

Weather detection using IoT

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



Submitted to:

Er. Charanpreet Kaur

Submitted by:

Mishra Nitin Sherbahadur | Dhruv Bhattacharjee

18BCS1988 | 18BCS2072

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Chandigarh University, Gharuan

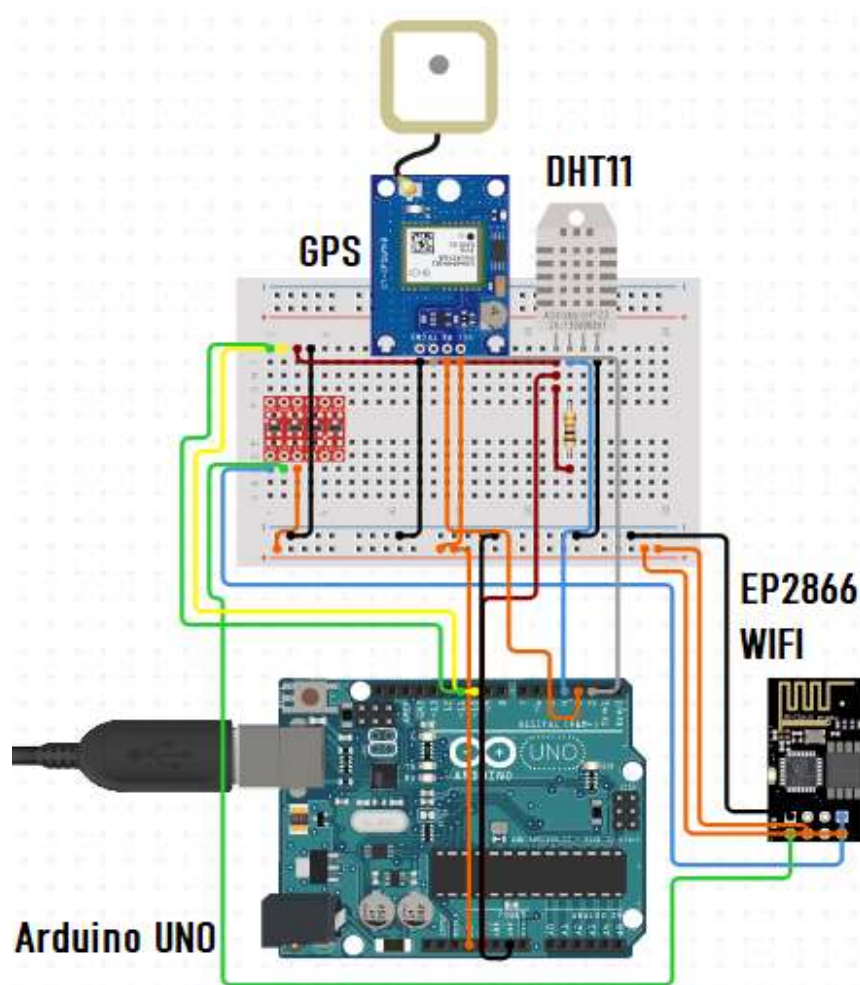
March 2021

Project Design

Aim:

The main aim of our project is to create a simple tool to monitor, collect, analyze and determine weather conditions. This tool makes monitoring and analyzing real-time data of temperature, humidity, and rain with GPS coordinates of place an ease. The data will be stored in a cloud-based database so the access to data is instant from anywhere. The outstanding advantage of this tool is it can be modified according to user needs by addition and removal of modules on Uno board.

Hardware:



This project is built using IoT based tools. The tools used are Arduino Uno board, DHT11 Temperature and humidity sensor, Raindrop detection sensor, EP2866 WIFI module, GPS Module, EP2866 NodeMCU module. The Arduino Uno board is connected to temperature and humidity sensor, raindrop sensor and WIFI module. This configuration collects temperature, humidity, rain, GPS data and sends it over to cloud-based database through internet which is connected via WIFI network. This data is stored in MongoDB database and will be used to predict nearby future weather conditions. This data can also be monitored in real-time by user by user from a mobile app or website.

Software:

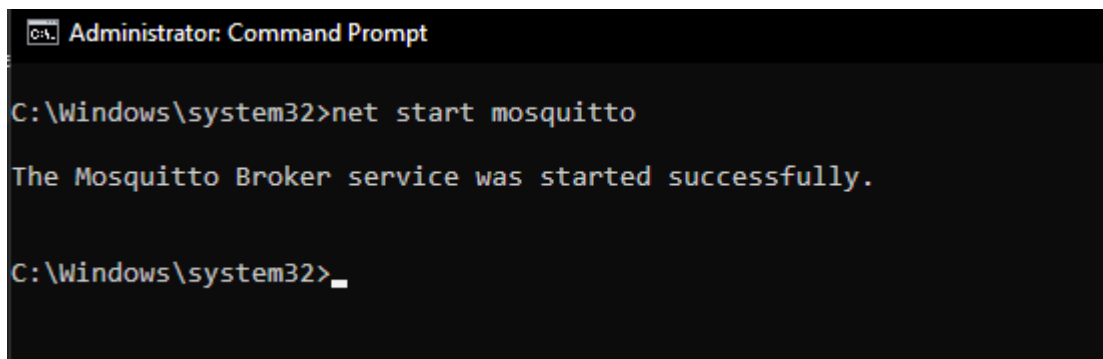
This project is developed in Arduino IDE and the collected data is stored in MongoDB database. MQTT protocol is used over here to transfer data between Arduino UNO and the server-side MQTT broker. The MQTT broker used over here is Mosquitto. Mosquitto receives the data and transfer it to MongoDB database. The data stored in database can be monitored in real-time and will be passed to machine learning to determine possible weather conditions in future.

Database (MongoDB)

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).

MongoDB stores our temperature, humidity and other weather statistics in NoSQL format. The data stored in this database will be accessed in real-time to monitor weather conditions and will be also used by machine learning algorithms.

MQTT Broker (Eclipse Mosquitto).



```
Administrator: Command Prompt

C:\Windows\system32>net start mosquitto

The Mosquitto Broker service was started successfully.

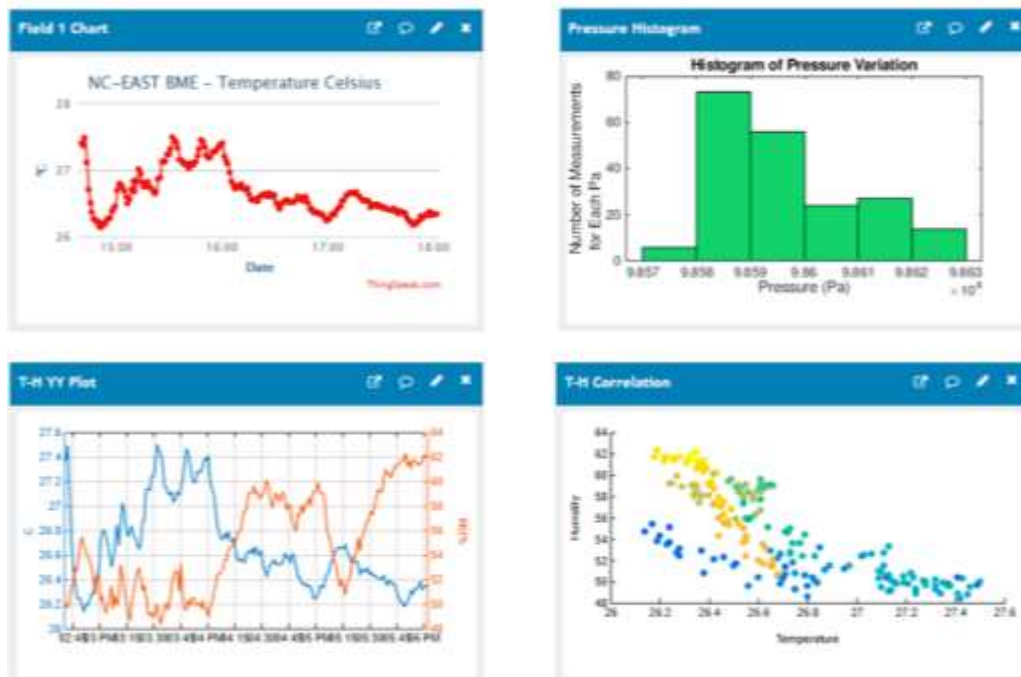
C:\Windows\system32>
```

MQTT (Message Queuing Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC 20922) lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

Monitoring

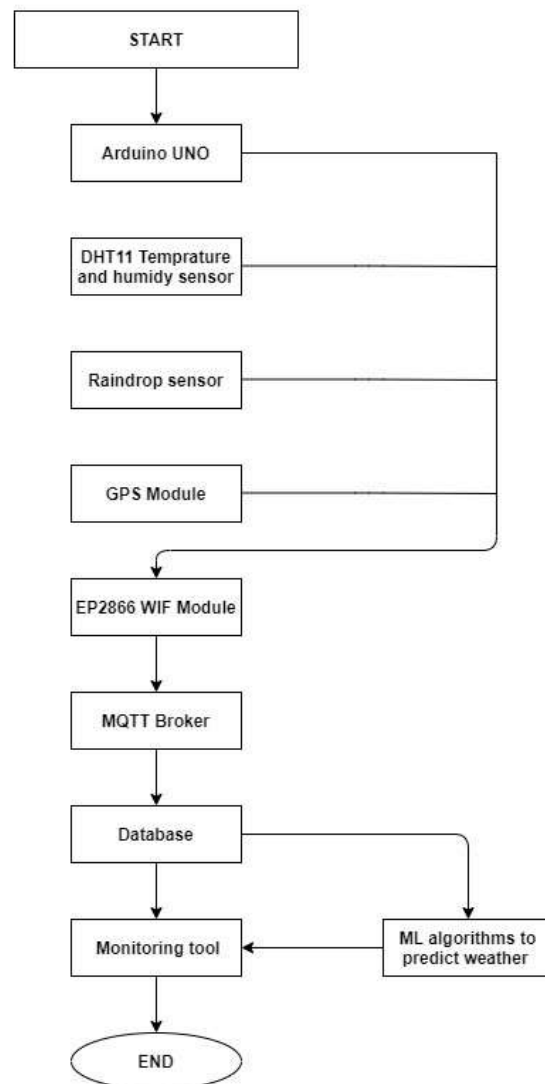


The data collected by sensors such as temperature, humidity, GPS location can be monitored over a light-weight webpage that will be served by Arduino board itself over local network. The data collected over time can also be monitored. Data is represented in form of numbers as well as in graph and tabular form. The monitoring utility also provide with the option of forecasting weather conditions using machine learning algorithms.

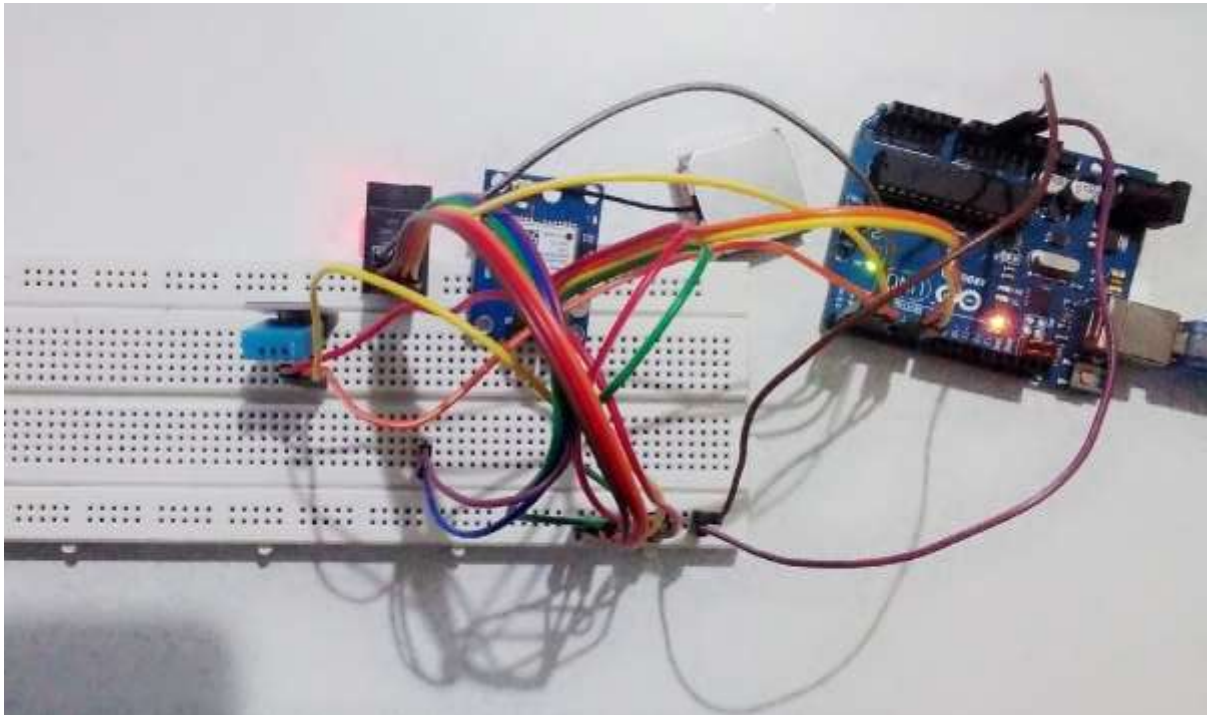
Innovations in model/design/solution.

The tool is based on IOT based devices so it is small in size and also can be modified to user needs. The device can send data over a WIFI network or by using a GSM module for internet connectivity from anywhere. The data can be seen in real-time as well as is collected in data-base so weather statistics of any time over which the tool was installed and enable can be accessed. The programming used in this tool I completely open-source so can be modified according to user need with ease and user has complete access over how the data is handled. The data collected is uses to determine future weather statistics. Advantage of using this tool is that it serves the tasks of weather monitoring, weather data collection and weather prediction in one package. The collected data of weather is available to be viewed in a analyzed manner in a graphical or tabular form. The monitoring utility also provides function to compare weather statistics.

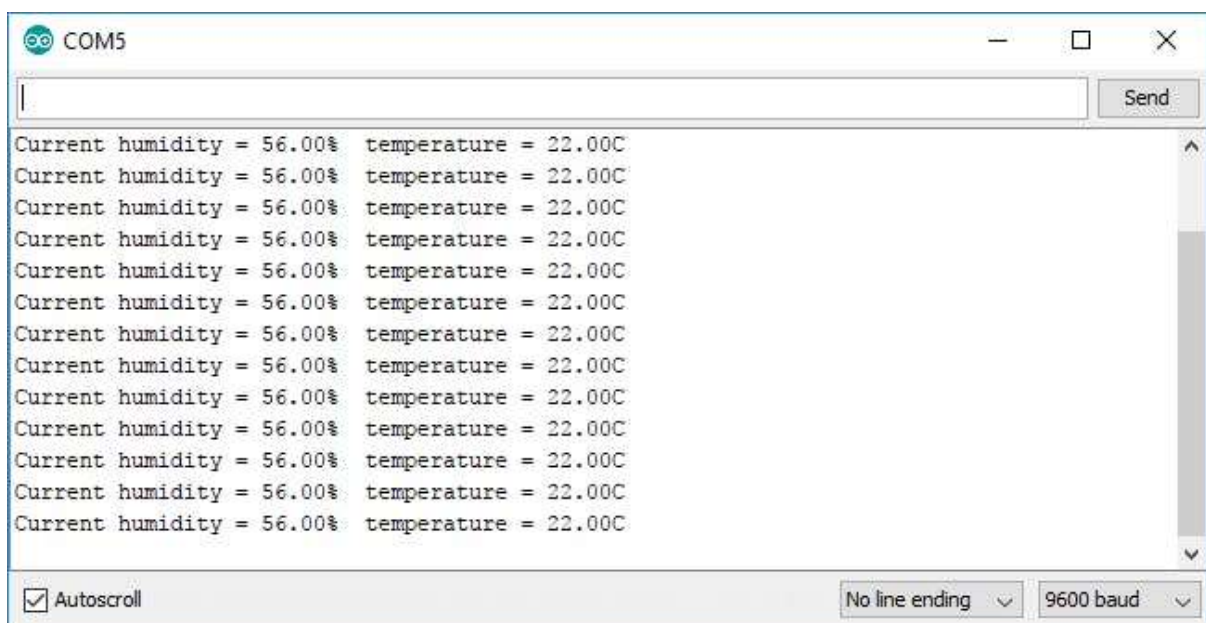
Flow chart:



Implementation



The hardware part of the project is ready. The Arduino Uno board is connected with all sensors. Sensors collect data and sends it over WIFI to the broker and currently the received data is displayed Arduino serial monitor.



The data is also stored in MongoDB database. Database is currently stored on local computer but as the project develops ahead the database will be moved on Google cloud services.

The following code stores data in MongoDB database:

```
exports = function(data){

  //Get the current time
  var now = new Date();

  var darksky = context.services.get("darksky");
  var mongodb = context.services.get("mongodb-atlas");
  var TempData = mongodb.db("Imp").collection("TempData");

  // Fetch the current weather from darksky.net

  darksky.get({"url": "https://api.darksky.net/forecast/" +
    context.values.get("DarkSkyKey") + '/' +
    context.values.get("DeviceLocation") +
    "?exclude=minutely,hourly,daily,alerts,flags&units=auto"
  }).then(response => {
    var darkskyJSON = EJSON.parse(response.body.text()).currently;

    var status =
      {
        "Timestamp": now.getTime(),
        "Date": now,
        "Readings": data,
        "External": darkskyJSON,
      };
    status.Readings.light = (100*(data.light/65536));
    context.functions.execute("controlHumidity", data.temp, data.humid);
    TempData.insertOne(status).then(
      results => {
        console.log("Successfully wrote document to TempData");
      },
      error => {
        console.log("Error writing to TempData collection: " + error);
      });
  });
};
```