ECE 442

Internet of Things and Cyber Physical Systems

Design Project

IoT based weather monitoring system using Raspberry PI

Created by:

Anand Baraguru Venkateshmurthy (A20429060)

Abstract:

Internet of Things gives the opportunity to build powerful smart systems utilizing the improvements in wireless networks, embedded technologies and sensor devices. Several IoT applications that have been developed, implemented and deployed in recent years. In the world today, automation has taken over manual system. With the fast increase in the number of users of internet in the past decade IoT is the latest and emerging internet technology. Internet of things is a growing network of everyday object-from most machine applications to consumer goods that can exchange information and complete tasks in a smart way. This design proposes that the weather monitoring by using temperature sensor, humidity sensor value to read and monitor using webserver Raspberry pi.

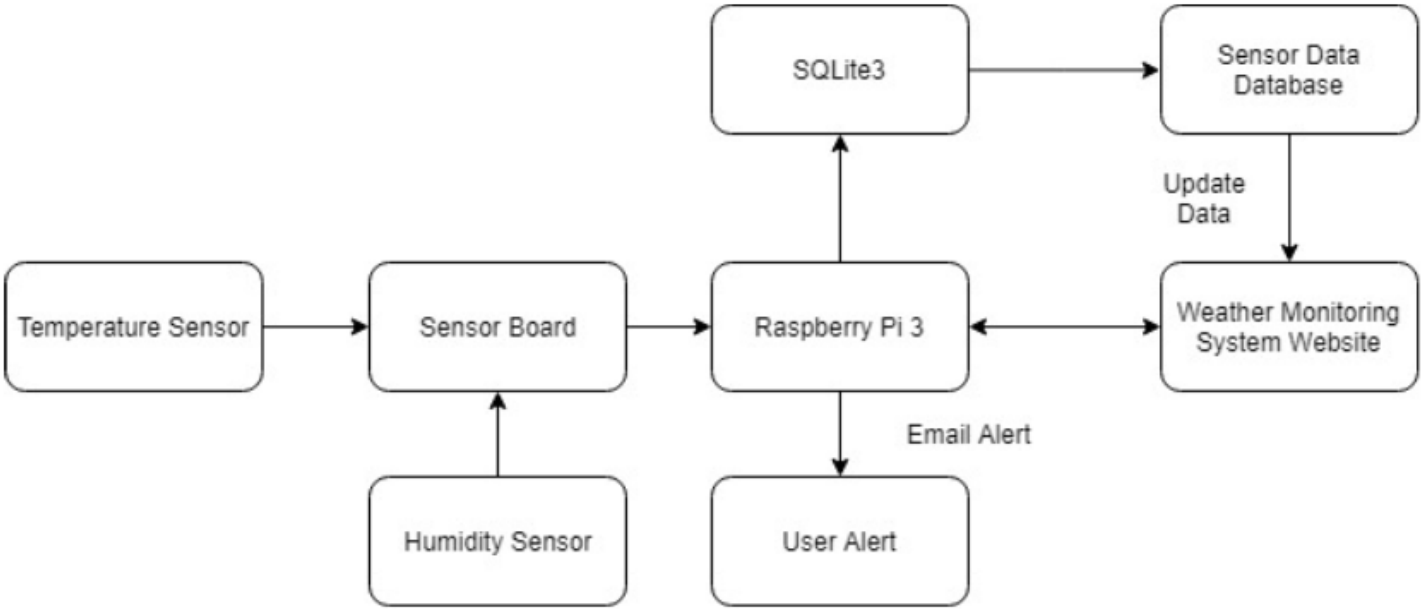
Introduction:

Many things that affect the weather conditions of a place it can have an adverse effect on most of the environment. At a weather study of different parameters using instruments and equipment's has been done. Sensors are necessary components that can be used for measuring traffic flow, weather parameters such as humidity, temperature, pressure etc. which are used for predicting the environmental conditions. So, to meet the goal of weather monitoring we have designed IoT based real-time, low-cost, portable and high-speed weather station using Raspberry Pi.

In our project a weather monitoring system website was developed was with Raspberry Pi webserver. A weather monitoring system based on embedded and web technology was implemented. The system is designed with the use of various hardware tools like raspberry pi, DHT22 sensor and software tools like python with flask framework. The data was stored into a sqlite3 database which contained the information on various parameters like temperature, humidity etc.

Project Description:

In this project we predict the conditions in a region based on the values of weather parameters such as humidity, temperature. The raspberry pi sensor modules that can be installed in different parts of the world and the weather conditions of that place can be accessed through the server where the data from the sensors is transmitted and stored. This system monitors temperature, humidity. The flowchart describes our system that we modeled.



Here we have used a raspberry pi as a webserver to host a website that displays the temperature and humidity data and also send email alerts to the user when there is an abnormal condition of the environment.

Hardware Components:

We use the following hardware components to implement the project:

Raspberry Pi:

Raspberry Pi is a small sized computer developed by the Raspberry Pi Foundation in the UK to make the basics of computer technology be available in schools of all developing countries. The original model name of Raspberry Pi came out to be very popular than expected and sold across various target markets like robotics and many other applications. We can attach a mouse, keyboard, monitor etc and most of the accessories that can be connected to a computer.

Raspberry Pi has a pre-installed operating system based on linux called Raspbian OS.

The Raspberry Pi Model that we are using has the following specs.

Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit system on chip clocked @ 1.4GHz
1GB LPDDR2 SDRAM
2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE.
Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
Extended 40-pin GPIO Headers
4 USB 2.0 ports
A Micro SD port is provided for loading the operating system and storing data
5V/2.5A input DC power
Power-over-Ethernet (PoE) support

Temperature & Humidity Sensor:

We are using a large plastic body temperature and humidity sensor DHT22(AM2302). The sensor is economical and uses a capacitive humidity sensor and a thermistor to measure the surrounding air. The sensor gives a digital signal on the data pin. The sensor provides a reading for 0-100% humidity with an accuracy of 2-5% and temperature between -40 to 80°C with an accuracy of ±0.5°C. The sampling rate of the sensor is 0.5Hz which is about 2 second for each sample.

DHT22 sensor was interfaced with the Raspberry Pi 3 by connecting the pin1 of the raspberry Pi which is the voltage pin to DHT22 sensors VCC pin which is pin1, the Raspberry Pi Ground pin which is pin 6 is connected to DHT22 Ground pin 4 and Raspberry Pi GPCLK0 pin 7 to is connected to Data pin which is pin 2 on the DHT22. Below is the specifications of the sensor

Software components used:

We use the following software components for our project

Raspbian OS:

Raspbian operating system is a computer operating system which is Debian-based created for Raspberry Pi. There are several versions of Raspbian operating system including Raspbian Stretch and Jessie. Raspbian was officially provided by the Raspberry Pi foundation since 2015 as the primary operating system for Raspberry Pi. Raspbian operating system is optimized for the Pi based low-performance ARM CPUs.

Python

Python is a very powerful programming language that is very easy to use and user friendly. We can use python to connect your Raspberry Pi to many things in the real world. The Pi comes with its own inbuilt Python IDLE which is an editor for python programs. This makes it very simple to start programming on the Pi. We need to update python it on the raspberry pi so that all the libraries are updated. We used python for our programming as it was already built on the raspberry pi. Python has many advantages over other languages such as a program in python is 5-10 times smaller than its equivalent program in java or cpp.

CSS

Cascading Style Sheets (CSS) is a styling language for the artistic representation of a webpage that is written in HTML. It describes how everything should be processed to be displayed on screen, on paper, in speech and on other media.

CSS is one of many important languages for the open web development and is a standard among most web browsers. CSS specifies the color graphics, texture and outline of the web pages.

HTML

It is abbreviated as Hyper Text Mark-up Language. It is a very widely used markup language for creating HTML webpages and web applications along with the usage of CSS and java script. It describes the webpage structure using markup. HTML elements build up the HTML webpages that eventually build up a website.

The web browser gets the HTML pages from a webserver or a local storage and process it so that it is displayed onto a website on a browser.

SQLite

SQLite is a relational DBMS tool that is in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.

Flask

Flask is classified micro web-framework that is used for the development of websites. Flask does not need specific tools or libraries. It is microframework that is built using python. Flask supports the use of extensions that will add features to web applications. These extensions support mapping of relational objects, validation of forms, file handling, user authentication and many common framework-based tools.

VNC Viewer

VNC viewer is a desktop sharing system that uses the RFB protocol to another computer remotely. We used the VNC viewer app to connect to the raspberry pi remotely.

The main challenge for this system is the accuracy of the sensor which is about 2-5% and the sensor reading is more than that of the real time data. If the reading of the sensor is late then it takes some time for the data to display on the website. This can be avoided by properly interfacing the sensor with the raspberry pi.

An important security issue for the pi is that the root/admin access ie the sudo access doesn't need a password for applying changes to the system. So, if you are into the on the network that the pi is connected to any unauthorized personnel can easily access the pi to do any modifications for the pi. So, if the pi userid and password are compromised, then the whole system and every system that is connected to its network LAN will also be tampered. So, if we configure the sudo access to require a username and password this problem can be avoided.

One other way attackers can get into the system is by SSH. If the raspberry pi is not configured for your personal credentials, it's not that difficult and long for an attacker to get onto the raspberry pi and tamper with it using the default credentials.

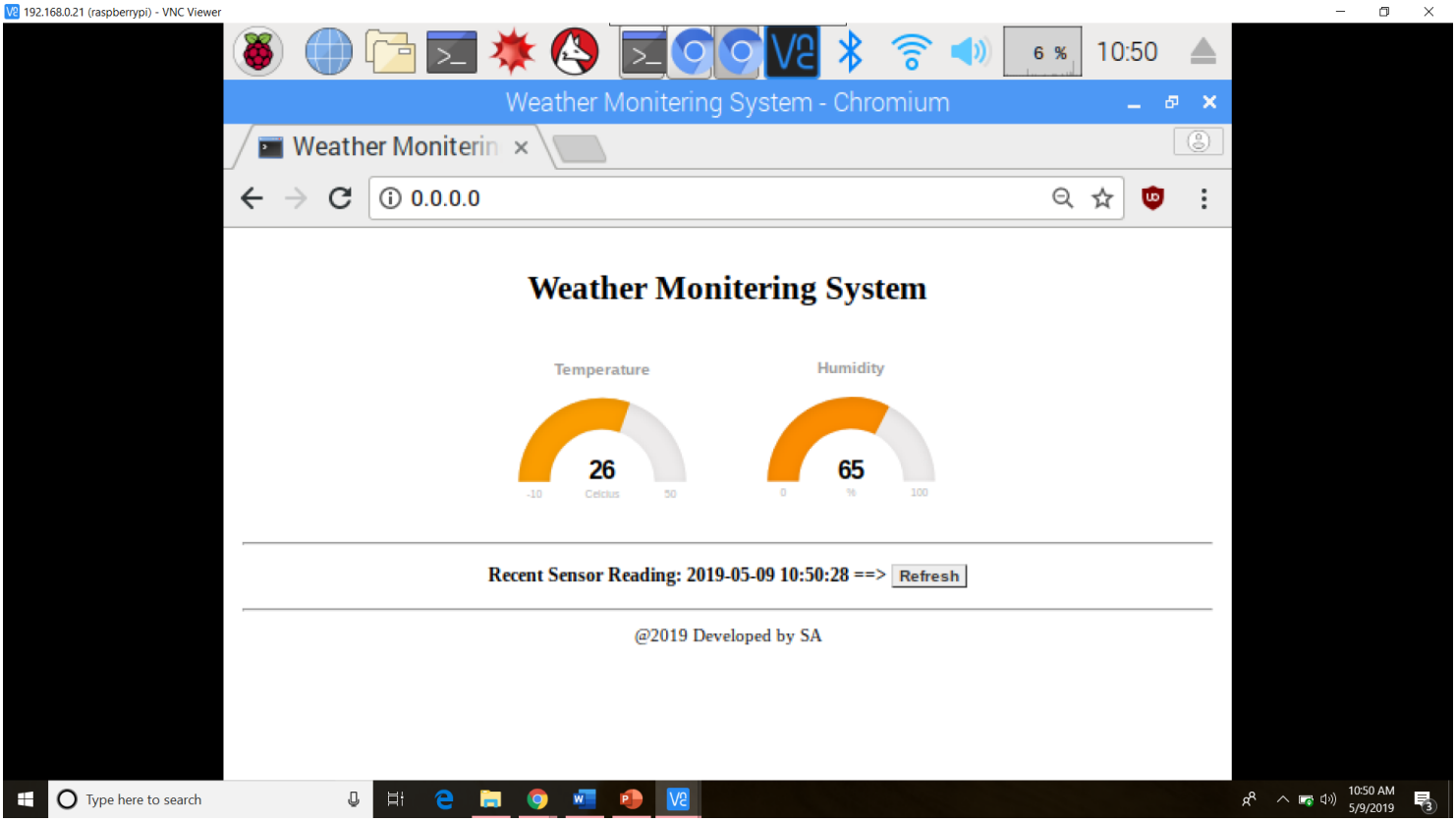
The entire code was written by taking references from many projects. We have also used some gauges which available as open source codes for the users are to represent the temperature and humidity in a pleasing way.

There are many projects that are similar to ours but not exactly like ours. In our project we send an email alert to the user if any abnormal rise in conditions.

One such similar project is IoT weather station using various sensors like temperature, humidity, piezoelectric etc. Here, the sensor data is being transferred to a remote webserver from the website called as things speak where the data is not stored in the data base but is portrayed on a graph on the thingspeak website. The data retrieved from the sensor is displayed and destroyed instantly. In our project we store the data form the server on the SQLite3 database for future study. Their project doesn't give an email alert but, we have included an email alert system to give an alert to the user.

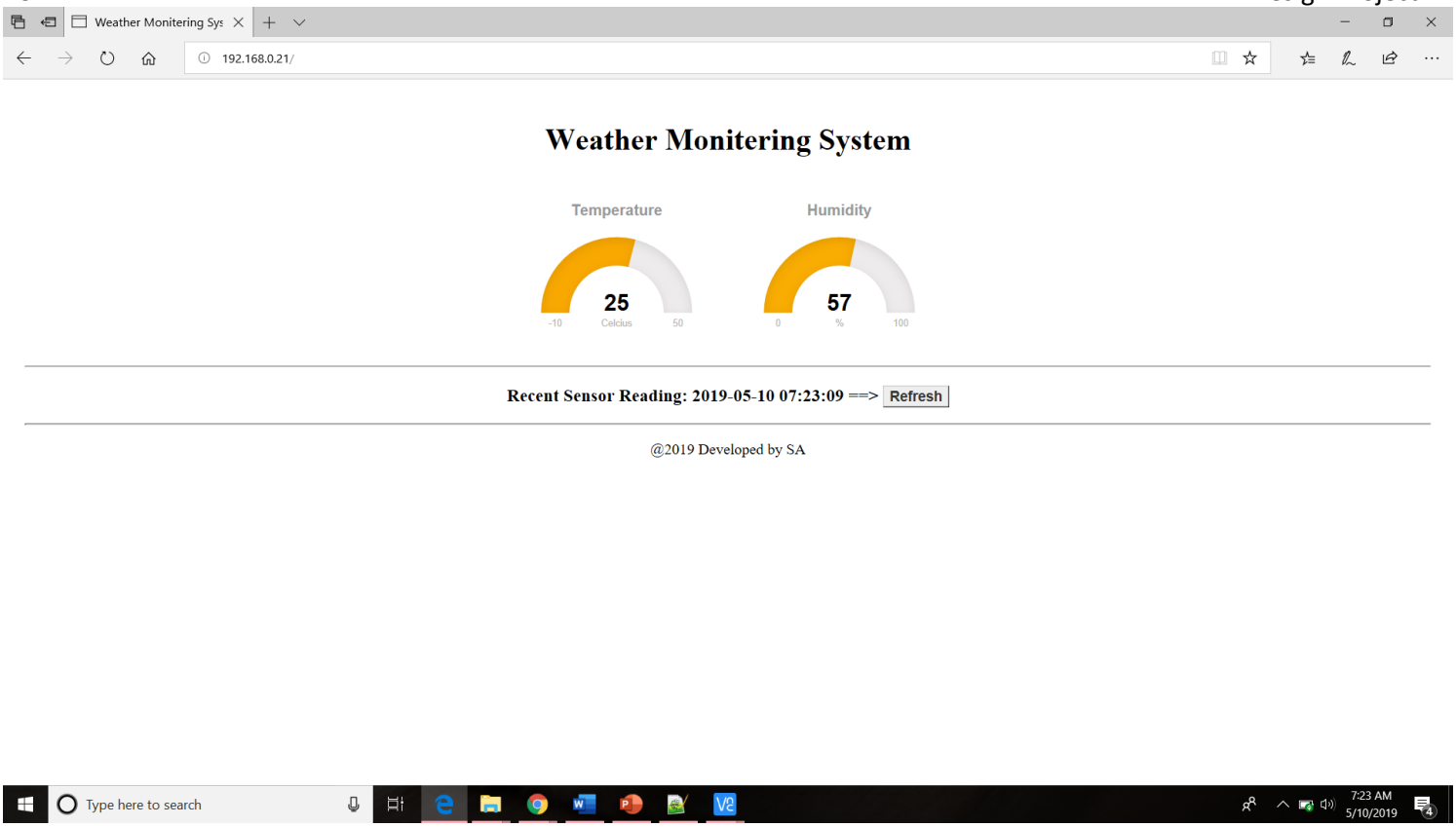
Results:

This is our website that we have developed for the weather monitoring system which consists of two gauges that depicts the values of temperature and humidity. When the refresh button is hit data is collected from the sensor and it is not only displayed on the website but also saved in the local database storage.



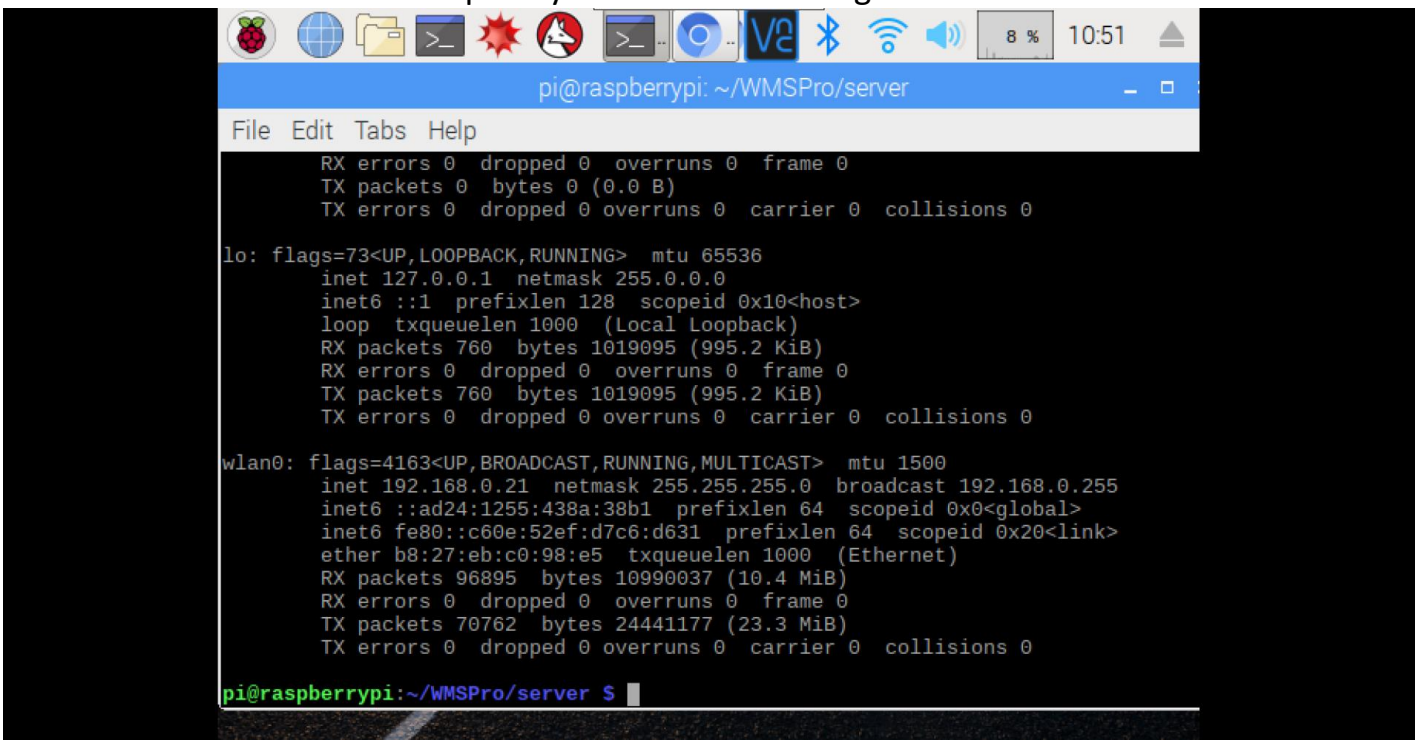
Weather Monitoring System Website hosted on the raspberry pi

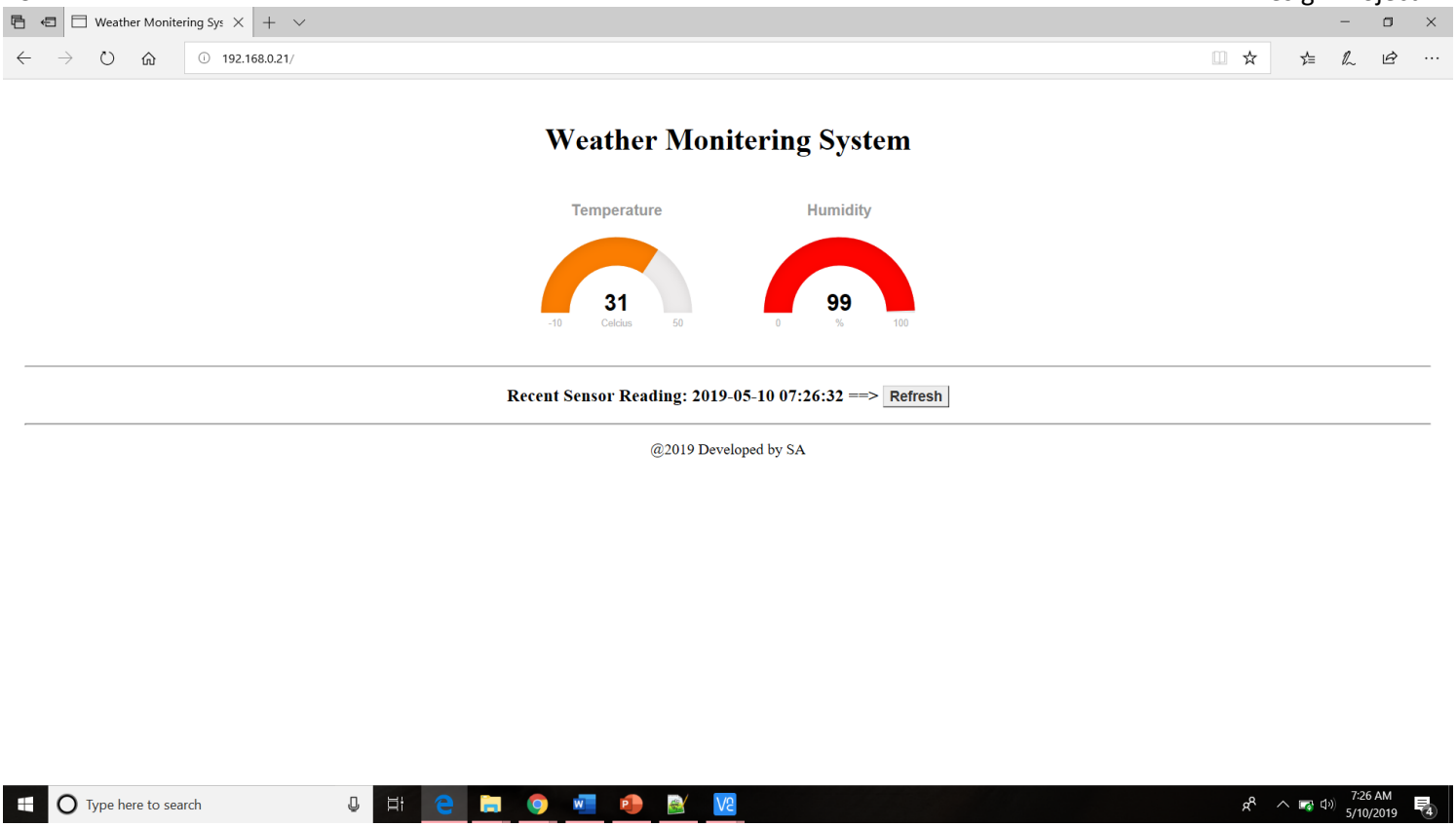
When we connect look up the IP address in the browser of our local machine that is connected to the same network as our raspberry pi we should be able to get our website. This tells us that the raspberry pi is successfully used as a webserver where we can retrieve the information from.



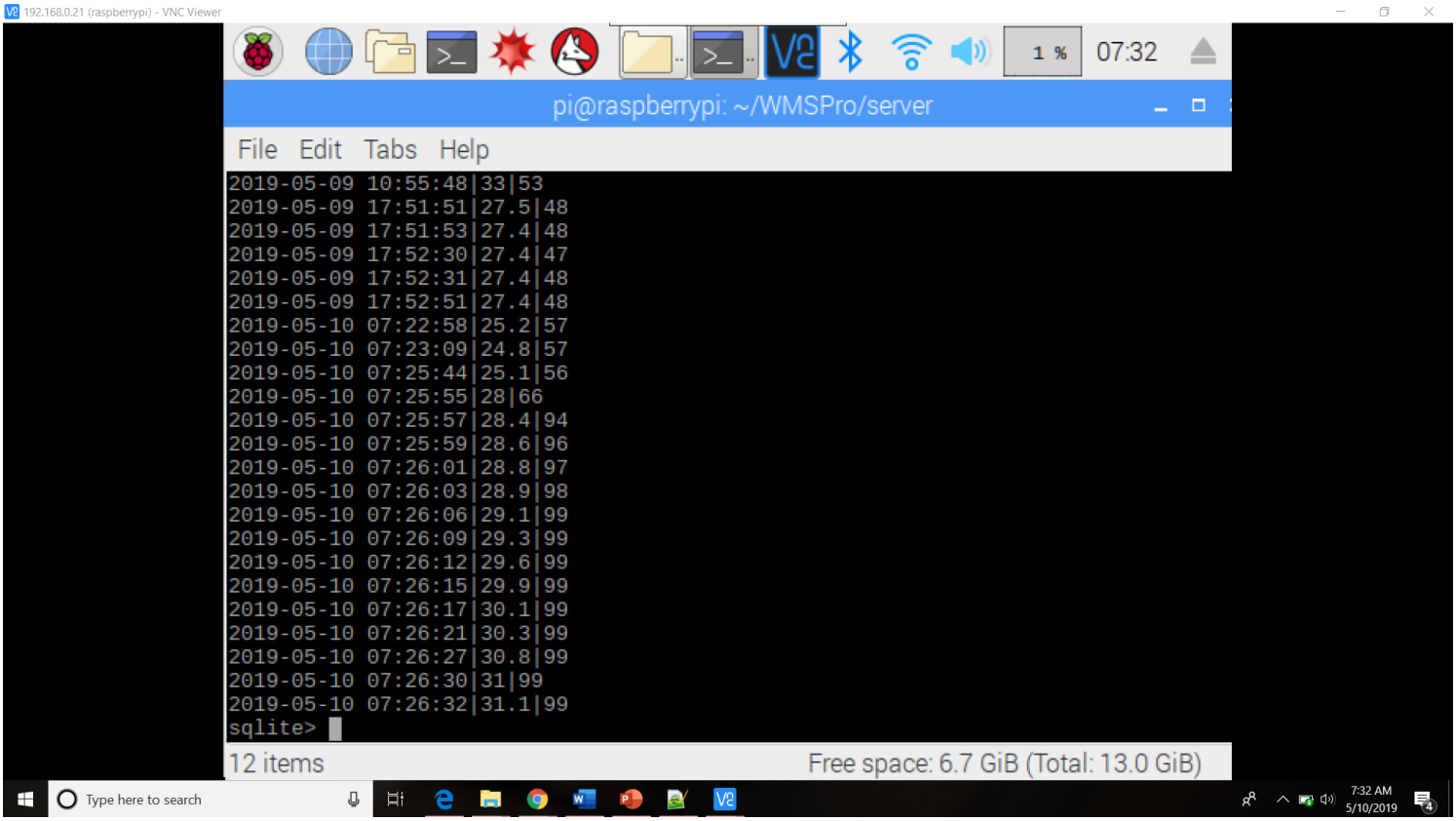
Our website hosted on our local machine

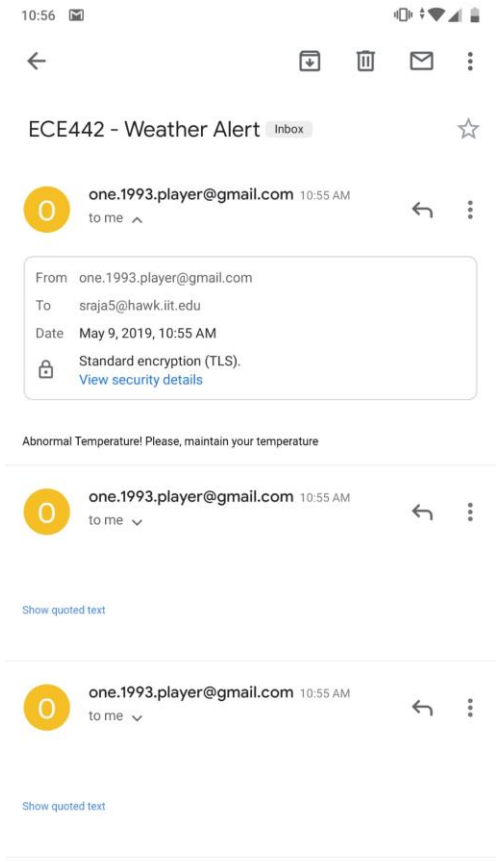
Below is the IP address of the raspberry Pi that we are using:





Above when the temperature reaches more than 30 degrees there is an email alert that is sent to the user. The below is the data stored in the SQLite3 database.





Email Alert

We initially thought that we would set up our website using a LAMP server but later since we found a good gauging tool which was a better way of representing data. It was built using JavaScript so we decided we will build our own website using python flask webserver, gauges using JavaScript and using SQLite3 as the database for our system. python as we know it is a very commonly used language and there are endless opportunities and along with flask framework it is extendable through various extensions provided by the flask framework. Flask was the best option for us along with python as flask is a lightweight micro web-framework and is suitable for our low processing powered assessment.

Conclusion:

Prediction of the Weather is an important factor, which forecasts the climate in a region based upon the values of weather parameters. The recorded data from this system can be used in forecasting the weather conditions of that area for a period. We used the Raspberry pi in this model, sent an alert message via e-mail to the user, when the parameters changes reach a certain level. Technology as we know it changes day by day. Using the sensor for temperature and humidity in combination with Raspberry PI we developed a system. Data from the sensors is transmitted to sever where it is stored, and can it can be accessed globally making it easily accessible to everyone. Our IoT based system gives real-time monitoring of environmental parameters like temperature and humidity. The client using the system can monitor each minute different parameters of the environment without the interaction with an additional server.

References:

1] https://www.instructables.com/id/Raspberry-Pi-Tutorial-How-to-Use-the-DHT-22/

2] https://www.instructables.com/id/Python-WebServer-With-Flask-and-Raspberry-Pi/

3] https://myhydropi.com/send-email-with-a-raspberry-pi-and-python

4] https://www.codewall.co.uk/creating-gauge-charts-with-the-justgage-javascript-library/

5] https://www.sqlite.org/lang.html

Future Work:

This can be further enhanced using various other sensors and monitor the system remotely and the data from the sensors can be accessed globally. We can further extend this application into an android app so that each and every android user can monitor the data specific to their area. This application is best suitable for farmers as environmental conditions are crucial for the cultivation of crops. So, by learning the changes in environmental conditions over the past time farmers can decide the type of crops to grow for that environmental condition. The modeling of such a system is very cheap and easily accessible.

Source Code:

HTML Code:

```
<!doctype html>
<html>
<head>
   <title>Weather Monitering System</title>
   <link rel="stylesheet" href='../static/style.css'/>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <style>
        body {
           text-align: center;
        }
        #g1,
```

```
        #g2 {
            width: 200px;
            height: 160px;
            display: inline-block;
            margin: 1em;
        }
    </style>
</head>
<body>
    <h1>Weather Monitering System </h1>
    <div id="g1"></div>
    <div id="g2"></div>
    <hr>
        <h3> Recent Sensor Reading: {{ time }} ==> <a
href="/"class="button">Refresh</a></h3>
        <hr>
        <p> @2019 Developed by SA</p>
    <script src="../static/raphael-2.1.4.min.js"></script>
    <script src="../static/justgage.js"></script>
    <script>
        var g1, g2;
//The temperature gauge
document.addEventListener("DOMContentLoaded", function(event) {
            g1 = new JustGage({
                id: "g1",
                value: {{temp}},
                valueFontColor: "black",
                min: -10,
                max: 50,
                title: "Temperature",
                label: "Celcius"
        });
//The humidity gauge
            g2 = new JustGage({
                id: "g2",
                value: {{hum}},
                valueFontColor: "black",
                min: 0,
                max: 100,
                title: "Humidity",
                label: "%"
        });
```

```
        });
      </script>
    </body>
    </html>
```

CSS File:

```
// determining the style of the body
    body{
        background: white;
        color: black;
        padding:1%
    }
// determining the style for the refresh button
    .button {
        font: bold 15px ComicSans;
        text-decoration: none;
        background-color: #EEEEEE;
        color: #333333;
        padding: 2px 6px 2px 6px;
        border-top: 1px solid #CCCCCC;
        border-right: 1px solid #333333;
        border-bottom: 1px solid #333333;
        border-left: 1px solid #CCCCCC;}
```

Python Code:

```
from flask import Flask, render_template, request
app = Flask(__name__)
# Importing the libraries
import time
# Importing the Sqlite3 library
import sqlite3
# Importing the DHT22 sensor library
import Adafruit_DHT
# Importing the email alert library
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
# Getting the source email address
source="one.1993.player@gmail.com"
# Getting the destination email address
destination="sraja5@hawk.iit.edu"
# Defining the class for message alert
```

```
def moniter():
        msg = MIMEMultipart()
        # Defining the source email
        msg['From'] = source
        # Defining the destination email
        msg['To'] = destination
        # Defining the subject
        msg['Subject'] = "ECE442 - Weather Alert"
        # Defining the body of the subject
        body = 'Abnormal Temperature! Please, maintain your temperature'
        # Defining the style of the body
        msg.attach(MIMEText(body, 'plain'))
        # defining the server mail id and port
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.ehlo
        server.starttls()
        server.login(source, "ece442project")
        text = msg.as_string()
        # sending the email
        server.sendmail(source, destination, text)
        # Closing the connection to the email server
        server.quit()

        # Defining the name of the database
        dbname='sensorsData.db'
# Getting the source email address
def getsensordata():
        DHT22Sensor = Adafruit_DHT.DHT22
        # Defining the sensor data pin
        sensorpin = 16
        # getting the data from the sensor
        hum, temp = Adafruit_DHT.read_retry(DHT22Sensor, sensorpin)
        # check if there is no value from the sensor
        if hum is not None and temp is not None:
                hum = round(hum)
                temp = round(temp, 1)
                logData (temp, hum)
# saving sensor data on to the database
def logData (temp, hum):
        # initialize the connection to the database
        conn=sqlite3.connect(dbname)
        # open the connection to the database
```

```
        curs=conn.cursor()
        # Execute the command for insertion of data to the database
        curs.execute("INSERT INTO DHTSensordata values(datetime('now','localtime'), (?), (?))",
(temp, hum))
        conn.commit()
        conn.close()
# fetching the data from database
def getData():
        # initialize the connection to the database
        conn=sqlite3.connect('/home/pi/WMSPro/server/sensorsData.db')
        # open the connection to the database
        curs=conn.cursor()
        # Execute the command for retrieval of data from the database
        for row in curs.execute("SELECT * FROM DHTSensordata ORDER BY temp desc LIMIT 1"):
                time = str(row[0])
                temp = row[1]
        # Check if the temperature if more than 30 then send a mail alert
                if temp > 30:
                        moniter()
                hum = row[2]
        conn.close()
        #send the time, temperature, humidity data to the caller
        return time, temp, hum

# main route of the website and rendering
@app.route("/")
def index():
        # Update the database to the latest value
        getsensordata()
        # retrieve data from the database
        time, temp, hum = getData()
        templateData = {
                'time': time,
                'temp': temp,
                'hum': hum
        }
        # return the data to the caller
        return render_template('index.html', **templateData)
if __name__ == "__main__":
#hostin the website through port 80
  app.run(debug=True, host='0.0.0.0', port=80)
from flask import Flask
```