

# **Flood Forecasting and early warning system Using based on IoT**

## **Introduction:**

In a peninsular country like India, with extreme weather and climatic conditions, the occurrence of heavy rainfall is normal. Multiple times, the arrival of very heavy rains results in the heavy discharge of water or because of the sudden melting of the glaciers due to global warming. Especially, in the monsoon which normally begins in the mid of June and lasts till October, thousands of people lost their lives by drowning and their habitats were collapsed. The left over were evacuated by the state and central disaster relief authorities. The severe water logging brought daily work to halt. In order to save the lives of the people, their habitat and the economy, the major step is to monitor the data on real time basis and if the situation is reaching a certain threshold, then alert is to be provided to each individual living in the area which is currently at risk. Even if it is difficult to abandon the natural calamity but the mandatory steps are to be taken by the government agencies to shift the population to a safe region and the losses will get reduced to less than 30%.

In this modern era, there are multiple systems working and are deployed at different locations but the alert notification is passed to government agencies and this ends up in slowing down the process. The reason behind this is that flood is very spontaneous disaster and government agencies have to follow multiple steps before reaching to a decision. In this case, awareness among the people is very necessary along with the government officials. So that a combined and better result will be achieved. In our system, it is combined with prediction through weather forecasting. The flow of water is sensed by water flow sensor which will ultimately help in evaluating the intensity of flood and water level by the help of ultrasonic sensor which will be done by propagating sound waves. All these information will be showcased in the application.

The android application will be developed using Java, XML, Android studio. The application not only includes the data from IOT but also includes weather forecasting. The weather information will be gathered using Open Weather API which will provide real time climatic information. This will result in the avoidance of loss to mankind and economy as well.

## **Purpose:**

The main purpose of the initiative was to develop a location specific early warning system which could help the administration in taking advance precautionary measures and issue flood alerts to those specific areas so that necessary measures can be undertaken by the people.

With this purpose the project was initiated keeping in view the following objectives

- 1) Issue of alert for possible flood situation in district/Circle level with best possible lead time.
- 2) Submission of annual periodic report on status of existing embankments.
- 3) Creating an environment of joint participation among all stakeholders in order to generate actionable product for management of flood in Assam
- 4) Development of optimum methodology for rainfall prediction from satellite based weather monitoring and numerical weather prediction models supported by insitu ground data.
- 5) Development of river specific rainfall-runoff models for forecasting of flood.
- 6) Development of inundation simulation for flood plain zonation.

## **Literature Survey:**

### **Existing system:**

In previous scenarios, performance of many flood forecasting system in an operational context is sub optimal and often below expectation .The information they designed to provide fail to reach

much of the target audience. Existing flood warning systems even with their manifest deficiencies can be effective in the mitigation of flood damage. In India most of the techniques for formulating the real time flood forecast are based on statistical approach. For some project network model and multi parameter hydrological model are used. Conventional systems of communication are normally used for transmitting the data in real time. Flash floods are usually experienced. As such there is no system for formulating the flash flood forecast. It results in heavy losses of lives and properties.

### Proposed system:

There are many earlier works provided by the researchers in the field of IOT but most of them either lacks precision or they are highly expensive. Thus, they are inaccessible to the user. In this module, we are making a device which will sense the possibility of flood, firstly by analyzing values from the IOT device and then checking the weather forecast. The work will not end here, it will keep on reading the values at each and every second and updating if it is higher than threshold. So, by installing it now you can easily save your life as well as your society.

### Block Diagram:

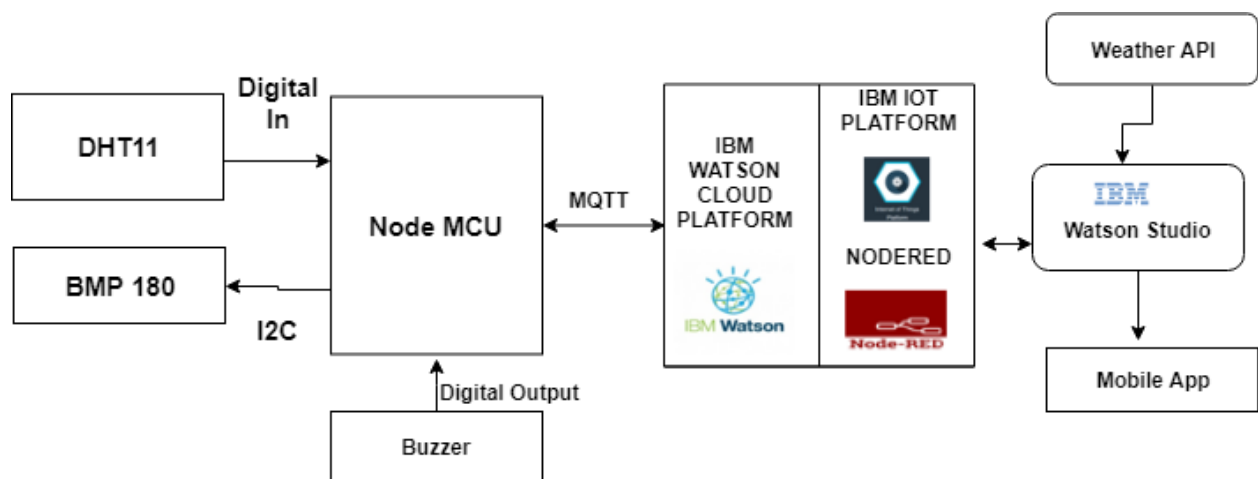


Fig.1: Block Diagram

## Software/Hardware Designing:

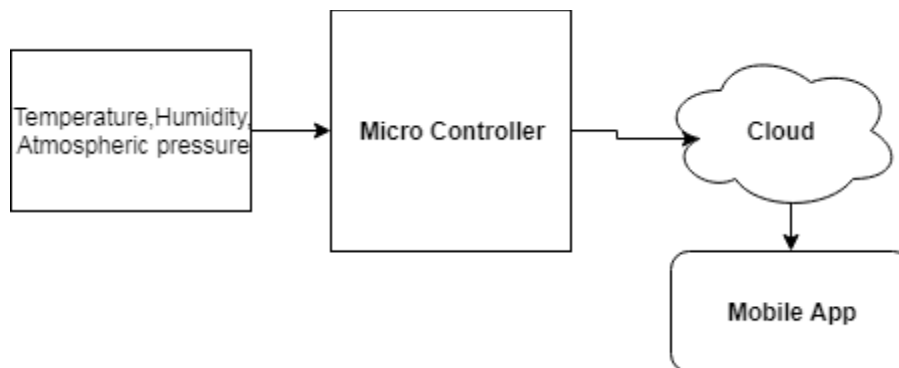
The hardware part of the project involves the ESP8266 model. The sensors DHT11, BMP180 and buzzer are connected to the ESP8266.

The software part of the project involves cloudant DB, Nodered, IBM Watson cloud Platform, iot IBM platform and Web/Mobile app.

## Experimental investigations:

There are several IoT authentication challenges and issues that need to be understood before employing the right security solution that can dynamically vary with the situation based on certain critical situations such as IOT forecasting applications, flood forecasting and early detection are necessary and cloud dynamically vary, potentially resulting in changes to the authorization of iot devices. To protect from floods we used such parameters as temperature, humidity and pressure by using these parameters we can give early warning of floods.

## Flow chart:



## Code using device simulator:

The screenshot shows the IBM Watson IoT Platform interface. The main page is titled "Browse Devices" and includes a table of devices. A modal window is open for configuring a device of type "navya".

**Device Configuration Modal:**

- Device Type:** navya
- Events:** 1
- Event type name:** event\_1
- Schedule:** 20 Every Minute
- Payload:**

```
0 {
1   "temperature": random(0, 100)
2   "humidity": random(0, 100)
3   "pressure": random(0, 100)
4 }
5
```
- Buttons:** Send, Upload a CSV file, Cancel, Save

**Device Table:**

Device ID	Status	Device Type	Class ID	Date Added
12345	Disconnected	navya	Device	5 Jul 2020 11:46

Items per page: 50 | 1-1 of 1 item

**Nodered:**

The screenshot displays the Node-RED web interface with two flows. The top flow, titled "smart health auto ai", starts with an "IBM IoT" node connected to three function nodes: "temperature", "humidity", and "Atmospheric Pressure". These function nodes output to "msg.payload", "humidity", and "atmosphericpressure" respectively. The bottom flow, titled "Flood Prediction Output", starts with an "IBM IoT" node connected to a "get token" node (with a "requesting" label), which then connects to a "Pre Prediction" node. The "Pre Prediction" node outputs to an "output parsing" node, which then connects to a "Flood Prediction Output" node. There is also a "retriving ibm iot health values" node connected to the "get token" node, and an "http request" node connected to the "output parsing" node. The right sidebar shows the debug console with JSON logs of the data flow.

```

    graph LR
      subgraph "smart health auto ai"
        IoT1[IBM IoT] --> temp_f[temperature]
        IoT1 --> hum_f[humidity]
        IoT1 --> AP_f[Atmospheric Pressure]
        temp_f --> msg_payload1[msg.payload]
        hum_f --> humidity_out[humidity]
        AP_f --> atmosphericpressure[atmosphericpressure]
      end

      subgraph "Flood Prediction Output"
        IoT2[IBM IoT] --> get_token[get token]
        get_token -- requesting --> pre_prediction[Pre Prediction]
        pre_prediction --> output_parsing[output parsing]
        output_parsing --> flood_prediction[Flood Prediction Output]
        retriving[retriving ibm iot health values] --> get_token
        http_request[http request] --> output_parsing
      end
  
```

The debug console on the right shows the following logs:

```

    [7/10/2020, 1:16:09 PM] node: 43f97e07.2a45c
    {
      "details": {
        "temperature": 9,
        "humidity": 62,
        "pressure": 77
      }
    }

    [7/10/2020, 1:16:09 PM] node: 43f97e07.2a45c
    {
      "details": {
        "temperature": 85,
        "humidity": 77,
        "pressure": 64
      }
    }
  
```

## Mit app designer:

The screenshot displays the MIT App Inventor web interface in the Designer view. The browser's address bar shows the URL `ai2.appinventor.mit.edu/#5576591581511680`. The app's title is "Flood\_forecasting".

**Palettes:**

- User Interface:** Includes components like Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, Switch, TextBox, TimePicker, and WebViewer.
- Layout:** Includes components like HorizontalArrangement1, HorizontalArrangement2, and HorizontalArrangement3.
- Media:** Includes an Upload File ... button.

**Viewer:** Displays a mobile phone mockup with the following UI elements:

- Status bar: 9:48
- Header: FLOOD FORECASTING AND EARLY WARNING SYSTEM
- Form fields: Temperature, Humidity, and Atmospheric pressure.

**Components:** A list of components on the screen, including Screen1, HorizontalArrangement1, Label1, TextBox1, HorizontalArrangement2, Label2, TextBox2, HorizontalArrangement3, Label3, TextBox3, Clock1, and Web1.

**Properties:** A list of properties for the selected component (Screen1), including AboutScreen, AccentColor, AlignHorizontal, AlignVertical, AppName, BackgroundColor, BackgroundImage, BlocksToolkit, CloseScreenAnimation, Icon, OpenScreenAnimation, PrimaryColor, and PrimaryColorDark.

## Mit app Blocks:

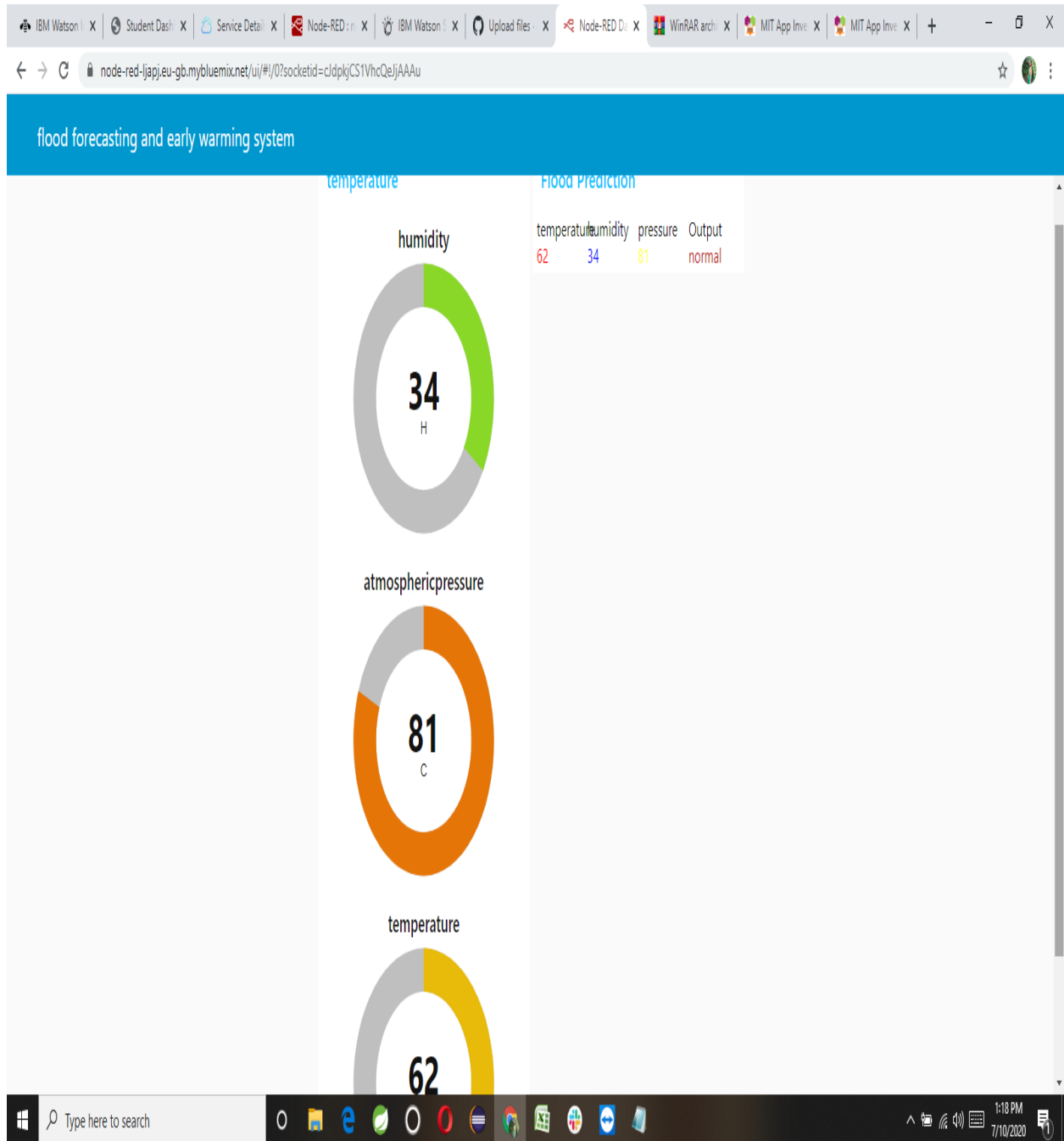
The screenshot displays the MIT App Inventor web interface for a project named "Flood\_forecasting". The interface is divided into several sections:

- Top Bar:** Contains navigation links like "My Projects", "Connect", "Build", "Settings", and "Help". It also shows the user's email address "namburina7@gmail.com".
- Left Sidebar (Blocks Palette):** Lists various building blocks categorized under "Built-in" (Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures) and "Screen1" (HorizontalArranger, Label1, TextBox1, Label2, TextBox2). It includes "Rename" and "Delete" buttons for the screen components.
- Viewer Area:** The central workspace for designing the app. It shows two main event-driven blocks:
  - when Clock1.Timer:** A block that triggers a "do" block containing:
    - set Web1.Uri to https://node-red-ljapi.eu-gb.mybluemix.net/data**
    - call Web1.Get**
  - when Web1.GetText:** A block that triggers a "do" block containing three "look up in pairs" blocks, each for a different data point:
    - temperature:** Uses "Web1.JsonTextDecode" to parse the response and "get responseContent" to retrieve the value, which is then set to "TextBox1".
    - humidity:** Similarly, retrieves the humidity value and sets it to "TextBox2".
    - pressure:** Similarly, retrieves the pressure value and sets it to "TextBox3".

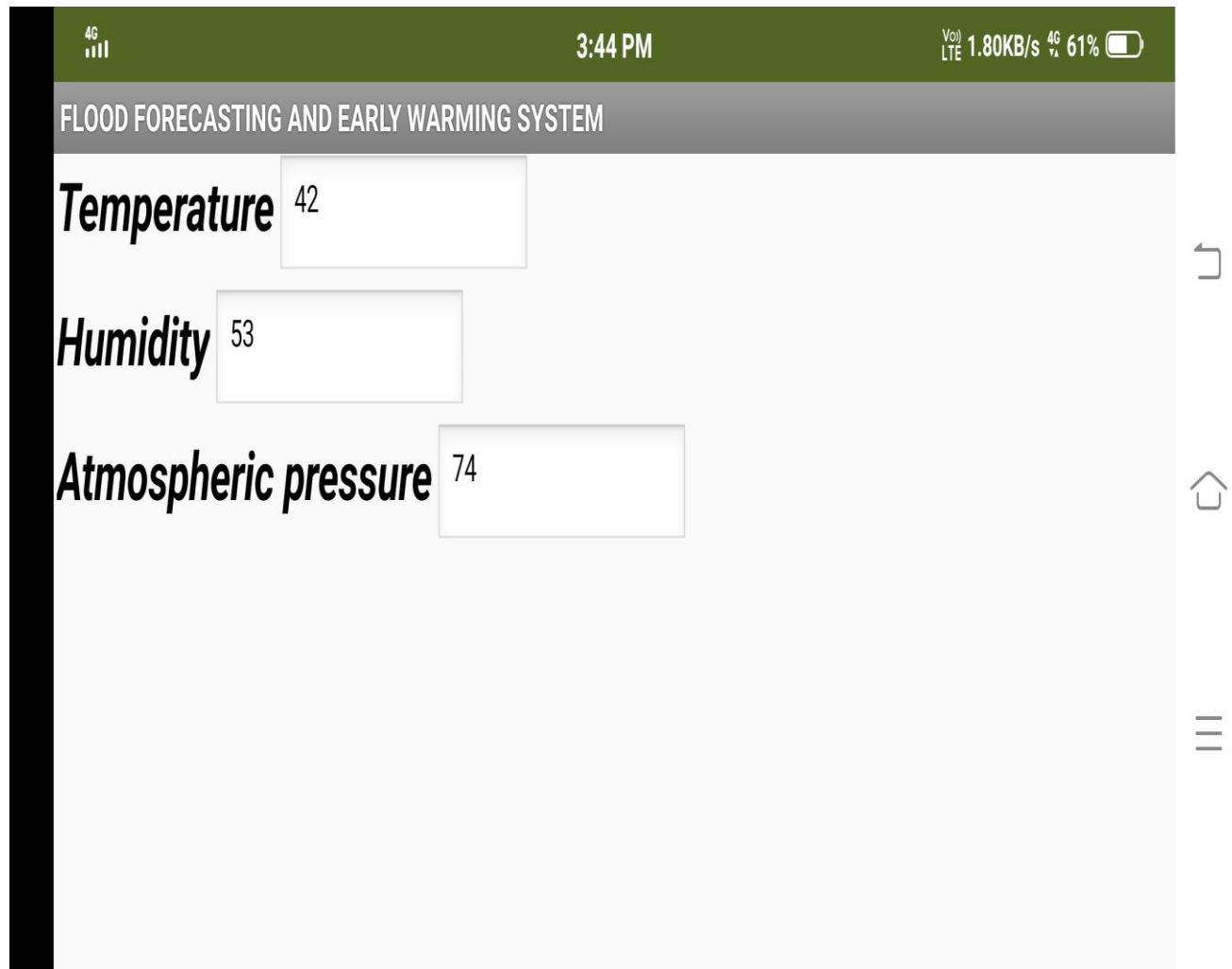
The interface also includes a "Designer" tab, a "Show Warnings" button, and a Windows taskbar at the bottom showing the time as 1:16 PM on 7/10/2020.



## Result:



## Final Result:



## Advantages and Disadvantages:

### Advantages of IoT in Flood Forecasting:

- The impact of flooding is reduced
- Warnings give people time to move possessions upstairs, put sandbags in position and to evacuate

### **Disadvantages of IoT in Flood Forecasting:**

- Warnings don't stop a flood from happening
- Living in a place that gets lots of warnings could make it difficult to get insurance
- People may not hear or have access to warnings

### **Conclusion:**

Cloud software can be used in developing the flood forecasting and warning system with the presence of real time and forecasted rainfall from ground and satellite.

### **Future scope:**

Making a modular system which could be a part of system and can give a real time positioning of flood water.