Guru Jambheshwar University of Science and Technology, Hisar

A Document on mini-project entitled

Smart Agriculture Water Storage System

Submitted by: The TechBuds

Team Members:

Pawan Kumar – IT 2nd Year

Lalit Kumar - IT 2nd Year

Subham Saini - IT 2nd Year

Sahib - ECE 2nd Year

Under the guidance of:

Mr. Mahankali Surya Tej

Mr. Akshay Kothuri

During the Summer Training of Internet of Things in 27 June- 23 July 2018.

CONTENTS:

Abstract:

The project describes the smart irrigation water storage system using the concept of IoT.

Wastage of water in the current scenario, merely due to overflowing tanks is not affordable. Conventional water tanks can neither monitor nor control the water level in tank, leading to large amount of wastage.

Plant monitoring is seen as one of the most important tasks in any farming or agriculture-based environment. Monitoring of field irrigation system reduces human intervention and allows remote monitoring and controlling. Plants are very sensitive to several factors such as in soil moisture and weather changes and lack of monitoring. Automatic Water supply and Monitoring System can help to resolve the problem.

The smart objects embedded with sensors enables interaction with the physical and logical worlds according to the concept of IoT. In this paper proposed system is based on IoT that uses real time input data.

In this paper, we are discussing a sustainable smart irrigation water storage system based on Internet of Things (IoT), which automates the water distribution and storage as well as regulation of water wastage.

The need of removal of these short-comings and providing an efficient and economical solution has been the main focus of this project. The results of the proposed design, and its evaluation are described in this paper.

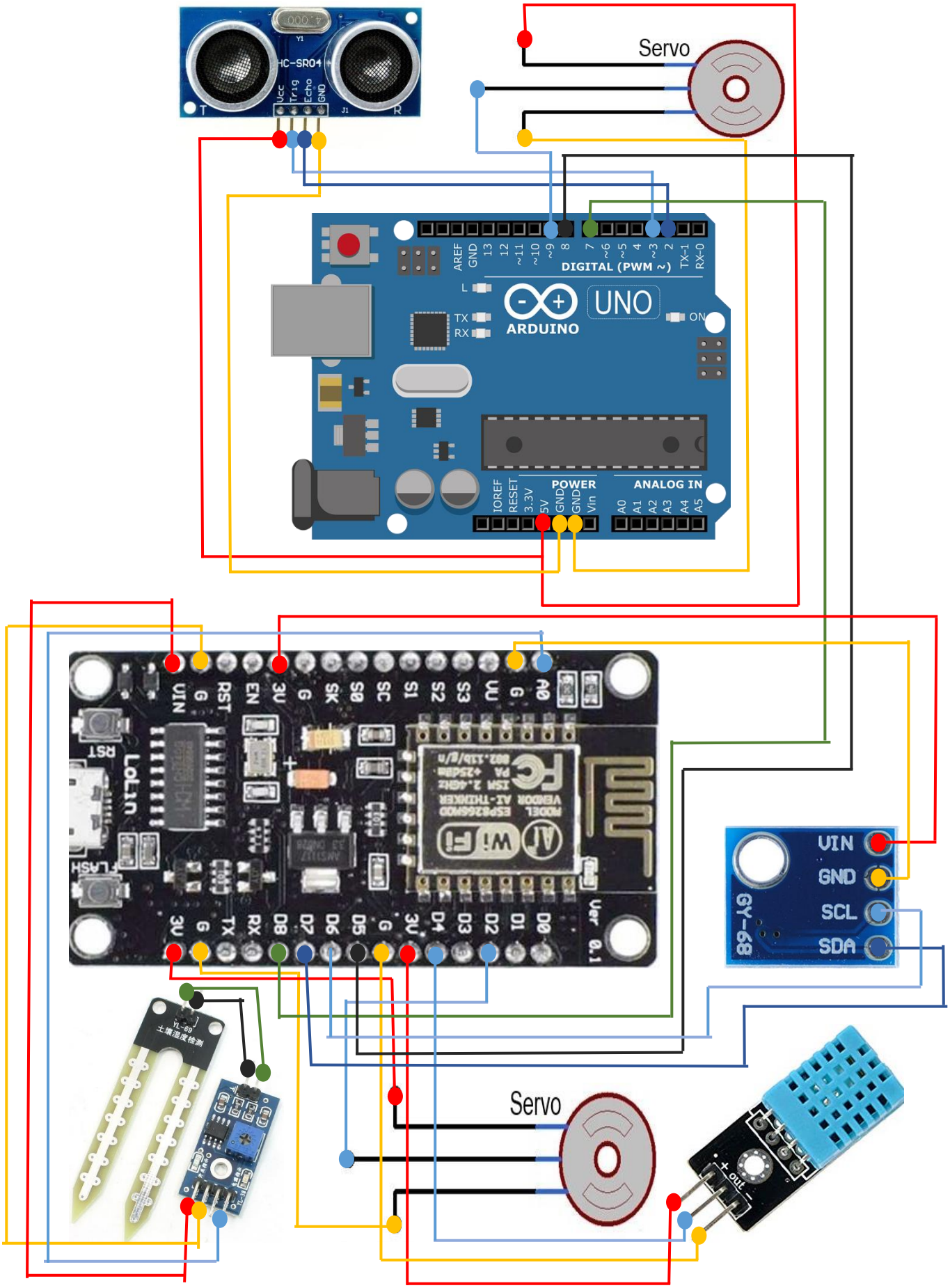
Introduction:

India is the country of village and agriculture plays and important role for development of country and in agriculture water plays a significant role. In our country, agriculture depends on the monsoons which has insufficient source of water. So, the irrigation is used in agriculture field. In irrigation system, depending upon the soil type, water is provided to plant. In agriculture, two things are very important, first to get information of about the fertility of soil and second to measure moisture content in soil.

Water storage and monitoring constantly is also the important matter. Whenever no one is home then the tank will be automatically full and tap will be automatically closed and water will be provided to the plant whenever the soil moisture will be dry.

This project uses Arduino which in turn is connected to ultrasonic sensor which provides the water level in the water tank and sends this data to esp8266-12 and nodemcu is connected to soil moisture sensor to measure the moisture in the soil and DHT11 and BMP180 to predict the weather. Two solenoid valves are connected to close the water supply tank is full or whenever the soil moisture is enough. Esp8266 is used to send data to internet (IBM Cloud) by using the MQTT protocol.

Implementation: Circuit Diagram



Implementation: Explanation of each block

1. **Arduino Board:**

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

The Uno is one of the more popular boards in the Arduino family and in this project also Arduino Genuino Uno board is used.

The pins on your Arduino are the places where you connect wires to construct a circuit probably in conjuction with a breadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

1. GND (3): Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

2. 5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

3. Analog (6): The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

4. Digital (13): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

5. PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for called Pulse-Width Modulation (PWM).

6. AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

7. Reset Button: Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

8. Power LED Indicator: Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

9. TX RX LEDs: TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

2. ESP8266 / Nodemcu:

NodeMCU is an open source IoT platform. Which includes firmware which runs on the ESP8266 Wi-Fi Module from Espressif Systems,and hardware which is based on the ESP-12 module.

The term “NodeMCU” by default refers to the firmware rather than the dev kits. NodeMCU firmware was developed so that AT commands can be replaced with Lua scripting making the life of developers easier.

So it would be redundant to use AT commands again in NodeMCU. NodeMCU is a Firmware on ESP8266.

Its basically an SoC (System on Chip) A System on a Chip or System on Chip(SoC) is an integrated circuit that integrates all components of a computer or other electronic systems.

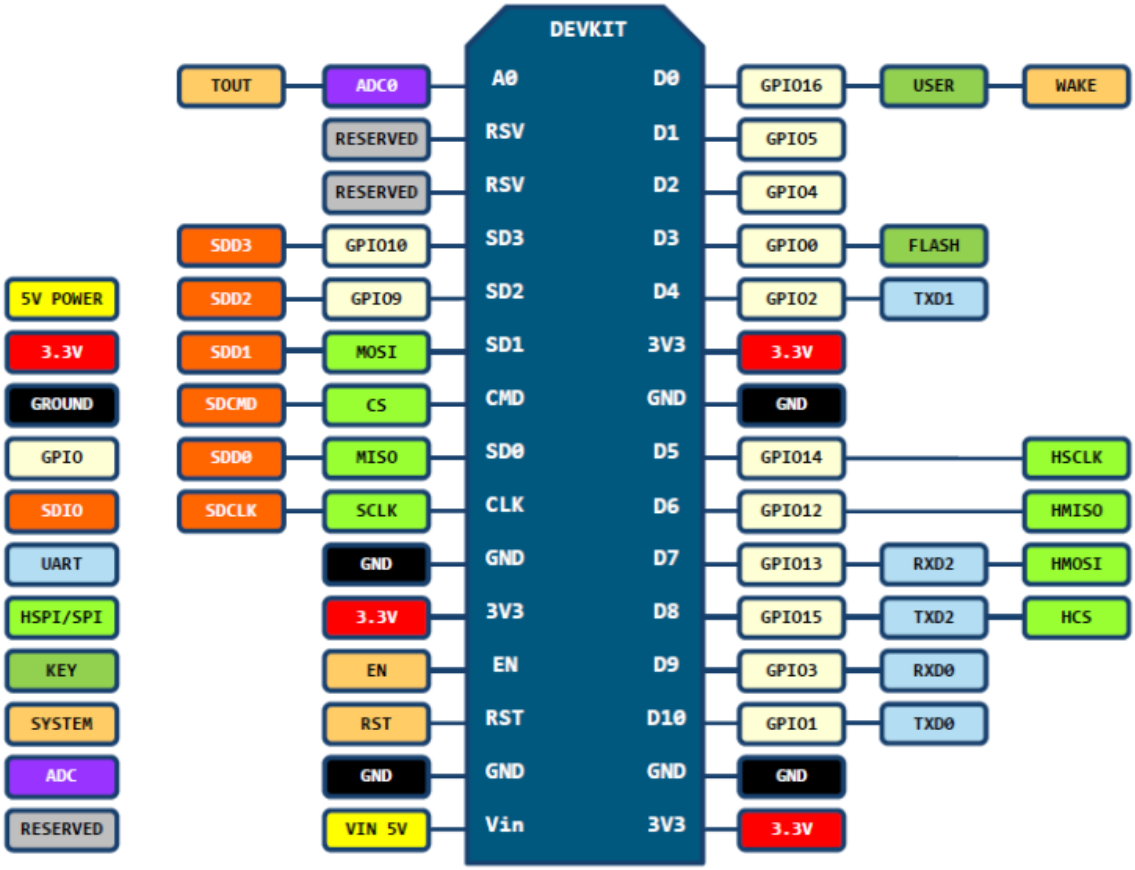
The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif.

ESP8266 Feature:

- Open-source
- Interactive
- Programmable
- Low cost
- Simple
- Smart
- WI-FI enabled
- USB-TTL included
- Plug & Play
- NodeMCU DEVKIT 1.0 Specification:
- Developer : ESP8266 Opensource Community
- Type : Single-board microcontroller
- Operating system : XTOS
- CPU : ESP8266
- Memory : 128kBytes

- Storage : 4Mbytes
- Power By : USB
- Power Voltage : 3v ,5v (used with 3.3v Regulator which inbuilt on Board using Pin VIN)
- Code : Arduino Cpp
- IDE Used : Arduino IDE

PIN DEFINITION



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

3. UltraSonic Sensor:

HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are VCC, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the

front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module.

4. **Servo Motor:**

You can connect small servo motors directly to an Arduino to control the shaft position very precisely. Because servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small in size, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino.

Most servo motors have the following three connections:

- Black/Brown ground wire.
- Red power wire (around 5V).
- Yellow or White PWM wire.

In this experiment, we will connect the power and ground pins directly to the Arduino 5V and GND pins. The PWM input will be connected to one of the Arduino's digital output pins.

5. **DHT11 Sensor:**

DHT11 Sensor is used to measure temperature and humidity. The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two

electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

The DHT11 measures temperature with a surface mounted NTC temperature sensor (thermistor) built into the unit.

6. **BMP180 Sensor:**

   This precision sensor from Bosch is the best low-cost sensing solution for measuring barometric pressure and temperature. Because pressure changes with altitude you can also use it as an altimeter! The sensor is soldered onto a PCB with a 3.3V regulator, I2C level shifter and pull-up resistor.

   This board is 5V compliant - a 3.3V regulator and a i2c level shifter circuit is included so you can use this sensor safely with 5V logic and power. s on the I2C pins.

7. **Moisture Sensor (YL-38 & YL-69):**

The soil moisture sensor or the hygrometer is usually used to detect the humidity of the soil. So, it is perfect to build an automatic watering system or to monitor the soil moisture of your plants.

The sensor is set up by two pieces: the electronic board and the probe with two pads, that detects the water content.

The voltage that the sensor outputs changes accordingly to the water content in the soil.

When the soil is:

Wet: the output voltage decreases

Dry: the output voltage increases

The YL-39 module has 4 pins:

- VCC: 3.3-5V
- GND
- A0: analog output that can be easily read by Arduino
- D0: digital pin that goes LOW or HIGH depending on a preset value.

Implementation: Explanation

This project uses Arduino which in turn is connected to ultrasonic sensor. This ultrasonic sensor is hanged on to a water tank which measures the water level in the water tank and as water level arises to a certain limit a signal will be sent to the servo motor to close the tap.

This distance value is constantly being sent to the esp8266-12 and this value is then sent to IBM Cloud.

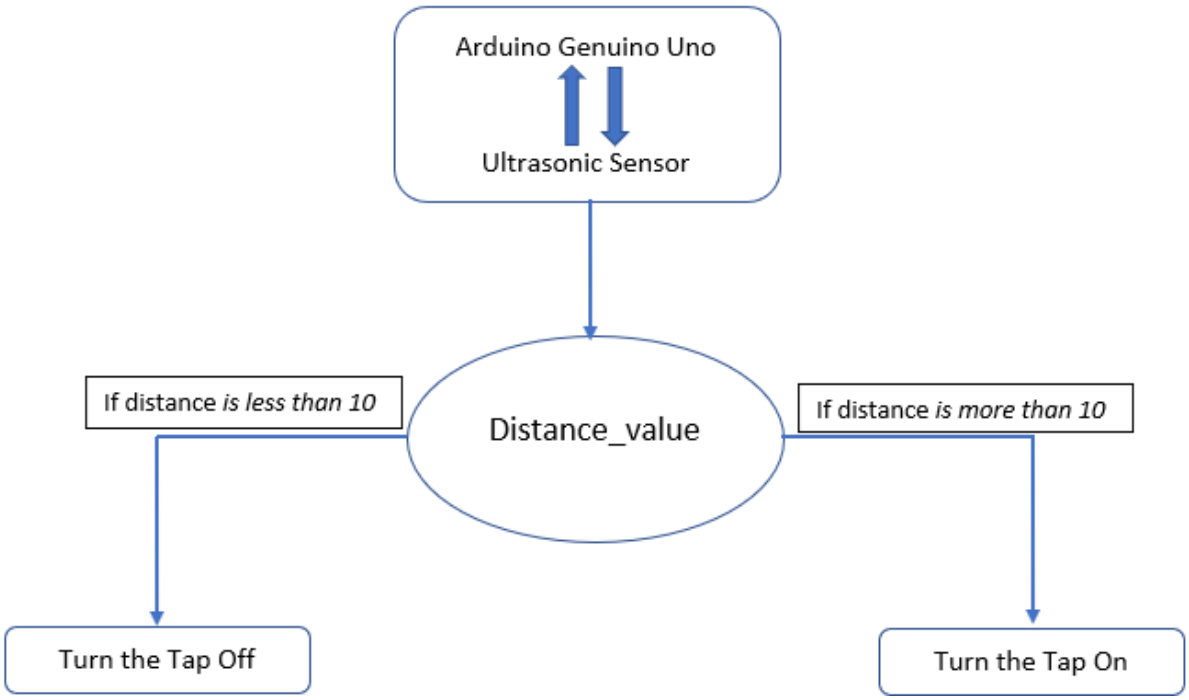
One the other way, Nodemcu is connected to Arduino using serial communication UART to send information from Arduino to ESP8266 and Nodemcu is connected to soil moisture sensor to measure the moisture in the soil and whenever the value gets dry then another tap will be open and then after watering the plant tap will be automatically closed.

DHT11 and BMP180 sensors are used to predict the weather means used for weather monitoring system. Whenever the rain will be predicted then Tap won’t be open.

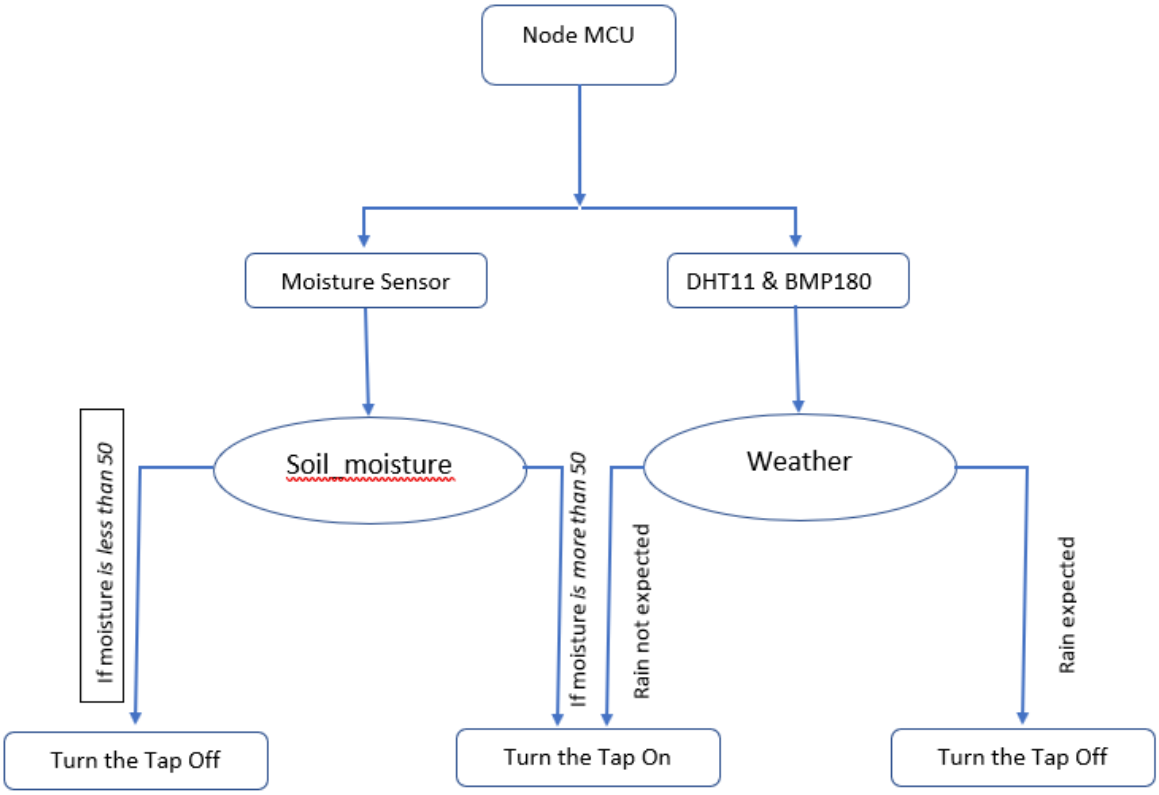
Esp8266 is used to send data to internet (IBM Cloud) by using the MQTT protocol. Then from there the value can be sent to mobile app or can be further analysed.

Implementation: Flow Chart

Flow Chart 1:



Flow Chart 2:



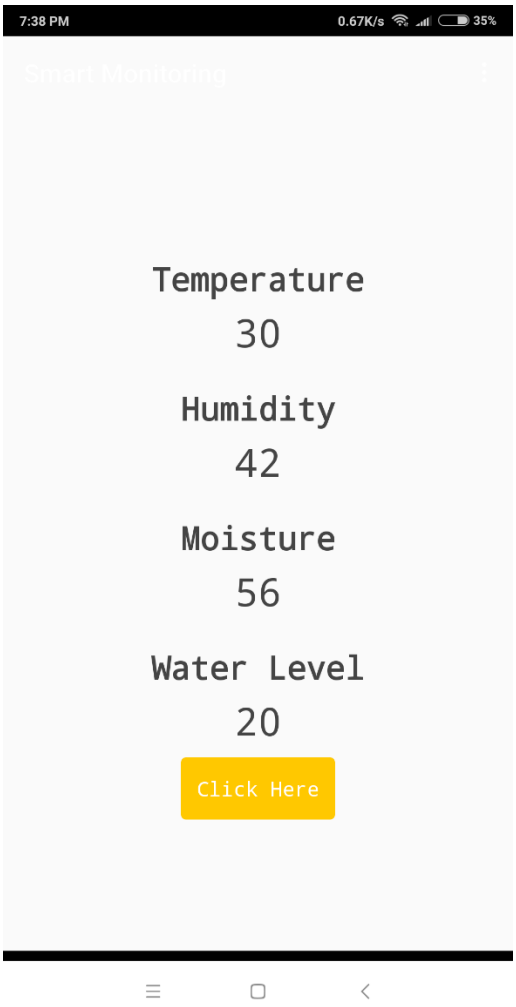
Results and Discussion:

Four parameters namely moisture of soil, humidity, temperature and water level are measured using the experimental setup.

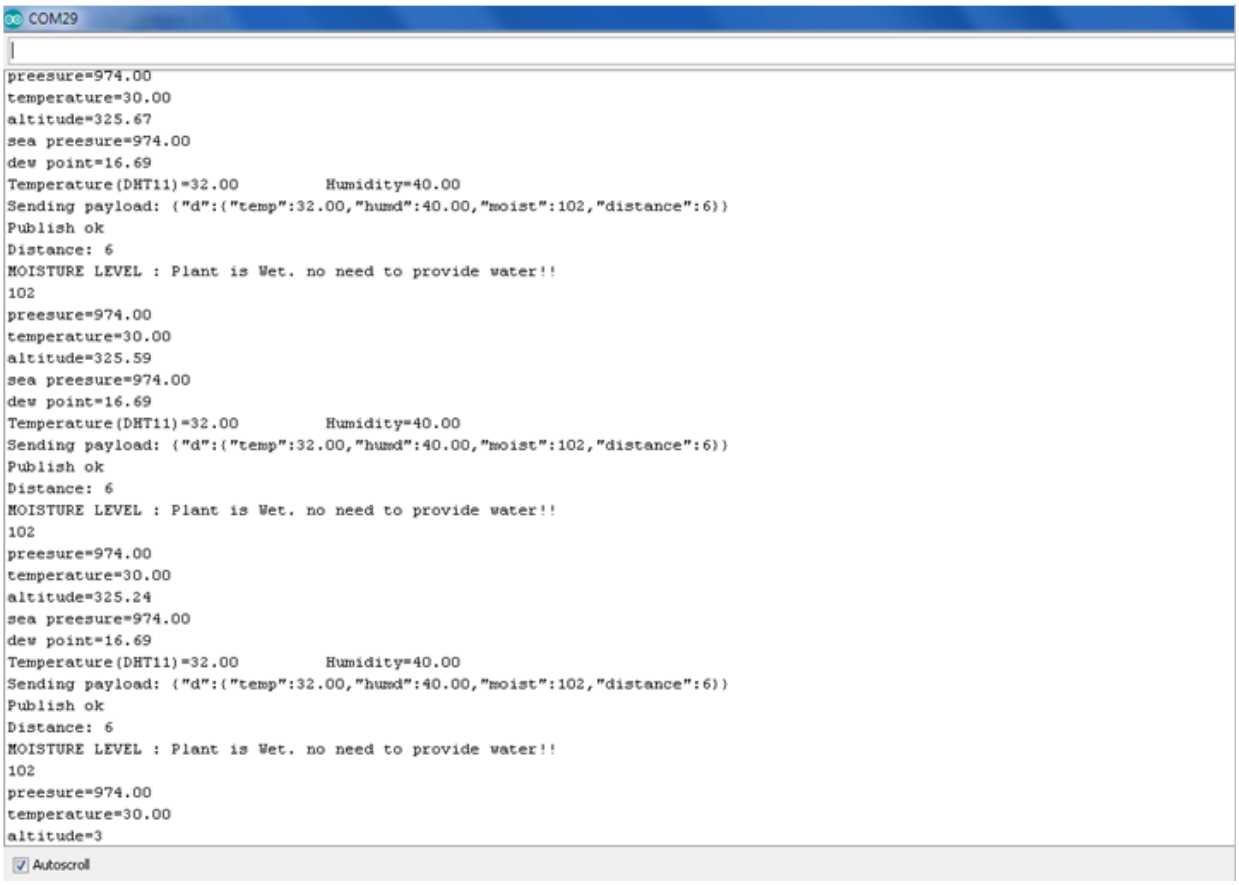
The setup is connected to the IBM Cloud. The measured result is then used to show the data onto the mobile app using MIT App Inventor 2.

Whenever the distance between water and ultrasonic sensor is less than 10 it will automatically close the tap and the send the data to the cloud consistently. On the other side, when moisture is below 50 then it will automatically close the tap and analyse temperature closely.

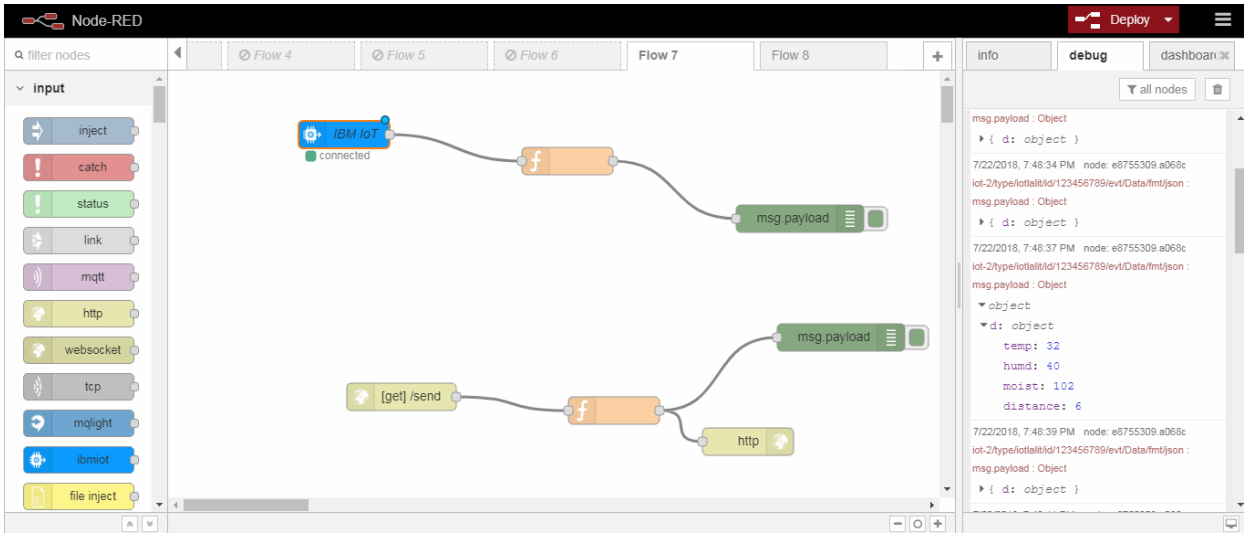
Here are some of the Screenshots:



Screen Shot of App Receiving Values



Screenshot of Serial Monitor Sending Values



Screenshot of Node Red Receiving Values and Nodes Used

Advantages and Disadvantages:

Advantages:-

- ❖ Helps in saving Time.
- ❖ Water saving
- ❖ Better for plant growth
- ❖ Whether prediction
- ❖ Easy to maintain
- ❖ Easy to Establish set up

Disadvantages:-

- ❖ Continuous power supply required
- ❖ Need internet and wi-fi services
- ❖ Need some area for storage tank
- ❖ Need some basic knowledge of technology

Future Scope:

The basics need of human being is water and it is one of the most important necessity for all living beings. But unfortunately, a huge amount of water is being wasted by uncontrolled use and due to our negligence. Some other automated water level monitoring system is also offered so far but most of the method has some shortness in practice.

We tried to overcome these problems and implemented an efficient automated water level monitoring and controlling system which can be used for smart irrigation as well. Main intension of this research work is to establish a flexible, economical and easy configurable system which can solve water losing problems and can help nourishing the plants well. In the near future as home automation web based water level monitoring and controlling system can be designed, through which the system can be controlled from any place via internet through mobile phone. This could have a substantial benefit from this research work for efficient management of water as well as for home automation for smart gardens.

Appendix:

Source Code of Arduino Genuino Uno

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial ArduinoUno(7,8);
Servo myservo;
int pos = 0;
String f;
void setup() {
  Serial.begin(9600);
  ArduinoUno.begin(115200);
  pinMode(3,OUTPUT);
  pinMode(2,INPUT);
  myservo.attach(9);
}

void loop() {
digitalWrite(3,HIGH);
delay(10);
digitalWrite(3,LOW);
int lp = pulseIn(2,HIGH);
int d = (lp/2)*0.0343;
if(d<=10) {
  Serial.println(d);
  Serial.print("less");
  for(pos=0;pos<=180;pos++) {
    myservo.write(pos);
```

```
    delay(15/10);


  }
}
else {
  Serial.println(d);
  Serial.print("more");
  for(pos=180;pos<=0;pos--) {
    myservo.write(pos);
    delay(15);


  }
}
f = String(d);
Serial.println("-----------------------------------");
Serial.print("\t Distance - "+String(d) + "\n");
Serial.println("-----------------------------------");
ArduinoUno.print(f);
delay(2000);
}
```

Source Code of Node MCU (ESP8266)

```
//Including Libraries
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <Wire.h>
```

```
#include <Adafruit_BMP085.h>
#include <PubSubClient.h>
//Setting Hotspot Pwd and SSID
const char* ssid = "Redmi Not 5 pro";
const char* password = "123456789";

//Declaring Sensors object and varibles globally.....
SoftwareSerial NodeMCU(D5,D8);
#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE); //Defining the pin and the dhttype
Adafruit_BMP085 bmp;
#define I2C_SCL D6
#define I2C_SDA D7
float dst,bt,bp,ba,t,h,dp;
String distance;
bool bmp085_present=true;
int sense_Pin = A0; // sensor input at Analog pin A0
int value = 0,value1;

//Device Credentials
#define ORG "gm69yw"
#define DEVICE_TYPE "iotlalit"
#define DEVICE_ID "123456789"
#define TOKEN "lalit123456"

// Default values of IBM Cloud in MQTT Protocol.....
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/Data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

// WIFI Connection to connect it through mqtt
WiFiClient wifiClient;
PubSubClient client(server, 1883,wifiClient);
// Void setup Code to be run once only....
void setup() {
  Serial.begin(9600); //Baud Rate 115200
  NodeMCU.begin(115200);
  pinMode(D5,INPUT);//rx
  pinMode(D8,OUTPUT);//tx
  dht.begin(); // DHT Sensor Begin
  Wire.begin(I2C_SDA, I2C_SCL); // Setting BMP 180 GPIO Pins in
Nodemcu
  Serial.print("Connecting to "); //Priting Connecting to SSID_name
  Serial.print(ssid);
  WiFi.begin(ssid, password); // WIFI is trying to connect
  // if wifi is not connected print dots(.) with delay of 0.5sec
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");// new line
```

```
  Serial.print("WiFi connected, IP address: ");// when wifi is connected the print wifi connected IP Address: wifi.localIP()
  Serial.println(WiFi.localIP());
}
//Void loop code to repeat always.....
void loop() {

  delay(2000);
  //Serial communication btw arduino and nodemcu.........

  String content = "";
  char character;
  while(NodeMCU.available()) {
      character = NodeMCU.read();
      content.concat(character);
  }
  if (content != "") {
    Serial.println("Distance: " + content);
  }
  distance = content;
  //.............................................................
  // if bmp wont work then print below statement
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085 sensor, check wiring!");
    return;
  }
   h = dht.readHumidity(); // Humidity by DHT11
   t = dht.readTemperature(); // Temperature by DHT11
```

```
  if (isnan(h) || isnan(t)) { // if dht11 don't work print below statement
    Serial.println("Failed to read from DHT sensor!");
    //return;
  }
  double gamma = log(h/100) + ((17.62*t) / (243.5+t)); //The dew point is the temperature where water vapour condenses into liquid water.
  dp = 243.5*gamma / (17.62-gamma);// dp means dew point - he dew point shows the amount of moisture in the air. The higher the dew point is,
  bp =  bmp.readPressure()/100;// this is pressure in pascal //the higher the level of moisture in the air at a given temperature.
  ba =  bmp.readAltitude(); // altitude
  bt =  bmp.readTemperature(); // Temperature in BMP180 USE either of one BMP Temperature or DHT11 Temperature...
  dst = bmp.readSealevelPressure()/100; // Sea level Pressure

  // Moisture sensor code
  Serial.print("MOISTURE LEVEL : ");
  value= analogRead(sense_Pin);
  value1 = value/10;
  if(value1<50) {
    Serial.println("Plant is Wet. no need to provide water!!");
  }
  else {
    Serial.println("Plant is Wet. no need to provide water!!");
  }
  Serial.println(value1);
  Serial.println("preesure="+String(bp));
  Serial.println("temperature="+String(bt));
```

```
  Serial.println("altitude="+String(ba));
  Serial.println("sea preesure="+String(dst));
  Serial.println("dew point="+String(dp));
  Serial.println("Temperature(DHT11)="+String(t) + "\t Humidity="+String(h));
  PublishData(t, h, dp, bp, ba,bt,dst,value1, distance); // publish data by sensors
  if (!client.loop()) { // call function mqttconnect function until completed function be in loop
    mqttConnect();
  }
 // delay(100);
}
void mqttConnect() { // try to connect to ibm device specific server
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to ");
    Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
    Serial.println();
  }
}



void PublishData(float t, float h, float dp, float bp, float ba, float bt, float dst, int value1, String distance) { // t, h, dp, bp, ba, bt, dst, value1 as arguments
```

```
 if (!!!client.connected()) { // again try to reconnect
   Serial.print("Reconnecting client to ");
   Serial.println(server);
   while (!!!client.connect(clientId, authMethod, token)) {
     Serial.print(".");
     delay(500);
   }
   Serial.println();
 }
 // adding all to one url
 String payload = "{\"d\":{\"temp\":";
 payload += t;
 payload +="," "\"humd\":";
 payload += h;
/* payload +="," "\"dew\":";
 payload += dp;
 payload +="," "\"pressr\":";
 payload += bp;
 payload +="," "\"altitude\":";
 payload += ba;
 payload +="," "\"bmp_temp\":";
 payload += bt;
 payload +="," "\"sea_pressure\":";
 payload += dst;*/
 payload +="," "\"moist\":";
 payload += value1;
 payload +=","\"distance\":";
```

```
  payload += distance;
  payload +="}}";
  // printing all payload
  Serial.print("Sending payload: ");
  Serial.println(payload);

  if(client.publish(topic, (char*) payload.c_str())) {
    Serial.println("Publish ok");
  }
  else {
    Serial.println("Publish failed");
  }
  //delay(1000);
}
```