SWE30011 – INDIVIDUAL ASSIGNMENT
(PRACTICAL)

PROJECT NAME
TEMPERATURE SENSOR

LE BAO DUY NGUYEN

102449993

Table of Contents

LINK TO PROJECT VIDEO:

https://drive.google.com/drive/folders/1MVCMzxOz4SpSY1Q9Ztx2Frq7kuQWsCmW?usp=sharing

1. Summary

1.1. Topic Background

IoT is receiving more and more popularity as we are heading to a more modernised society. Normal objects are combined, hardware and software, to create smart object that can behave and serve our desire and purpose. IoT is a system that consists of hardware (smart objects) and software that can communicate with each other directly.

An IoT system consists of:

- Sensor (Publisher): A device or module that detects the surrounding and send the information signal to the processor.
- Actuator (Subscriber): Usually a motor, fan, etc. Output the action based on given information.
- Data: Data from the sensor are stored within the cloud and used for processing, analysing, and monitoring.
- Server: Data is transferred to server for hosting and processing. Server can transmit data to subscriber or a system through a request (e.g., API).

1.2. Proposed System

The project is inspired by real-world use case of a smoke/temperature detector. The idea is if temperature reaches above a certain threshold, user will be notified by the buzzer and can take precaution accordingly.

This project is using Arduino board and a DHT11 temperature sensor to record surrounding temperature and transmitted into the MariaDB stored in virtual machine Raspberry Pi. The data will then be fetched using Python into localhost web interface that displays data of temperature as well as analytical data (min, max, median, mean).

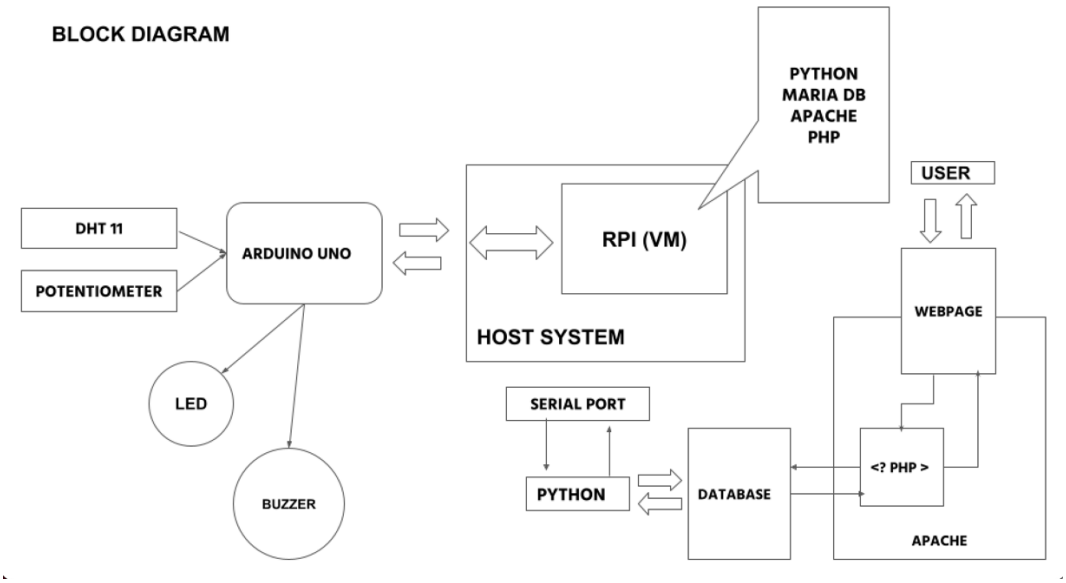
Virtual Raspberry Pi is a Linux Debian 64-bit environment. Python is used to fetch data and create automation rule for the project. PHP is used to design the web interface.

To enhance data visualisation, different analytical data is added within the web interface. These are minimum, maximum, mean and median of the recorded temperature. By including data within the web interface, user can take action and design use cases based on collected temperatures.

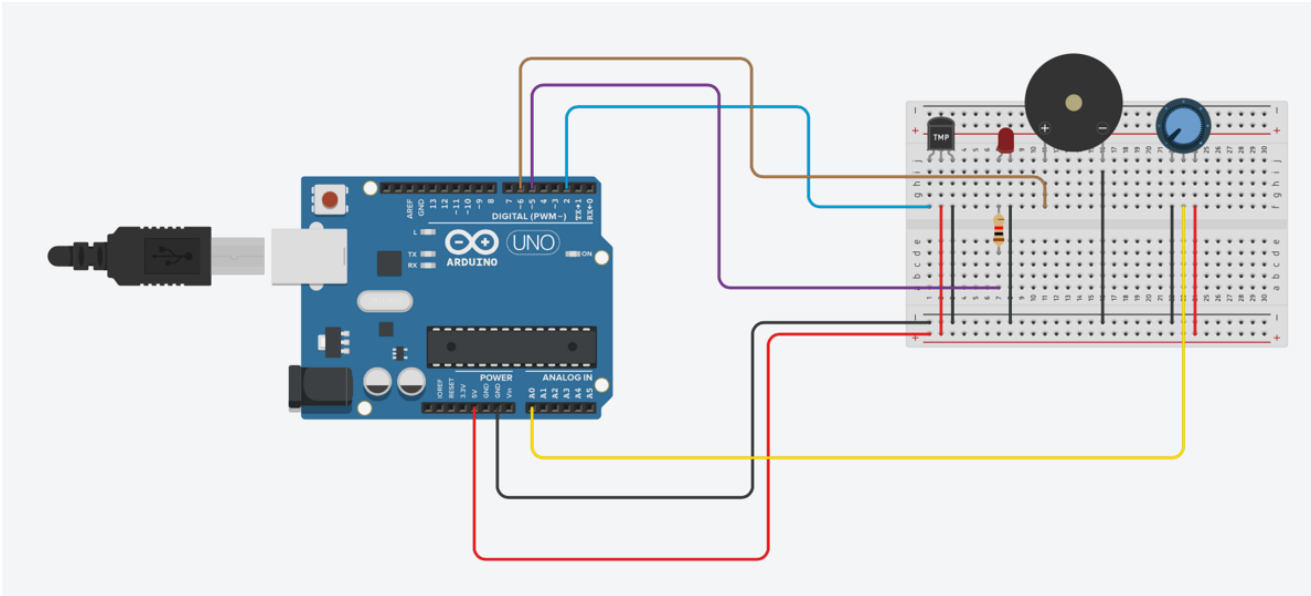
Actions user can take in the web interface include turning up/down volume of warning buzzer. Option buttons are only enabled when temperature reaches over a certain threshold. Also, the LED will turn on when the system is up and running, and the potentiometer can be used to adjust the brightness.

2. Conceptual Design

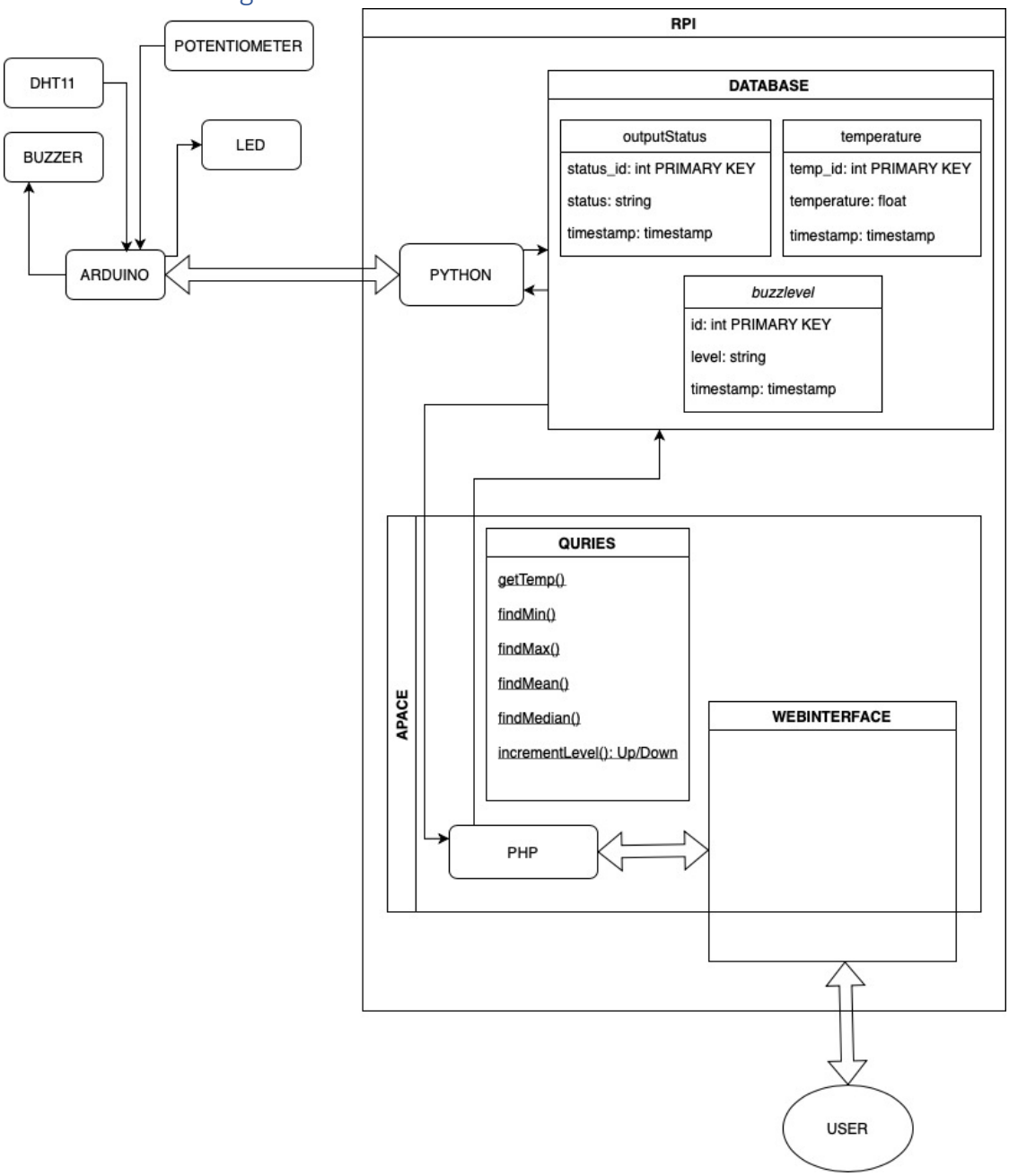
2.1. Block Diagram



2.2. IOT Node Diagram



2.3. UML Diagram



3. Implementation

3.1. Sensor

Digital sensor DHT11: Measure temperature input. It can measure from 0 – 50 degrees C with an accuracy of +/-2. This temperature acts as a publisher data and records it in the database.
Analog sensor Potentiometer: Used to increase/decrease LED brightness.

3.2. Actuator

LED light: If the system is connected, LED is on. Its brightness level is controlled by the potentiometer.
Buzzer: If temperature reaches above a certain threshold (i.e., 20 degrees Celsius) and under a certain threshold (i.e., 15 degrees Celsius), buzzer projects sound automatically. The sound level can be controlled in the web interface. When temperature is in between, user can manually turn on/off the buzzer.

3.3. Virtual Machine Raspberry Pi

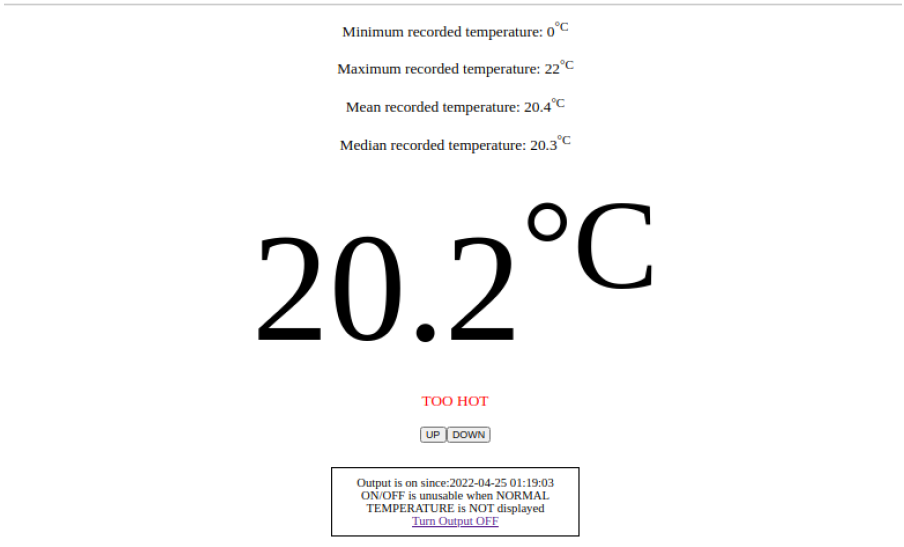
Raspberry Pi acts as a host system that hosts the MariaDB database and the web interface. Python script is run to create the data as well as fetch.

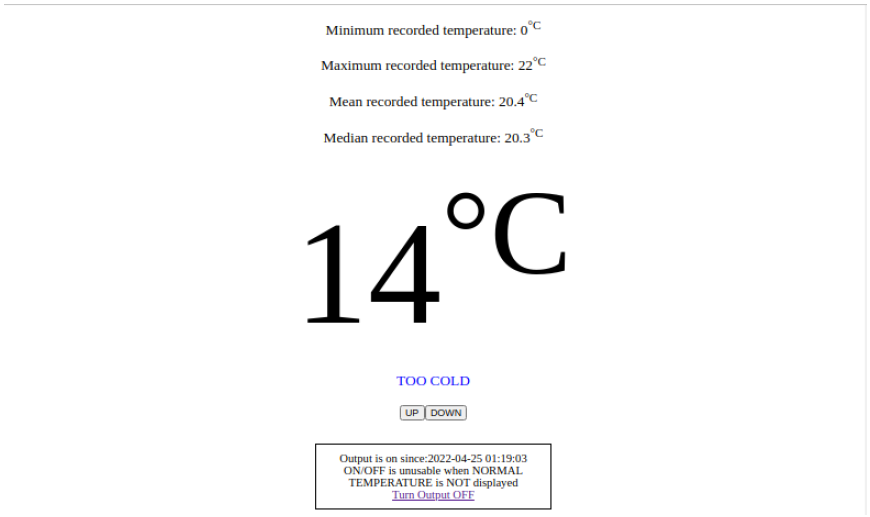
3.4. Database

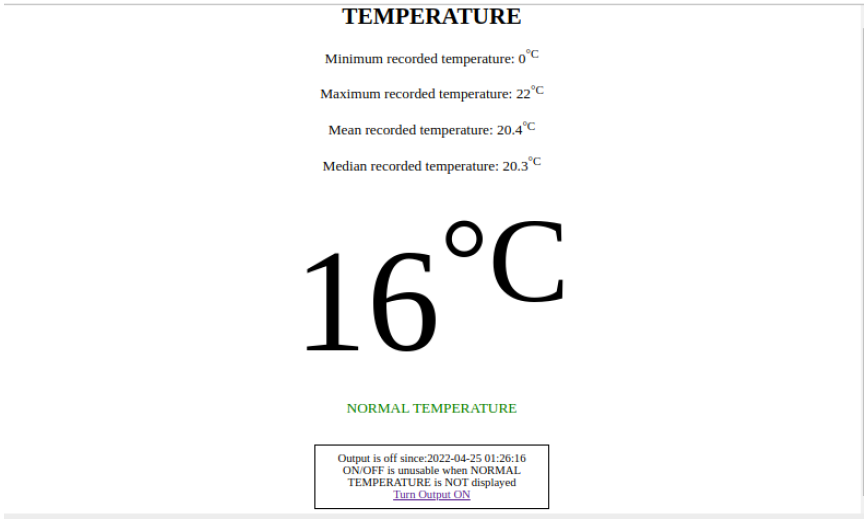
MariaDB is used to store data received from Arduino and within the Raspberry Pi edge server. 3 databases are used. 1 is *temp* to record the temperature. 1 is outputStatus to control ON/OFF of the buzzer. 1 is *buzzlevel* to control volume of buzzer.

3.5. Web interface

PHP that gets data from MariaDB that stores the records of temperature and time recorded. Displayed data includes minimum, maximum, mean and median of temperature recorded. If temperature crosses a threshold (i.e., under 15 and above 20 degrees Celsius), user is notified on the screen and can turn up/down volume of the buzzer. If temperature between 15-20 degrees Celsius, user will have option to manually turn on/off the buzzer.







4. Resources

Ametherm.com. n.d. [online] Available at: <https://www.ametherm.com/blog/thermistors/temperature-sensor-types> [Accessed 18 April 2022].

Arduino Project Hub. 2020. *Arduino Tutorial: Using Potentiometer Control LED Light*. [online] Available at: <https://create.arduino.cc/projecthub/wieselly/arduino-tutorial-using-potentiometer-control-led-light-0dbbd1?ref=part&ref_id=11332&offset=25> [Accessed 20 April 2022].

Arduino Project Hub. 2020. *How to connect DHT11 Sensor with Arduino UNO*. [online] Available at: <https://create.arduino.cc/projecthub/pibots555/how-to-connect-dht11-sensor-with-arduino-uno-f4d239> [Accessed 16 April 2022].

Barnes, R., 2018. *Program an Arduino UNO with your Raspberry Pi — The MagPi magazine*. [online] The MagPi magazine. Available at: <https://magpi.raspberrypi.com/articles/program-arduino-uno-raspberry-pi> [Accessed 15 April 2022].

Jumaa, N., Hameed, O. and Abbas, R., 2018. [online] Research Gate. Available at: <https://www.researchgate.net/publication/329894874_A_Theoretical_Background_of_IoT_Platforms_based_on_FPGAs> [Accessed 20 April 2022].

5. Appendix

5.1. Arduino sketch

Configure pins, potentiometer, LED, buzzer and temperature.

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 2   //connect DHT 11 to pin 2
#define OUTPIN 5    // 5 is a PWM pin for Brightness control
#define POT  A0 // connect the potentiometer to A0
#define BUZZER  6 //Connect Buzzzer to pin 6


//uncomment if you don't need any debug messages

//#define ENABLE_DEBUG

#define DHTTYPE    DHT11     // DHT 11

//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)
String command;
int pot_read=0;
int led_brightness=0;
bool buzzer_state=false;
int buzzer_level=100; //Default cound level range 0 to 225
float temp=0;
```

analogWrite uses the potentiometer as an analog sensor to control brightness of the LED light.

```
DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;

void setup() {
  Serial.begin(9600);

  pinMode(OUTPIN,OUTPUT);
  pinMode(BUZZER,OUTPUT);
  pinMode(POT,INPUT);

  // Initialize device.
  dht.begin();
    sensor_t sensor;
     dht.humidity().getSensor(&sensor);
     dht.temperature().getSensor(&sensor);
  delayMS = sensor.min_delay / 1000;
  debugprintln("Serial Monitor Started");
}

void loop() {
  // Delay between measurements.
  //delay(delayMS);
  delay(100);
  sensors_event_t event;
  dht.temperature().getEvent(&event);


   if (!isnan(event.temperature)) {
    temp=event.temperature;
  }
  Serial.println(temp);

  pot_read=analogRead(POT);
    led_brightness=map(pot_read,0,1024,0,225);
    debugprintln(led_brightness);
  analogWrite(OUTPIN,led_brightness);
```

Control buzzer. If ON then turn on, OFF turns off, UP turns up the volume, OFF turns down the volume.

```
  if(buzzer_state)
  {

analogWrite(BUZZER,constrain(buzzer_level,0,225));

  }
```

```
   if(Serial.available())
  {
    command=Serial.readStringUntil('\n');
    command.trim();
    if(command.equals("ON")){
    buzzer_state=true;
//    debugprintln("LED ON");
 // Serial.println("LED ON");

    }
    else if(command.equals("OFF")){

  digitalWrite(BUZZER,LOW);
   buzzer_state=false;
   //debugprintln("LED OFF");

   }
     else if(command.equals("UP")){

  buzzer_level+=50;

   }
    else if(command.equals("DOWN")){

  buzzer_level-=50;
```

5.2. Python

```
import time
import mysql.connector
import serial

global lastID
global temp
global lastState

if __name__=='__main__':
    ser=serial.Serial('/dev/ttyACM0',9600,timeout=1)
    ser.flush()

//MariaDB connection
mydb=mysql.connector.connect(
    host="localhost",
    user="tempUser",
    password="123456",
    database="tempApp"
    )
mycursor =mydb.cursor()
```

```
//get last ID from buzz
sql=("select id from buzzlevel order by id desc limit 1")
mycursor.execute(sql)
lastID=mycursor.fetchall()
lastID=lastID[0][0]
print(lastID)

//get latest status to determine buzzer on or off
sql=("select status from outputStatus order by update_time desc limit 1")
mycursor.execute(sql)
lastState=mycursor.fetchall()
lastState=lastState[0][0]
print(lastState)
if lastState=="ON":
    ser.write('ON\n'.encode('utf-8'))
    print("SERIAL SEND ON")

else:
    ser.write(b"OFF\r\n")

while True:
    if ser.in_waiting > 0:
         line=ser.readline().decode('utf-8').rstrip()
         temp=float(line)
         sql="insert into temp(temp) values (%s)" %(temp)
         mycursor.execute(sql)
         mydb.commit()

        //get current ID to compare with latest ID
         sql=("select id from buzzlevel order by id desc limit 1")
         mycursor.execute(sql)
         currentID=mycursor.fetchall()
         currentID=currentID[0][0]
         if currentID != lastID:
             //get buzzlevel to determine to turn up or down the volume of buzzer
             sql=("select level from buzzlevel order by id desc limit 1")
             mycursor.execute(sql)
             level=mycursor.fetchall()
             level=level[0][0]
             print(level)
             lastID=currentID
             if level == "UP":
                 ser.write(b"UP\r\n")
                 print("UP LEVEL")
             if level == "DOWN":
                 ser.write(b"DOWN\r\n")
                 print("DOWN LEVEL")

        //get status to determine buzzer should be on or off
         mycursor.execute ("select status from outputStatus order by update_time
desc limit 1")
```

```
        outputStatus=mycursor.fetchall()
        outputStatus=outputStatus[0][0]
        if lastState!= outputStatus:
            lastState = outputStatus
            if outputStatus=="ON":
                ser.write(b"ON\r\n")
            elif outputStatus=="OFF":
                ser.write(b"OFF\r\n")
            else:
                ser.write(b"error\r\n")
                print("error")


    print(temp, "OUTPUT STATUS IS",outputStatus)

    //temperature control: codition
    if temp > 20 and outputStatus == 'OFF':
        sql=("INSERT INTO outputStatus(status) VALUE ('ON')")
        mycursor.execute(sql)
        mydb.commit()
    if temp < 15 and outputStatus == 'OFF':
        sql=("INSERT INTO outputStatus(status) VALUE ('ON')")
        mycursor.execute(sql)
        mydb.commit()
```

5.3. PHP

```
<meta http-equiv="refresh" content="5" url="index.php" >


<h1 style ="text-align:center;">TEMPERATURE </h1>

<?php
$outputStatusUpdate =$_GET["outputStatus"];

//connect to MariaDB
$servername="localhost";
$username="tempUser";
$password="123456";
$db="tempApp";

$conn = new mysqli($servername,$username,$password,$db);
if($conn -> connect_error){
    print"CONNECTION FAILED";
die("Connection Failed ".$conn->connect_error);

}

//Calculate MIN
$sql="SELECT MIN(temp) FROM tempApp.temp";
$result=$conn->query($sql);
```

```
$row=$result->fetch_assoc();
print "<p style='text-align:center; font-size:20px;'>Minimum recorded temperature:
" .$row["MIN(temp)"]."<font><sup>°C</sup></font></p>";

//Calculate MAX
$sql="SELECT MAX(temp) FROM tempApp.temp";
$result=$conn->query($sql);
$row=$result->fetch_assoc();
print "<p style='text-align:center; font-size:20px;'>Maximum recorded temperature:
" .$row["MAX(temp)"]."<font><sup>°C</sup></font></p>";

//Calculate MEAN
$sql="SELECT ROUND(AVG(temp), 1) AS Mean FROM tempApp.temp";
$result=$conn->query($sql);
$row=$result->fetch_assoc();
print "<p style='text-align:center; font-size:20px;'>Mean recorded temperature: "
.$row["Mean"]."<font><sup>°C</sup></font></p>";

//CALCULATE MEDIAN
$sql="SET @rowindex:=1";
$result=$conn->query($sql);
$sql="SELECT ROUND(AVG(t.temp),1) AS Median FROM (SELECT @rowindex := @rowindex + 1
AS rowindex, temp.temp AS temp FROM temp ORDER BY temp.temp) AS t WHERE t.rowindex
IN (FLOOR(@rowindex/2),CEIL(@rowindex/2))";
$result=$conn->query($sql);
$row=$result->fetch_assoc();
print "<p style='text-align:center; font-size:20px;'>Median recorded temperature: "
.$row["Median"]."<font><sup>°C</sup></font></p>";

$sql="SELECT * FROM tempApp.temp ORDER BY update_time DESC limit 1";
$result=$conn->query($sql);

if($result->num_rows >0 ){
while($row=$result->fetch_assoc()){
print "<p style='text-align:center; font-size:200px; margin-top:10px;margin-
bottom:10px;'>".$row["temp"]."<font><sup>°C</sup></font></p>";
print "<p style='text-align:center; font-size:30px; margin-top:10px;margin-
bottom:10px;'>".$row["updatetime"]."</p>";

//IF TOO HOT, SHOWS VOLUME UP/DOWN BUTTONS
    if($row["temp"] > 20){
        print "<div style='text-align:center; color:red;'> ";
        print "<p style='text-align:center; font-size:20px;'>TOO HOT</p>";
        print "<form action='' method='POST'>";
        print "<input type='submit' name='UP' value='UP'></input>";
        print "<input type='submit' name='DOWN' value='DOWN'></input>";
        print "</form>";
        print "</div>";
        if(isset($_POST['UP'])){
            $sql="INSERT INTO buzzlevel (level) VALUES ('UP');";
            $result=$conn->query($sql);
```

```
        }
        else if(isset($_POST['DOWN'])){
            $sql="INSERT INTO buzzlevel (level) VALUES ('DOWN');";
            $result=$conn->query($sql);
        }

    }
//IF TOO COLD, SHOWS VOLUME UP/DOWN BUTTONS
    else if($row["temp"] < 15){
        print "<div style='text-align:center; color:blue;'> ";
        print "<p style='text-align:center; font-size:20px;'>TOO COLD</p>";
        print "<form action='' method='POST'>";
        print "<input type='submit' name='UP' value='UP'></input>";
        print "<input type='submit' name='DOWN' value='DOWN'></input>";
        print "</form>";
        print "</div>";
        if(isset($_POST['UP'])){
            $sql="INSERT INTO buzzlevel (level) VALUES ('UP');";
            $result=$conn->query($sql);
        }
        else if(isset($_POST['DOWN'])){
            $sql="INSERT INTO buzzlevel (level) VALUES ('DOWN');";
            $result=$conn->query($sql);
        }

    }
    else {
//IF NORMAL, VOLUME UP/DOWN BUTTONS ARE HIDDEN
        print "<p style='text-align:center; font-size:20px; color:green;'>NORMAL
TEMPERATURE</p>";
    }
}
}
else{
echo "Temp table is EMPTY";

}


$sql="select *from outputStatus order by update_time desc limit 1";
$result=$conn->query($sql);


print"<br> <div style='border: 1px solid black;width:300px;margin-left:auto;margin-
right:auto;padding:10px;text-align:center; '> ";


if($result->num_rows>0){
while($row=$result->fetch_assoc()){
    $outputStatus=$row["status"];
    $output_timestamp=$row["update_time"];
        if($outputStatus !=$outputStatusUpdate){
```

```
        $sqlOutputStatus="insert into outputStatus(status) values
('$outputStatusUpdate')";
        if($conn->query($sqlOutputStatus)==FALSE){
        echo "GET  ERROR";
}}}}
else
{
echo "OUTPUT STATUS LOG EMPTY<br>";

}

//MANUAL BUTTONS TO ON/OFF BUZZER

//BUTTONS ARE NOT USABLE WHEN TEMPERATURE'S AUTOMATICALLY CONTROLLED (TOO HOT/COLD)

if($outputStatus=="ON"){
print "Output is on since:".$output_timestamp."ON/OFF is not usable when
temperature is not NORMAL TEMPERATURE<br>";
print "<a href='index.php?outputStatus=OFF'>Turn Output OFF</a>";
}
else if($outputStatus=="OFF"){
print "Output is off since:".$output_timestamp."ON/OFF is not usable when
temperature is not NORMAL TEMPERATURE<br>";
print "<a href='index.php?outputStatus=ON'>Turn Output ON</a>";
}
else
{
print "HIT RESET. ON/OFF is not usable when temperature is not NORMAL TEMPERATURE
<br>";
    print "<a href='index.php?outputStatus=OFF'> RESET </a>";

}
print"</div>";
$conn->close();

?>
```

5.4. Database

temp to record temperatures.

```
MariaDB [tempApp]> describe temp;
+-------------+-----------+------+-----+---------------------+----------------+
| Field       | Type      | Null | Key | Default             | Extra          |
+-------------+-----------+------+-----+---------------------+----------------+
| Temp_id     | int(11)   | NO   | PRI | NULL                | auto_increment |
| temp        | float     | YES  |     | NULL                |                |
| update_time | timestamp | NO   |     | current_timestamp() |                |
+-------------+-----------+------+-----+---------------------+----------------+
3 rows in set (0.001 sec)
```

outputStatus to record ON/OFF, which determine to turn on/off the buzzer.

```
MariaDB [tempApp]> describe outputStatus;
+-------------+-----------+------+-----+---------------------+----------------+
| Field       | Type      | Null | Key | Default             | Extra          |
+-------------+-----------+------+-----+---------------------+----------------+
| Status_id   | int(11)   | NO   | PRI | NULL                | auto_increment |
| status      | text      | YES  |     | NULL                |                |
| update_time | timestamp | NO   |     | current_timestamp() |                |
+-------------+-----------+------+-----+---------------------+----------------+
3 rows in set (0.001 sec)
```

buzzlevel(level) to record UP/DOWN of volume and change volume accordingly.

```
MariaDB [tempApp]> describe buzzlevel
    -> ;
+-------+-----------+------+-----+---------------------+----------------+
| Field | Type      | Null | Key | Default             | Extra          |
+-------+-----------+------+-----+---------------------+----------------+
| id    | int(11)   | NO   | PRI | NULL                | auto_increment |
| level | text      | YES  |     | NULL                |                |
| T_S   | timestamp | NO   |     | current_timestamp() |                |
+-------+-----------+------+-----+---------------------+----------------+
3 rows in set (0.001 sec)
```