Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENCS411

Project Report

Arduino in Tinkercad

Prepared by : Tareq Shannak

ID Number : 1181404

Instructor : Dr. Abdallatif Abuissa

Teaching Assistant : Ziad Adeela

Section : 1

Date : 23/11/2020

Abstract

In this project, we will use Tinkercad to implement a circuit that connect two Arduinos using serial cable and dealing with different components which help us in display a result, sensor the temperature, generate a power, sound a tone and etc. the circuit surely have resistors and some little components.

Table of Contents

Theory

Arduino

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. It's a hardware products are licensed under a CC-BY-SA license, while software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL). permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors. A simple Arduino is shown in Figure1.

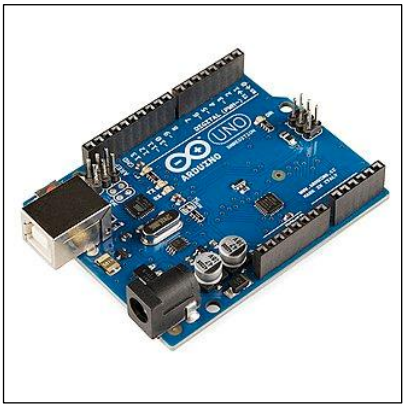

Figure 1 - Simple Arduino

Breadboard

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below in Figure 2. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

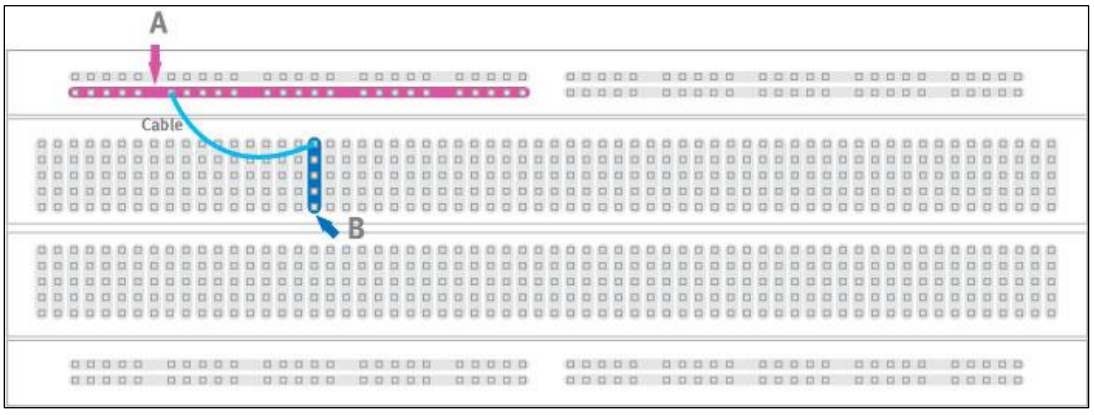

Figure 2 - Breadboard

LCD

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface. It looks like Figure 3.

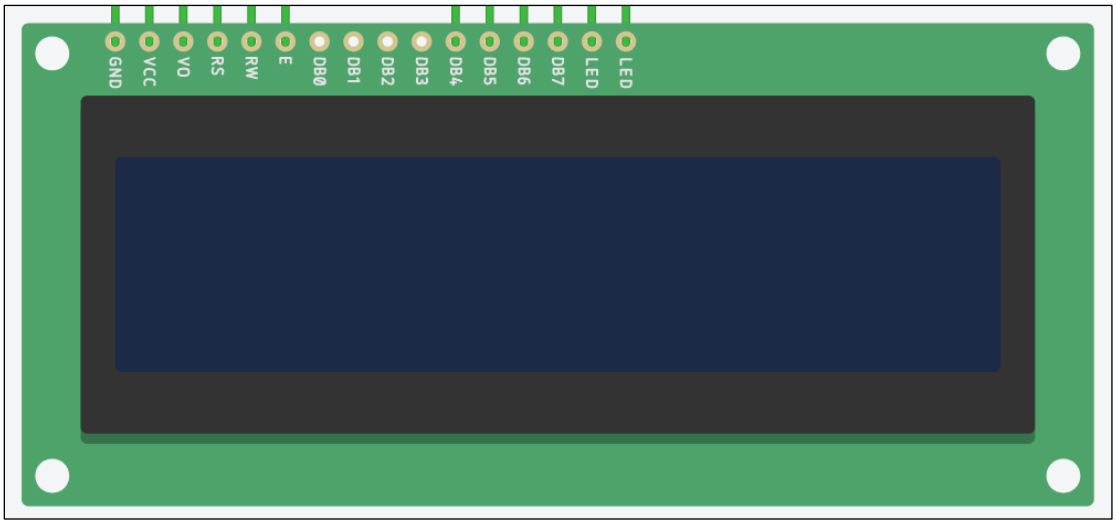

Figure 3 – LCD

There are two modes in LCD: **8-bit mode**: 8 bits of a byte are sent at the same time in pin D0 to D7. **4-bit mode**: 8 bits of a byte is sent two times, each time 4 bits in pin D4 to D7. 8-bit mode is faster than the 4-bit mode, but use more pins than 4-bit mode. We uses 4-bit mode, which is the most common-used. Table 1 shows LCD pins in 4-bit mode as we used.

LCD PIN	Connected to
GND	GND
VCC	5V
V_O	5V
RS	Arduino's Pin
RW	GND
EN	Arduino's Pin
$D_0 – D_3$	Not Connected
$D_4 – D_7$	Arduino's Pin
A	5V
K	GND

Table 1 - LCD pins in 4-bit mode

H-Bridge Motor Driver

The L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control. Figure 4 shows a nearly shape of the H-Bridge, we will use it in controlling the DC motor in our project.

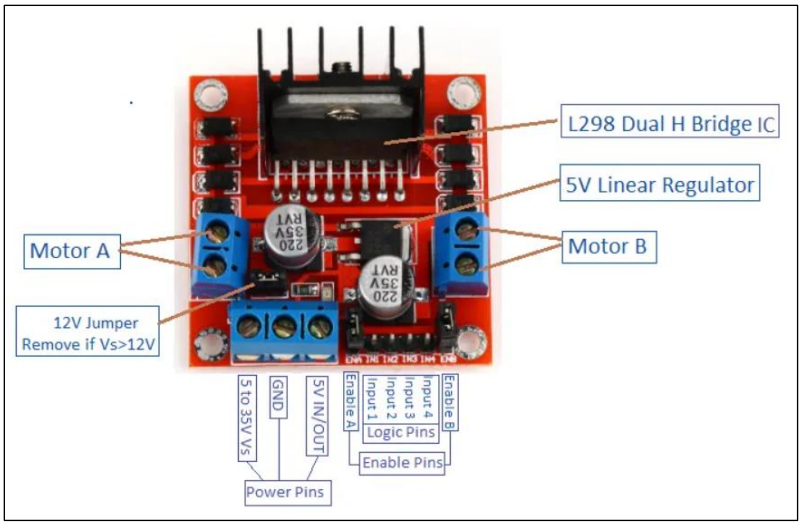

Figure 4 - L298N

DC MOTOR

A DC motor as in Figure 5 (Direct Current motor), is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction. We cannot drive the motor directly from Arduino board pins. This may damage the board, so we used a driver Circuit like H-Bridge.

Figure 5 - DC Motor

Temperature Sensor

The Temperature Sensor LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −40°C to 150°C temperature range. The Temperature Sensor LM35 has 3 pins as shown in Figure 6: ground, Vcc and pin to read the value in Arduino.

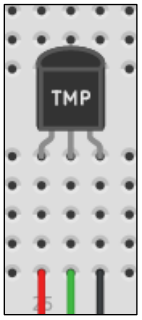

Figure 6 - LM35

Buzzer

Buzzer is an alarm that we will use in our project, we can send a high byte to the buzzer for sounding a sound, or we can send a tone in certain frequency to sound it.

Project Idea

We can use Tinkercad or any other tool we want to implement the project, there are two Arduinos which are connected using serial cable: Arduino#1 is connected to the following components: LCD, DC motor, Temperature Sensor and Arduino#2 is connected to a Buzzer and LED, and there are other components like resistors, etc.

The system should do read the temperature every 1.5 seconds without using 'Delay'. Starting at t=6 seconds, calculate the average temperature (Tavg) during the last 6 seconds. Repeat this process every 1.5 seconds (moving average). LCD will show Tavg every 1.5 second. If the Tavg is larger than 29°C, drive a DC motor (fan) with 20% duty cycle. And then increase the speed of the fan with 10% for every 2°C increase above 29. If the temperature goes below 27 C, turn the DC motor off. If the Temperature become more than 35°C, then the alarm will work on Arduino#2 and the LED will be ON.

Design and Implementation

Circuit

Figure 7 shown how we connect the components with each other. There is a wire between T pin from Arduino#1 to R pin in Arduino#2 and they are been connected with a gnd wire to be serial cable. LCD connected as shown in the figure and as we explained before in 4-bit pin mode. The temperature sensor connected with A_0 pin in Arduino#1. DC motor connected with H-Bridge which connected with Arduino#1. The buzzer is connected with Arduino#2 and also a LED with resistors.

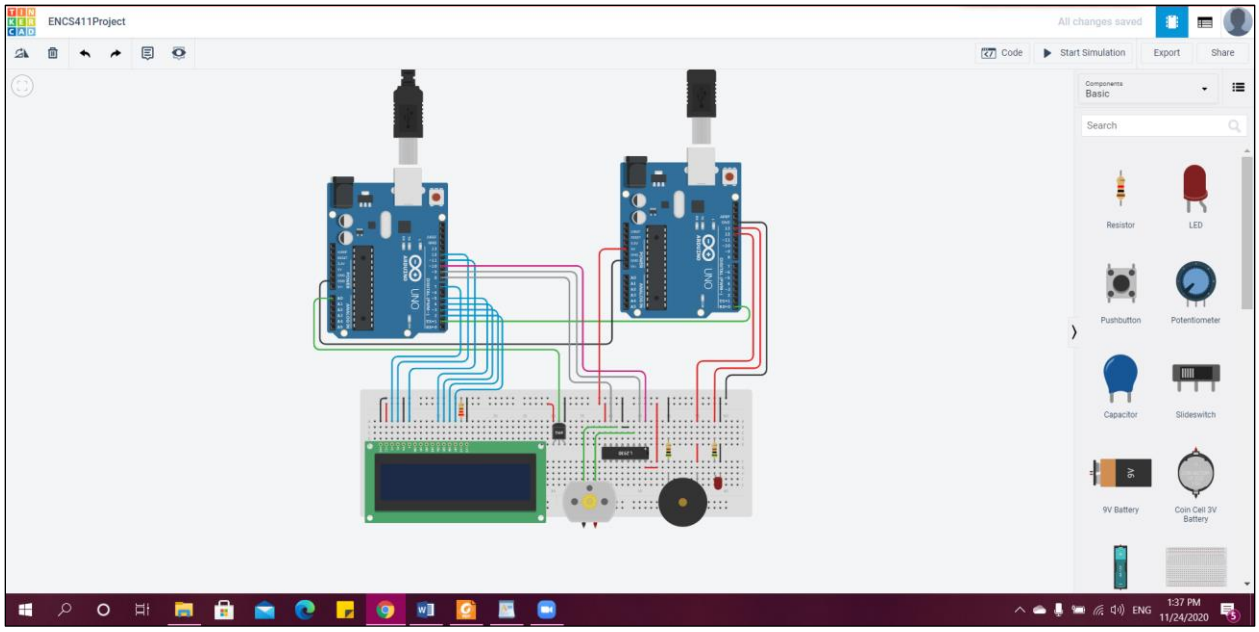

Figure 7 – Circuit

Arduino 1

Figure 8 shows the code in Arduino 1, and the appendix includes it.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int numOfTmpChecks = 0;
unsigned int lastMillis = 0;
float TmpChecks[4];
float tavg;
int baseDutyCycle;
int Extra;

void setup() {

  lastMillis = millis();

  //DC motor pinmodes
  pinMode(10, OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(8,OUTPUT);
  digitalWrite(9,HIGH);
  digitalWrite(8,LOW);

  // LCD pin mode
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW);
  // LCD settings
  lcd.begin(16, 4);
  lcd.setCursor(0, 0);

  //TMP pinmode
  pinMode(A0, INPUT);

  Serial.begin(9600);
}

void loop() {

  //for every 1.5 seconds been past
  if(millis() - lastMillis >= 1500){

    lastMillis = millis();
    //if 6 seconds have not been past, increment numOfTmpChecks
    numOfTmpChecks++;

    //if 6 seconds has not been passed since the start of program
    if(numOfTmpChecks < 5){
      //store the temperature read to array
      TmpChecks[numOfTmpChecks - 1] = ((analogRead(A0)/1020.0) * 500) - 50;
    }
    //if 6 seconds has been passed since the start of program
    else{
      //shift the temperature array to left by 1
      for(int i = 0; i < 3; i++)
        TmpChecks[i] = TmpChecks[i + 1];
      TmpChecks[3] = ((analogRead(A0)/1020.0) * 500) - 50;
    }

    //This 'if condition' just to calculate tavg and display it,
    //only when the first 6 seconds hasn't been passed
    if(numOfTmpChecks <= 3){
      tavg = 0;
      for(int i = 0 ; i < numOfTmpChecks ; i ++)
        tavg += TmpChecks[i];
      tavg /= numOfTmpChecks;
    }else{ //after 6 seconds, for every past 1.5 seconds find out the average
        //calculating avg temperature
        tavg = 0;
        for(int i = 0 ; i < 4 ; i ++)
            tavg += TmpChecks[i];
        tavg /= 4;

        //if average temperature us greater than 35
        if(tavg > 35){
            //send ON string to arduino2 connected serially
            Serial.print("ON\n");
        }else{
            //send OFF string to arduino2 connected serially
            Serial.print("OFF\n");
            //if average temperature is greater than 29..
            if(tavg > 29){
                //base duty cycle of DC fan is 20%
                baseDutyCycle = 0.20 * 255;
                //for each increase by 2 degree in tavg temp above 29
                // increment the base duty cycle of fan by 10%
                Extra = 0.10 * 255 * ((tavg - 29)/2);

                //set the duty cycle of fan to baseDutyCycle
                analogWrite(10, baseDutyCycle + Extra);
            }
            //if average temperature is less than 27
            else if(tavg < 27){
                //stop the DC motor or set the duty cycle to 0%
                analogWrite(10, 0);
            }
        }
    }
    //clearing the lcd
    lcd.clear();

    //showing the avg temperature on lcd
    lcd.print("Tavg = " + String(tavg) + "C");
  }
}
```

Figure 8 - Arduino Code

Explicit Code

Lines (1 - 8): Defines a library and variables to use them, also define LCD with its pins.

Line (12): Store the current time in a variable in milliseconds.

Lines (15 - 19): Define 3 pins as an output to the dc motor using H-Bridge, there is a low logic pin and high logic pin, these two pins select the direction of rotation, so we need to make them fixed because we don't need to change the direction. The third pin select the speed of DC motor with ration in duty cycle.

Lines (22 - 26): define a pin to the LCD in low logic to enable displaying on the LCD, set the initial cursor on zero and where to begin.

Line (29): define a pin to the temperature sensor to read the temperature.

Line (31): setting up the baud rate.

Line (37): check if a 1.5 second has been passed or not.

Lines (39 - 41): update the current time in the variable and increment the number of checks.

Lines (44 -47): if a 6 seconds has not been passed, store the new check in the empty array.

Lines (49 - 54): if more than 6 seconds has been passed, shift the array to make an empty place to the new check.

Lines (58 - 62): It's an **extra instructions** from me to display the temperature average even a 6 seconds has not been passed.

Lines (63 - 68): calculate the temperature average of the full array elements.

Lines (71 - 73): if the average is more than 35℃, the serial will print and send the string "ON".

Lines (74 - 93): if the average is less than 35℃, the serial will print and send the string "OFF". If the average is more than 29℃, turn on the DC motor with speed 20% of the duty cycle and increase by 10% on every 2℃ above the degree 29℃. If the average is less than 27℃, turn off the DC motor.

Lines (96 - 99): update the info on the LCD with the new temperature average (it happens every 1.5 seconds).

Arduino 2

Figure 9 shows the code in Arduino 2, and the appendix includes it. These Arduino is connected with a buzzer, LED and it supply the breadboard with supply power 5V and GND.

```
1  void setup() {
2
3   //buzzer and led pinmodes
4    pinMode(12, OUTPUT);
5    pinMode(13, OUTPUT);
6
7    Serial.begin(9600);
8  }
9
10 void loop() {
11
12     if(Serial.available() > 0){
13
14         String str = Serial.readStringUntil('\n');
15         Serial.println(str);
16         if(str == "ON"){
17
18             //turn on
19             digitalWrite(13, HIGH);
20             tone(12, 1000);
21         }else if(str == "OFF"){
22
23             //turn off
24             digitalWrite(13, LOW);
25             noTone(12);
26         }
27     }
28 }
```

Figure 9 - Arduino Code

Explicit Code

Lines (1 - 8): setup the Arduino, define two output pins, one for the buzzer and the other for the LED and setting up the baud rate.

Lines (14 - 15): read the message from the first Arduino and print it

Lines (16 - 20): if the message is ON, turn on the LED and make a tone in the buzzer with frequency 1K.

Lines (21 - 26): if the message is OFF, turn off the LED and delete the tone in the buzzer.

Testing

Simulation

I had recorded a video screen to the simulation and take screenshots from the video as I attached below, and there is a table after the figures explain the results.

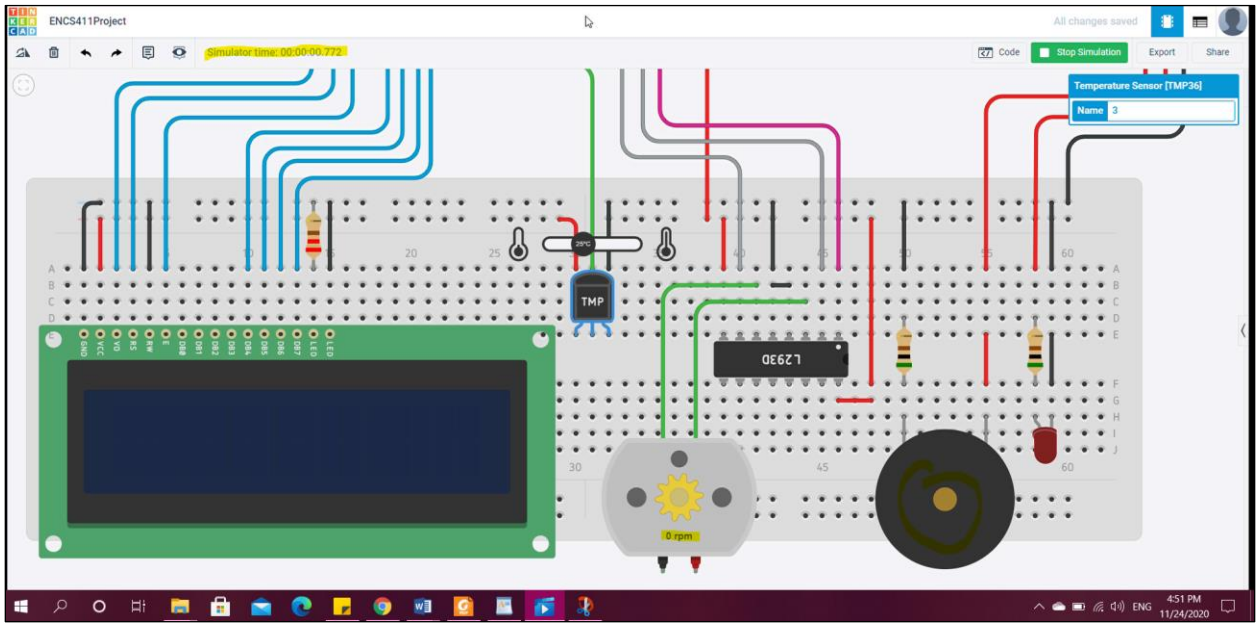

Figure 10 - t=0

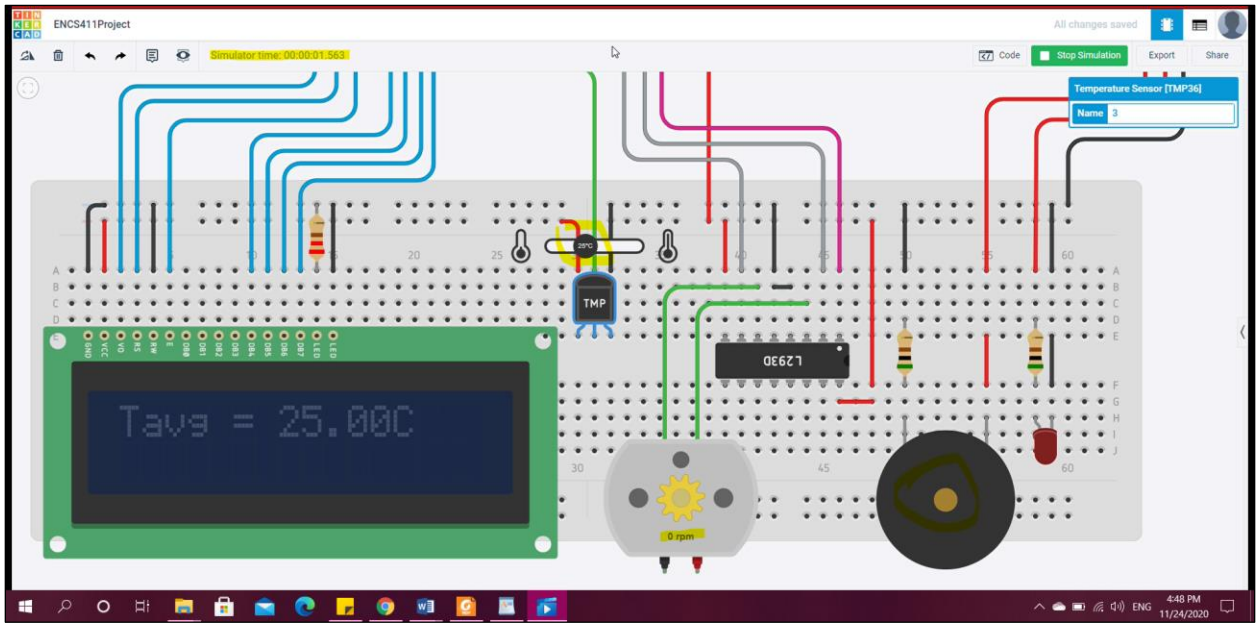

Figure 11 - t =1.5 sec

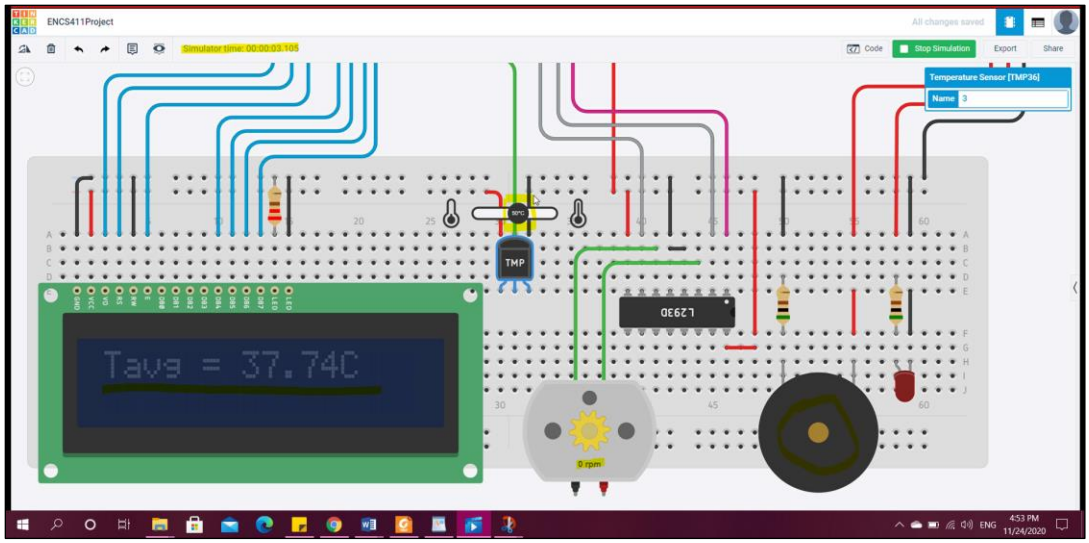

Figure 12 - t = 3 sec

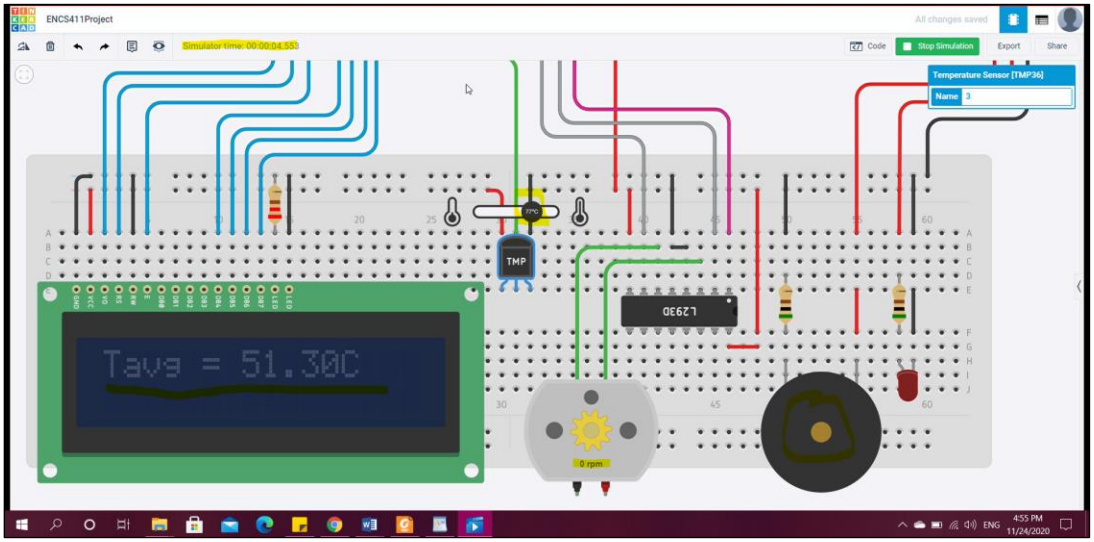

Figure 13 - t = 4.5 sec

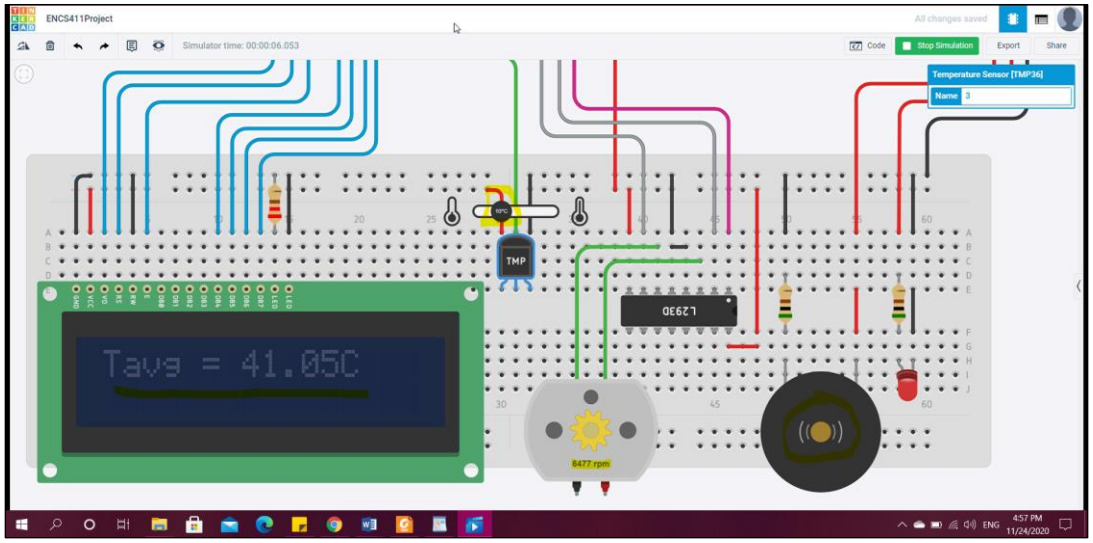

Figure 14 - t = 6 sec

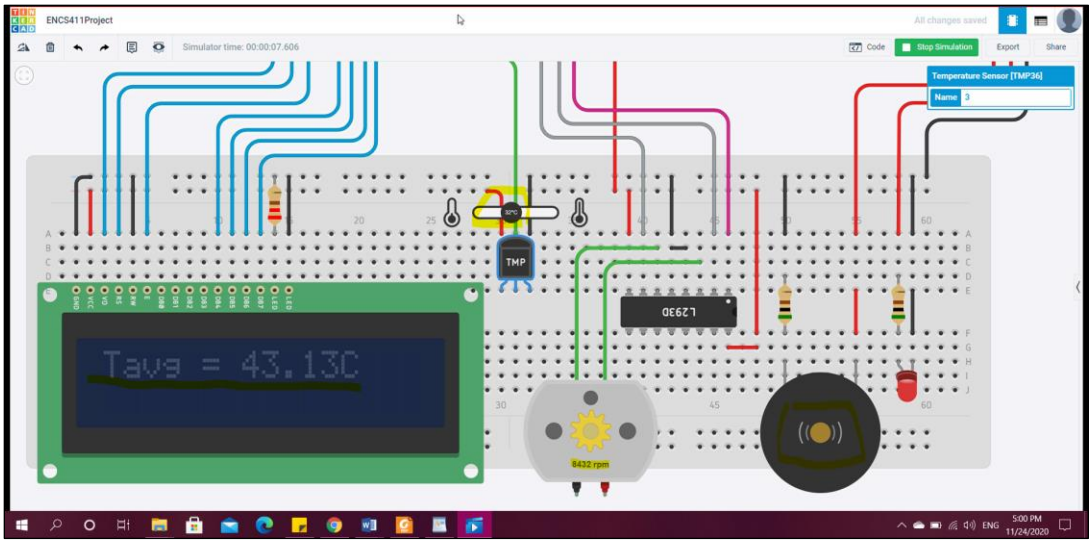

Figure 15 - t = 7.5 sec

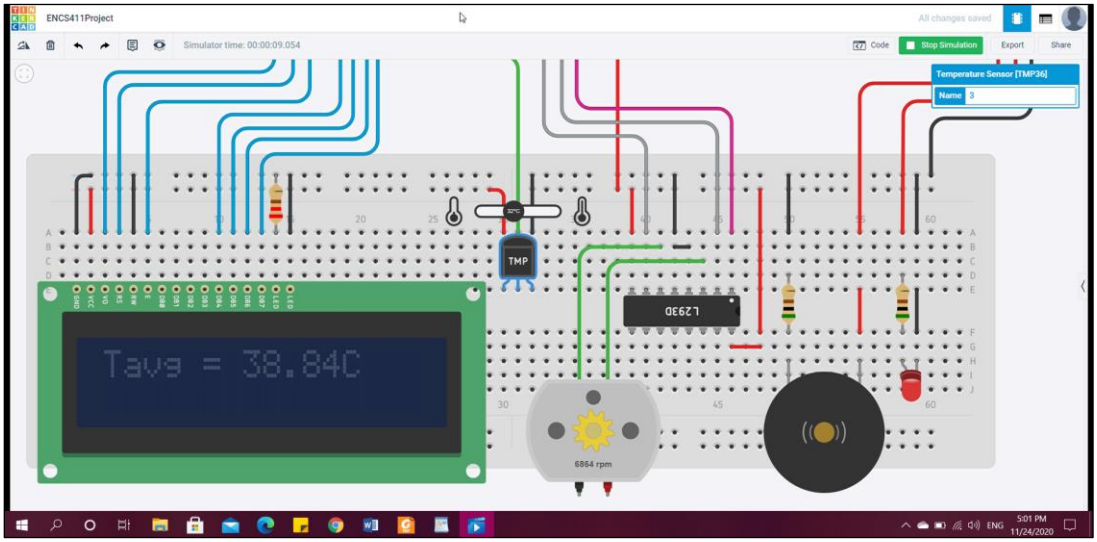

Figure 16 - t = 9 sec

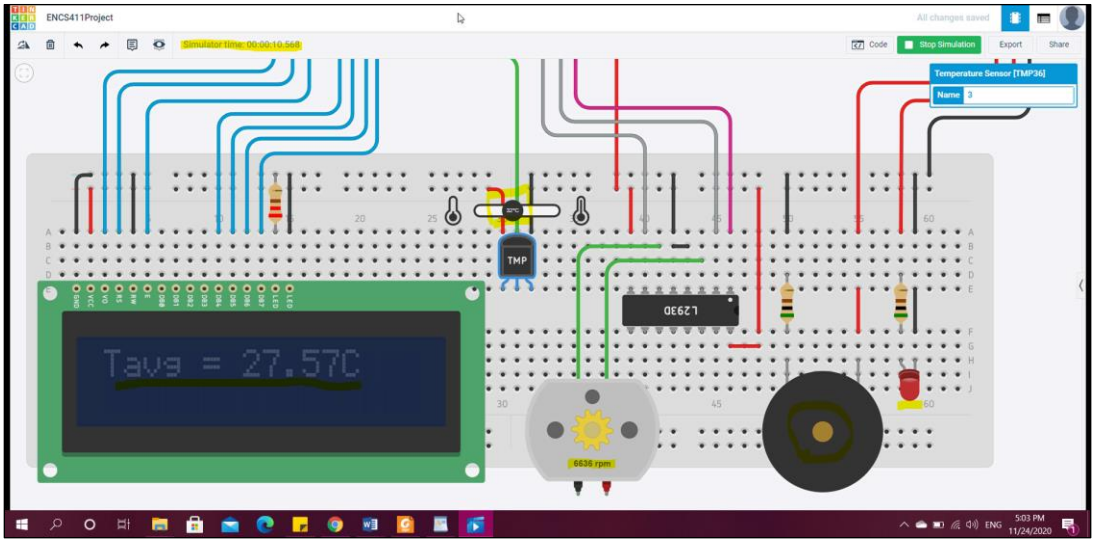

Figure 17 - t = 10.5 sec

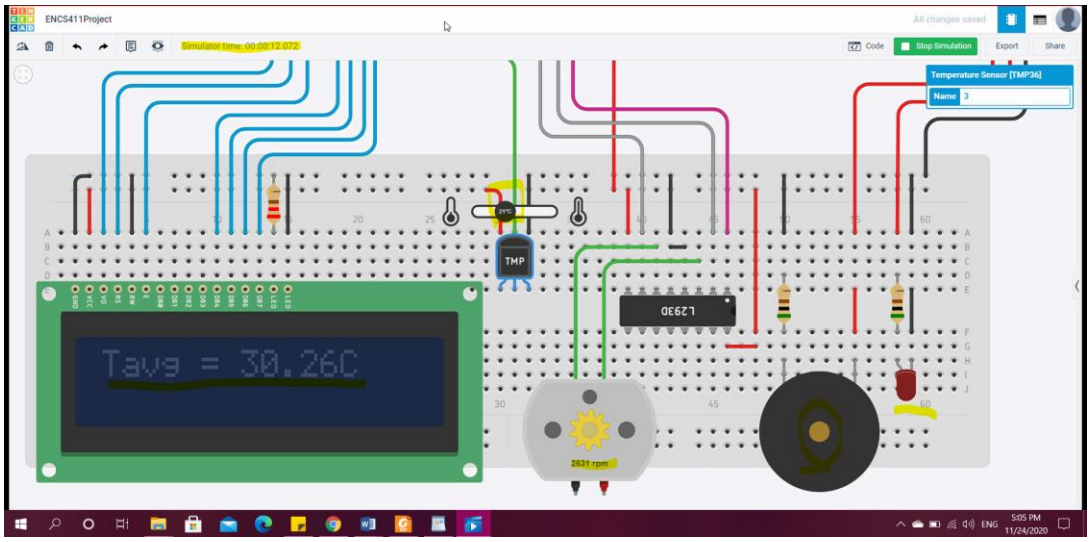

Figure 18 - t = 12 sec

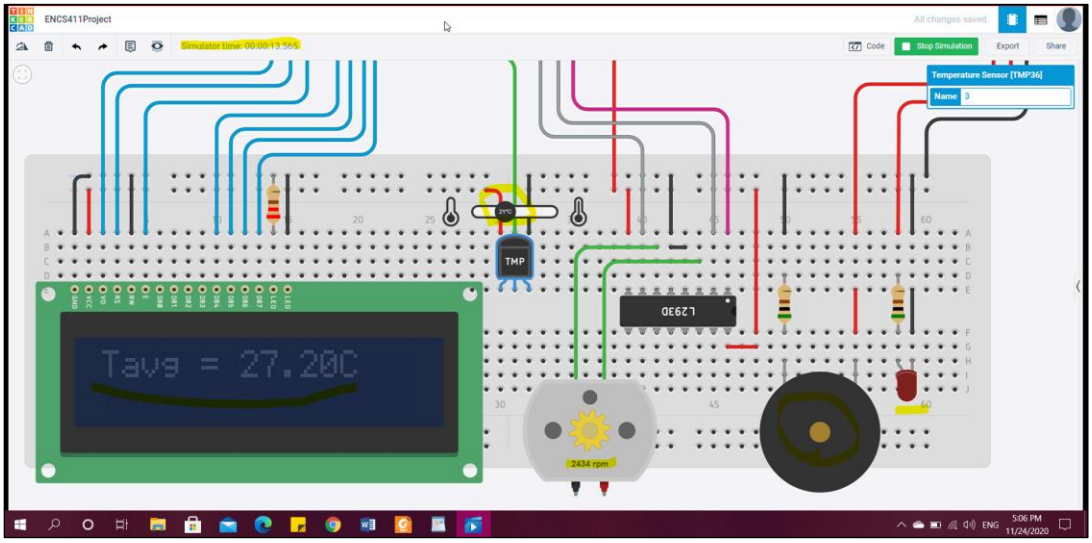

Figure 19 - t = 13.5 sec

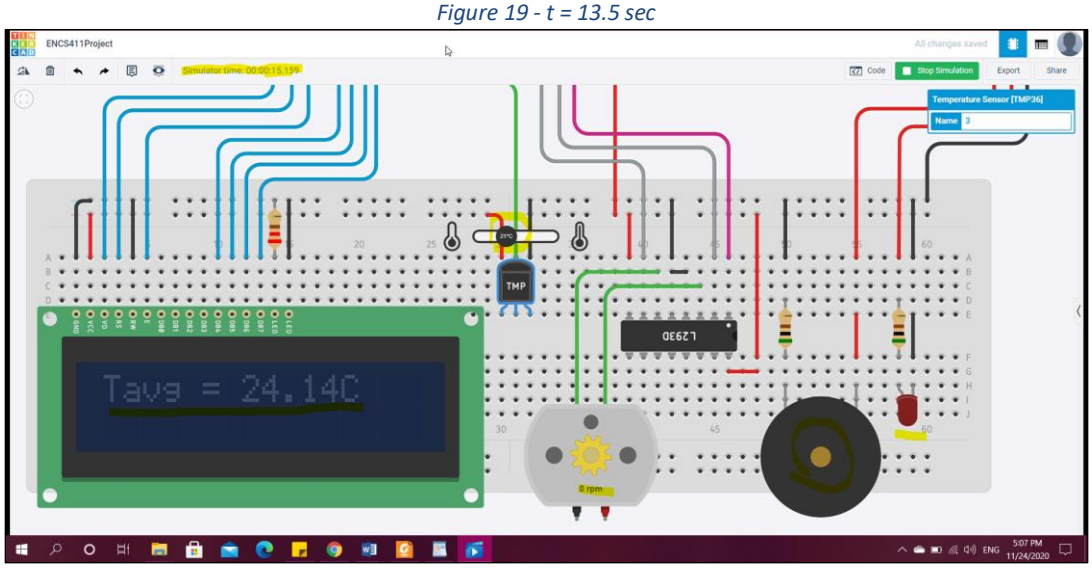

Figure 20 - t = 15 sec

Testing Results

The results in Table 2 are reasonable and correct.

Time(sec)	Temperature(˚C)	T_{avg}	DC motor(rpm)	Buzzer	LED
T = 0.0	25	-	0	OFF	OFF
T = 1.5	25	25.00	0	OFF	OFF
T = 3.0	50	37.74	0	OFF	OFF
T = 4.5	77	51.30	0	OFF	OFF
T = 6.0	10	41.05	6477	ON	ON
T = 7.5	32	43.13	8432	ON	ON
T = 9.0	32	38.84	6864	ON	ON
T = 10.5	32	27.57	6636	OFF	OFF
T = 12.0	21	30.26	2631	OFF	OFF
T = 13.5	21	27.20	2434	OFF	OFF
T = 15.0	21	24.14	0	OFF	OFF

Table 2 - Results

Conclusion

The results that were obtained from the previous testing agree with the theoretical results. Moreover, we conclude that we can construct a lot of jobs using Arduino, we learned how to treat with Arduino and how much it useful in sending signals and deal with the other components.

The following objectives were completed: understand how the devices work, how we can connect the buzzer and LED, how to send a tone to the buzzer, how to deal with the temperature sensor and read the degree, how to control the DC motor using H-Bridge, treat with Tinkercad to simulate circuits and designs, know the basic concepts of serial communication and to practically apply it using Arduino.

We learned how to deal with the time without using DELAY function. Instead, we used MILLIS() that returns how much milliseconds from the beginning of the simulation to the current time, we become more familiar with coding for serial monitor and familiar with the functions.

I tried too hard to understand this part of the lab because I'm not familiar with this material and neither in ENCS338 nor ENCS238 it had been teached to me. I can't put the project's link in references because it will stay private for prevent cheating from other students.

References

- https://core-electronics.com.au/tutorials/use-lcd-arduino-uno.html Accessed on 23/11 12:56PM
- https://create.arduino.cc/projecthub/trduunze/temperature-sensor-36c420 Accessed on 23/11 2:10PM
- https://arduinogetstarted.com/tutorials/arduino-temperature-sensor-lcd Accessed on 23/11 2:12PM
- https://www.programmingelectronics.com/an-easy-way-to-make-noise-with-arduino-using-tone/ Accessed on 23/11 3:14PM
- https://drive.google.com/drive/folders/1btqPqSjwu9DLWGUNQ5wxYmYjr8iB_ZD4 Accessed on 23/11 9:21PM
- https://www.ardumotive.com/how-to-use-a-buzzer-en.html Accessed on 23/11 10:02PM
- https://en.wikipedia.org/wiki/Arduino Accessed on 24/11 12:13AM
- http://wiring.org.co/learning/tutorials/breadboard/ Accessed on 24/11 12:33AM
- https://arduinogetstarted.com/tutorials/arduino-lcd Accessed on 24/11 12:19AM
- https://components101.com/modules/l293n-motor-driver-module Accessed on 24/11 12:40AM
- https://www.tutorialspoint.com/arduino/arduino_dc_motor.htm Accessed on 24/11 1:00AM
- Micro Lab Manual
- Barry B. Brey - The Intel Microprocessors-Prentice Hall (2008)
- ENCS338 Lecture Notes And Slides
- Recorded videos of the experiments

Appendix

Arduino 1

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int numOfTmpChecks = 0;
unsigned int lastMillis = 0;
float TmpChecks[4];
float tavg;
int baseDutyCycle;
int Extra;

void setup() {

  lastMillis = millis();

  //DC motor pinmodes
  pinMode(10, OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(8,OUTPUT);
  digitalWrite(9,HIGH);
  digitalWrite(8,LOW);

  // LCD pin mode
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW);
  // LCD settings
  lcd.begin(16, 4);
  lcd.setCursor(0, 0);

  //TMP pinmode
  pinMode(A0, INPUT);

  Serial.begin(9600);
}

void loop() {

  //for every 1.5 seconds been past
  if(millis() - lastMillis >= 1500){

    lastMillis = millis();
    //if 6 seconds have not been past, increment numOfTmpChecks
    numOfTmpChecks++;
```

```
//if 6 seconds has not been passed since the start of program
if(numOfTmpChecks < 5){
  //store the temperature read to array
  TmpChecks[numOfTmpChecks - 1] = ((analogRead(A0)/1020.0) * 500) - 50;
}
//if 6 seconds has been passed since the start of program
else{
  //shift the temperature array to left by 1
  for(int i = 0; i < 3; i++)
    TmpChecks[i] = TmpChecks[i + 1];
  TmpChecks[3] = ((analogRead(A0)/1020.0) * 500) - 50;
}

//This 'if condition' just to calculate tavg and display it,
//only when the first 6 seconds hasn't been passed
if(numOfTmpChecks <= 3){
  tavg = 0;
  for(int i = 0 ; i < numOfTmpChecks ; i ++)
    tavg += TmpChecks[i];
  tavg /= numOfTmpChecks;
}else{ //after 6 seconds, for every past 1.5 seconds find out the average temperature
        //calculating avg temperature
        tavg = 0;
        for(int i = 0 ; i < 4 ; i ++)
              tavg += TmpChecks[i];
        tavg /= 4;

           //if average temperature us greater than 35
        if(tavg > 35){
        //send ON string to arduino2 connected serially
        Serial.print("ON\n");
        }else{
                  //send OFF string to arduino2 connected serially
        Serial.print("OFF\n");
    }
                  //if average temperature is greater than 29..
                  if(tavg > 29){
                            //base duty cycle of DC fan is 20%
                  baseDutyCycle = 0.20 * 255;
                  //for each increase by 2 degree in tavg temp above 29
                  // increment the base duty cycle of fan by 10%
                  Extra = 0.10 * 255 * ((tavg - 29)/2);

                  //set the duty cycle of fan to baseDutyCycle
                  analogWrite(10, baseDutyCycle + Extra);
       }
```

```
                    //if average temperature is less than 27
                else if(tavg < 27){
                //stop the DC motor or set the duty cycle to 0%
                    analogWrite(10, 0);
        }
   }
   //clearing the lcd
   lcd.clear();

   //showing the avg temperature on lcd
   lcd.print("Tavg = " + String(tavg) + "C");
  }
}
```

Arduino 2

```
void setup() {

 //buzzer and led pinmodes
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

  Serial.begin(9600);
}

void loop() {

        if(Serial.available() > 0){

        String str = Serial.readStringUntil('\n');
                Serial.println(str);
        if(str == "ON"){

        //turn on
        digitalWrite(13, HIGH);
        tone(12, 1000);
        }else if(str == "OFF"){

                //turn off
                digitalWrite(13, LOW);
                noTone(12);
      }
   }
}
```