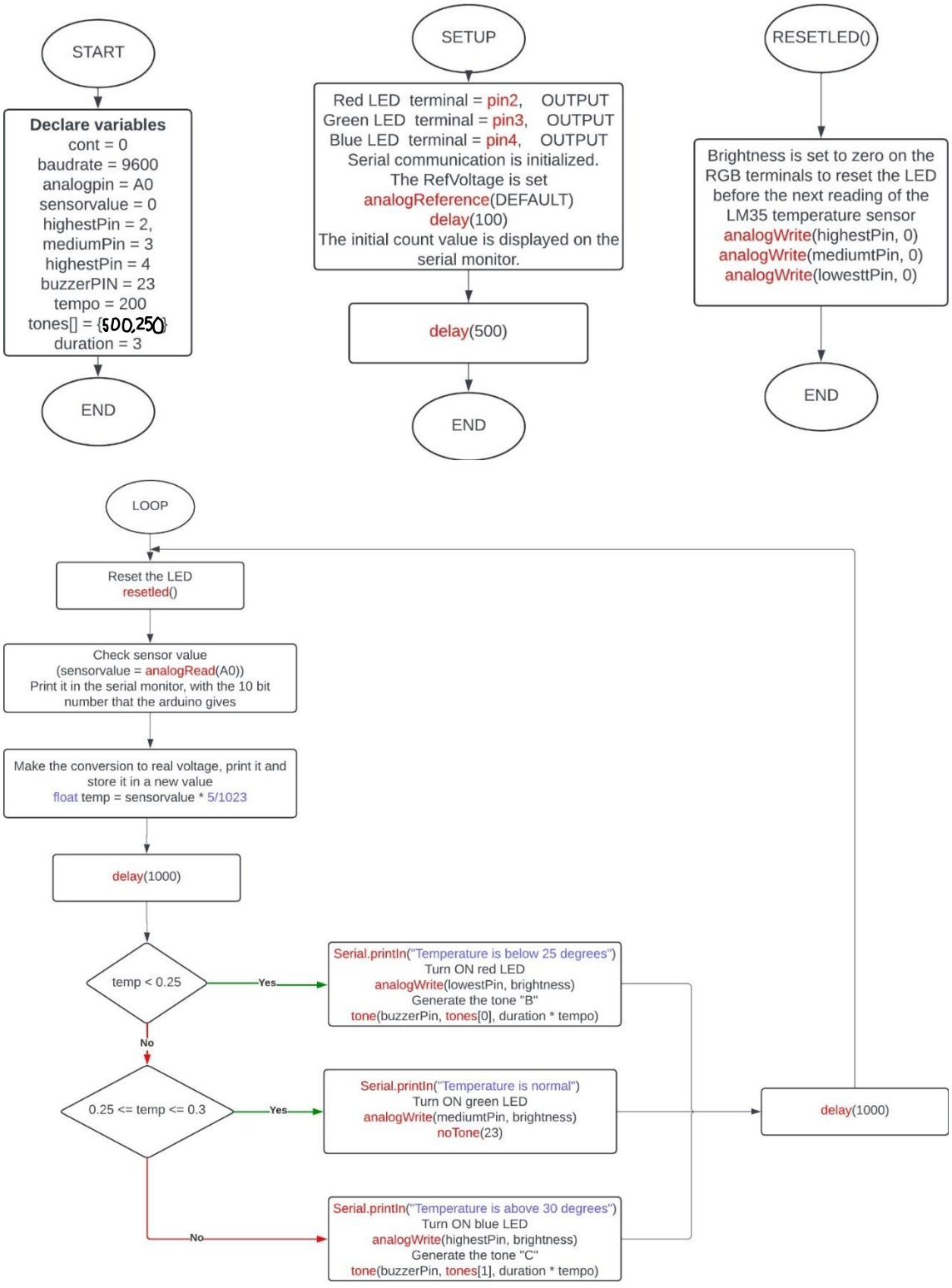
Exercise 1






LOOP
Sensor value is checked
(sensorvalue = analogRead(AO))
It is printed in the serial monitor, with the 10 bit number that the arduino gives for measurements of analog values
Make the conversion to real voltage and print it on the serial monitor
sensorvalue*5/1023
delay(200)
If 0 <= sensor value < 171
Yes
cont = 0
brightness = 255
Turn on RGB LED with the brightness value
delay(10)
No
If 171 <= sensor value < 342
Yes
cont = 1
brightness = 210
Turn on RGB LED with the brightness value
delay(10)
No
If 342 <= sensor value < 513
Yes
cont = 2
brightness = 160
Turn on RGB LED with the brightness value
delay(10)
No
If 513 <= sensor value < 684
Yes
cont = 3
brightness = 105
Turn on RGB LED with the brightness value
delay(10)
No
If 684 <= sensor value < 855
Yes
cont = 4
brightness = 55
Turn on RGB LED with the brightness value
delay(10)
No
If 855 <= sensor value < 1023
Yes
cont = 5
brightness = 0
Turn on RGB LED with the brightness value
delay(10)
No
PASS
Print the light range in the monitor
delay(200)

Exercise 2


START
Declare variables
cont = 0
baudrate = 9600
analogpin = A0
sensorvalue = 0
highestPin = 2,
mediumPin = 3
highestPin = 4
buzzerPIN = 23
tempo = 200
tones[] = {500,250}
duration = 3
END
SETUP
Red LED terminal = pin2, OUTPUT
Green LED terminal = pin3, OUTPUT
Blue LED terminal = pin4, OUTPUT
Serial communication is initialized.
The RefVoltage is set
analogReference(DEFAULT)
delay(100)
The initial count value is displayed on the serial monitor.
delay(500)
END
RESETLED()
Brightness is set to zero on the RGB terminals to reset the LED before the next reading of the LM35 temperature sensor
analogWrite(highestPin, 0)
analogWrite(mediumtPin, 0)
analogWrite(lowesttPin, 0)
END
LOOP
Reset the LED
resetled()
Check sensor value
(sensorvalue = analogRead(A0))
Print it in the serial monitor, with the 10 bit number that the arduino gives
Make the conversion to real voltage, print it and store it in a new value
float temp = sensorvalue * 5/1023
delay(1000)
temp < 0.25
Yes
Serial.println("Temperature is below 25 degrees")
Turn ON red LED
analogWrite(lowestPin, brightness)
Generate the tone "B"
tone(buzzerPin, tones[0], duration * tempo)
No
0.25 <= temp <= 0.3
Yes
Serial.println("Temperature is normal")
Turn ON green LED
analogWrite(mediumtPin, brightness)
noTone(23)
No
Serial.println("Temperature is above 30 degrees")
Turn ON blue LED
analogWrite(highestPin, brightness)
Generate the tone "C"
tone(buzzerPin, tones[1], duration * tempo)
delay(1000)