IoT Based detection of Flood with Arduino UNO and NodeMCU

Shubham Raj,
Department of Chemical Engineering,
180107059
November 5, 2020

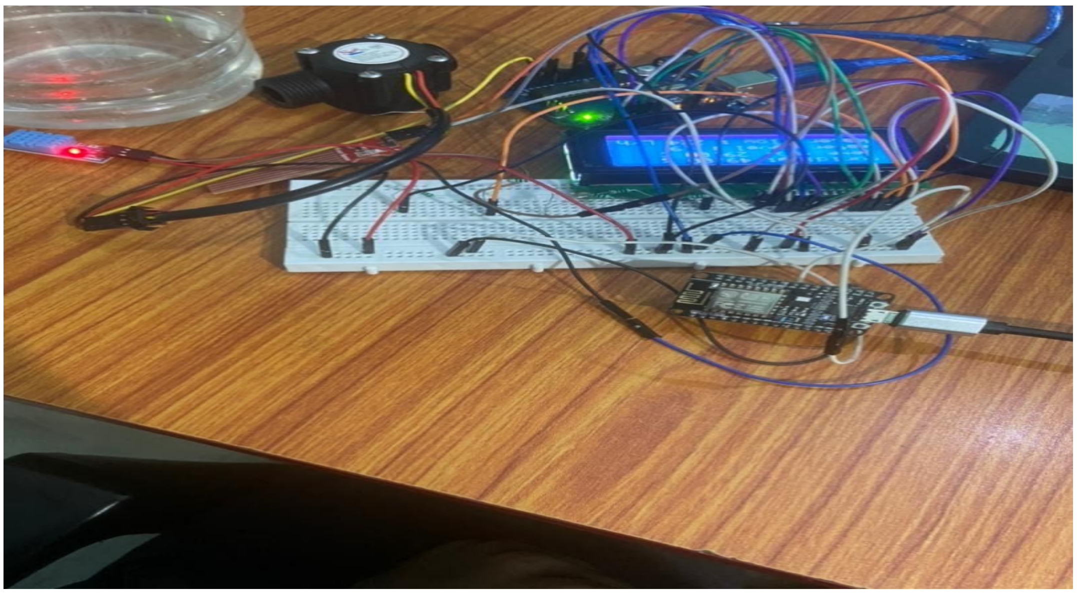
Main Objective - Flood is an inevitable and damaging natural disaster. On the other hand, it creates a huge economic damage and cause a significant loss in human life. Using different sensors,namely DHT11, Water level sensor and Water Flow Sensor, with the of Arduino UNO to measure the parameters then, a NodeMCU is placed in the flood prone areas where the NodeMCU acts as the transmitting unit which consists of above sensors that is used transmitting data to Thingspeak IoT platform and then the data is displayed through the LCD. Proposed model is also useful for us in anticipating the for the coming calamities and to take essential activities by emergency and recover experts to spare the life of thousands of individuals before this basic condition happens. The Flood Detector System using Arduino is created to be one of the quickest strategies to monitor flood and Internet of Things (IoT) is one of the most important technical trends which is utilized to monitor flood and human made resources to help in predicting and detecting essential events like flood, fire, gas and water leak that can position an threats to human life.

Keywords: Flood alert system, Arduino Uno, Internet of Things, DHT11, Water level sensor, Water Flow Sensor, Thingspeak.

Implemented Attributes:-

1. Implemented DHT11 library on arduino to measure Temperature and Humidity.
2. Implemented arduino code to measure water level by Water Level Sensor.
3. Implemented arduino code to measure water flow by measuring flow sensor pulses.
4. Implemented LCD pin out diagram along with Liquid Crystal library.
5. Implemented esp8266 library to transfer data on Thingspeak using api key.

Configuration Diagram:-



Code-

To measure sensor data using Arduino UNO:-

```
#include <LiquidCrystal.h>
#include "DHT.h"

#define DHTPIN 7    // what digital pin we're connected to

#define DHTTYPE DHT11

#define sensorPower 6
#define sensorPin A0

int val = 0;

int readSensor() {
  digitalWrite(sensorPower, HIGH);  // Turn the sensor ON
  delay(10);              // wait 10 milliseconds
  val = analogRead(sensorPin);    // Read the analog value form sensor
  digitalWrite(sensorPower, LOW);   // Turn the sensor OFF
  return val;             // send current reading
}

volatile int flow_frequency; // Measures flow sensor pulses
unsigned int l_hour; // Calculated litres/hour
unsigned char flowsensor = 9; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;
void flow () // Interrupt function
{
   flow_frequency++;
}

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  pinMode(sensorPower, OUTPUT);
  digitalWrite(sensorPower, LOW);
  pinMode(flowsensor, INPUT);
  digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
  Serial.begin(9600);
  //20 by 4 character display
  //If you're using a 16x2 display, change it to lcd.begin(16,2);
   attachInterrupt(0, flow, RISING); // Setup Interrupt
   sei(); // Enable interrupts
   currentTime = millis();
   cloopTime = currentTime;
  lcd.begin(20,4);
  Serial.println("DHT11 test!");
  dht.begin();
}
```

```
void loop() {
  // Wait a few seconds between measurements.
  delay(1000);
  int level = readSensor();


  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  currentTime = millis();
   // Every second, calculate and print litres/hour
   if(currentTime >= (cloopTime + 1000))
   {
      cloopTime = currentTime; // Updates cloopTime
      // Pulse frequency (Hz) = 7.5Q, Q is flow rate in L/min.
      l_hour = (flow_frequency * 60 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in
L/hour
      flow_frequency = 0; // Reset Counter
    }

  dht.read(h);
  dht.read(t);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Humidity: ");
  lcd.print(h);
  lcd.print(" %");
  lcd.setCursor(0,1);
  lcd.print("Temp (C): ");
  lcd.print(t);
  lcd.print(" C");
  lcd.setCursor(0,2);
  lcd.print("Water level : ");
  lcd.print(level);
  lcd.setCursor(0,3);
  lcd.print("Water Flow : ");
  lcd.print(l_hour);
  lcd.print(" L/hour");

  //Serial monitor output
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\n");
  Serial.print("Temp (C): ");
  Serial.print(t);
  Serial.print(" C\n");
```

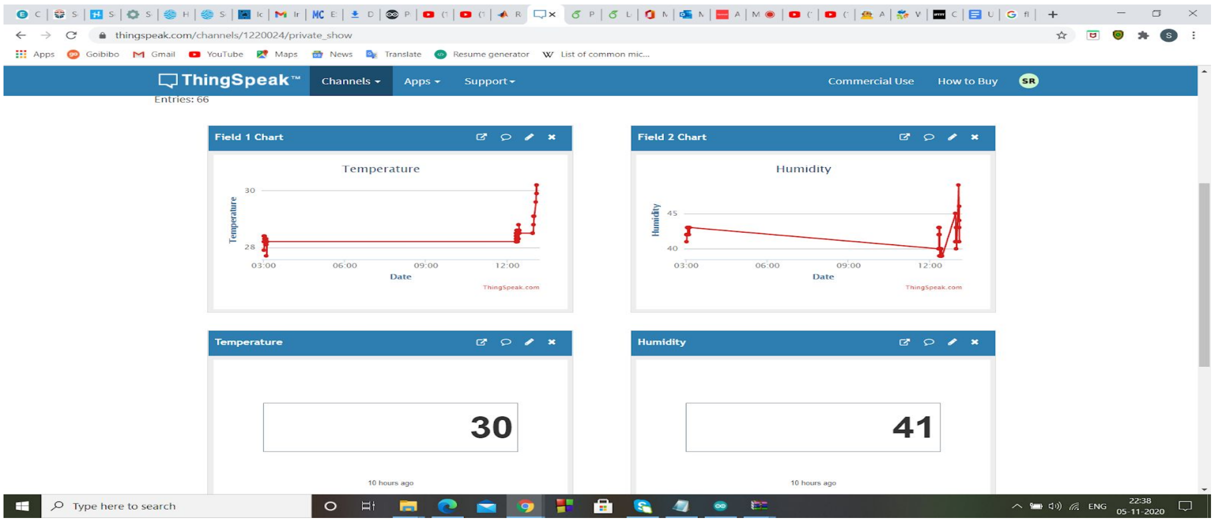
```
 Serial.print("Water level : ");
 Serial.print(level);
 Serial.print("\n");
 Serial.print("Water flow : ");
 Serial.print(l_hour);
 Serial.print("L/hour\n");
 Serial.print("_____________________________\n");
}
```

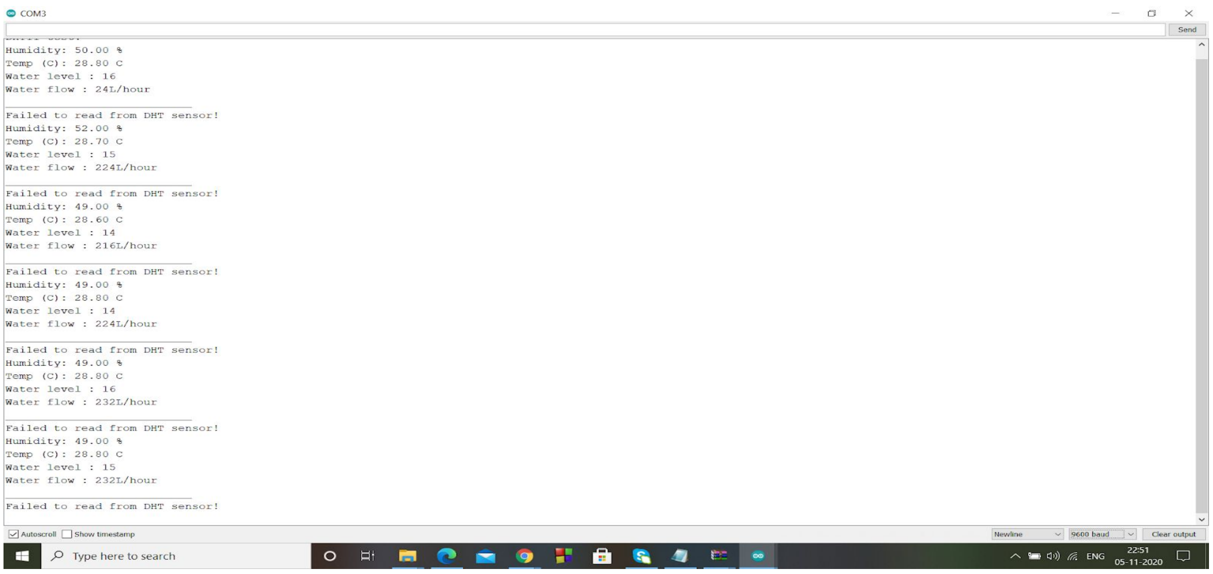
To transmit data to Thingspeak:-

```
#include <DHT.h>  // Including library for dht

#include <ESP8266WiFi.h>

String apiKey = "3QBVICOQRO79EBBO";     //  Enter your Write API key from ThingSpeak

const char *ssid =  "iPhone";     // replace with your wifi ssid and wpa2 key
const char *pass =  "wyw5c3g1uzjp6";
const char* server = "api.thingspeak.com";

#define DHTPIN 0          //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);

WiFiClient client;

void setup()
{
       Serial.begin(115200);
       delay(10);
       dht.begin();

       Serial.println("Connecting to ");
       Serial.println(ssid);


       WiFi.begin(ssid, pass);

      while (WiFi.status() != WL_CONNECTED)
     {
            delay(500);
            Serial.print(".");
     }
      Serial.println("");
      Serial.println("WiFi connected");

}

void loop()
{

      float h = dht.readHumidity();
      float t = dht.readTemperature();
```

```
          if (isnan(h) || isnan(t))
            {
              Serial.println("Failed to read from DHT sensor!");
               return;
            }

                  if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
                {

                      String postStr = apiKey;
                      postStr +="&field1=";
                      postStr += String(t);
                      postStr +="&field2=";
                      postStr += String(h);
                      postStr += "\r\n\r\n";

                      client.print("POST /update HTTP/1.1\n");
                      client.print("Host: api.thingspeak.com\n");
                      client.print("Connection: close\n");
                      client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                      client.print("Content-Type: application/x-www-form-urlencoded\n");
                      client.print("Content-Length: ");
                      client.print(postStr.length());
                      client.print("\n\n");
                      client.print(postStr);

                      Serial.print("Temperature: ");
                      Serial.print(t);
                      Serial.print(" degrees Celcius, Humidity: ");
                      Serial.print(h);
                      Serial.println("%. Send to Thingspeak.");
                  }
        client.stop();

        Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates
  delay(1000);
}



//Arduino codes with required libraries.
```

Sample Outputs:-




```
Humidity: 50.00 %
Temp (C): 28.80 C
Water level : 16
Water flow : 24L/hour

Failed to read from DHT sensor!
Humidity: 52.00 %
Temp (C): 28.70 C
Water level : 15
Water flow : 224L/hour

Failed to read from DHT sensor!
Humidity: 49.00 %
Temp (C): 28.60 C
Water level : 14
Water flow : 216L/hour

Failed to read from DHT sensor!
Humidity: 49.00 %
Temp (C): 28.80 C
Water level : 14
Water flow : 224L/hour

Failed to read from DHT sensor!
Humidity: 49.00 %
Temp (C): 28.80 C
Water level : 16
Water flow : 232L/hour

Failed to read from DHT sensor!
Humidity: 49.00 %
Temp (C): 28.80 C
Water level : 15
Water flow : 232L/hour

Failed to read from DHT sensor!
```

User Manual:-

1.Make the connections on breadboard.
2.Write the Arduino Code.
3.Compile and Upload it by choosing the correct board and port.
4.Observe the outputs on serial monitor and LCD.
5.Make an account on Thinspeak.
6.Make a channel on it and create the necessary visualizations.
7.Note the API key and use it upload data from nodemcu via esp8266.
8.Observe the data and trigger response when the data is above threshold.