**ABSTRACT:**

Deficiency in fresh water resources globally has raised serious alarms in the last decade. Efficient management of water resources play an important role in the agriculture sector. Unfortunately, this is not given prime importance in the third world countries because of adhering to traditional practices. This paper presents a smart system that uses a bespoke, low cost soil moisture sensor to control water supply in water deficient areas. The sensor, which works on the principle of moisture dependent resistance change between two points in the soil, is fabricated using affordable materials and methods. Moisture data acquired from a sensor node is sent through XBEE wireless communication modules to a centralized server that enable the farmer to visualize the moisture levels of the fields through the IBM cloud user-friendly interface.The low-cost and wireless nature of the sensing hardware presents the possibility to monitor the moisture levels of large agricultural fields. Moreover, the proposed moisture sensing method has the ability to be incorporated into an automated mobile controlled irrigation scheme.

**PROBLEM STATEMENT:**

Irrigation is mostly done using canal systems in which water is pumped into fields after regular interval of time without any feedback of water level in field. This type of irrigation affects crop health and produces a poor yield because some crops are too sensitive to water content in soil. A large agricultural field presents an additional problem in the sense that different parts areas of it may have different evaporation rates due to foliage, the presence of rocks at different heights underground, parts of the field being in close proximity to canals or ponds, etc.

Inorder to solve these problems the smart irrigation system using wireless sensor network has been devised.This method is contrary to a traditional irrigation method, regulates supplied water according to the needs of the fields and crops using moisture sensor mechanism.

**EXPERIMENTATION:**

The experiment was conducted by placing the moisture sensor of router XBee module in a plastic pot which is filled with soil.

At the start, the sensor was placed in soil in the middle of the plastic pot, and the sensor gives the moisture value. This value is transmitted from the router XBee module to the Co-Ordinator XBee module through wireless communication. Now, the moisture value from the co-ordinator XBee module is transmitted to the personal computer through serial communication, and then to the farmer through the IBM cloud interface for monitoring the temperature, humidity and moisture. If(when moisture sensor placed in dry soil) the moisture value is greater than the optimum value then the sprinkler motor is turned on else(when placed in wet soil) the motor is turned off.

**SPECIFICATIONS OF COMPONENTS USED:**

NODEMCU ESP8266  
**Developer :**  ESP8266 Opensource Community  
**Type :**   Single-board microcontroller  
**Operating system :** XTOS  
**CPU :**  ESP8266  
**Memory :**  128kBytes  
**Storage :**  4MBytes  
**Power By :**  USB  
**Power Voltage :**  3v ,5v   
**Code :**  Arduino Cpp  
**IDE Used :**  Arduino IDE  
**GPIO :**  10

DHT11 SENSOR:

Operating Voltage: 3.5V to 5.5V. Operating current: 0.3mA (measuring) 60uA (standby) Output: Serial data. Temperature Range: 0°C to 50°C. Humidity Range: 20% to 90% Resolution: Temperature and Humidity both are 16-bit. Accuracy: ±1°C and ±1%

SOIL MOISTURE:

Range: 210mtr Moisture: 0-100%,8bit Operation range: -40°C (-40°F) to 85°C (185°F)  
Sensor accuracy: +/-1°C typical, -2/+4°C max  
Sensor quantization level (resolution): 10-bit, about 0.25°C (0.45°F)

XBEE MODULE:

Maximum input RF level at antenna port :6 dBm Operating temperature: -40°C to 85°C Digital I/O: 13 I/O lines,5 output lines Analog-to-digital converter (ADC): 4 10-Bit analog inputs Pulse width modulator (PWM): 2 outputs

MINI WATER PUMP:

Operating Voltage : 2.5 ~ 6V Operating Current : 130 ~ 220Ma Flow Rate : 80 ~ 120 L/H Maximum Lift : 40 ~ 110 mm Continuous Working Life : 500 hours Driving Mode : DC, Magnetic Driving Material : Engineering Plastic Outlet Outside Diameter : 7.5 mm Outlet Inside Diameter : 5 mm

**CODE AT THE XBEE CO-ORDINATE SIDE:**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

const char\* ssid = "lenovo k8 note";

const char\* password = "chikki09";

#include "DHT.h"

#define DHTPIN D2

#define DHTTYPE DHT11

DHT dht (DHTPIN, DHTTYPE);

#define ORG "f6rm4b"

#define DEVICE\_TYPE "gnits.s"

#define DEVICE\_ID "3113"

#define TOKEN "9121326820"

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;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient);

#include <SoftwareSerial.h>

SoftwareSerial mySerial(D6,D7);//rx,tx

int ldrMSB,ldrLSB,ldr;

void setup()

{

Serial.begin(9600);

Serial.println();

dht.begin();

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

mySerial.begin(9600);

pinMode(D4,OUTPUT);

}

void loop()

{

int moistmsb;

int moistlsb;

int moist;

if(mySerial.available()>21){

Serial.println("moisture data");

if(mySerial.read() == 0x7E){

for (int i=1;i<=20;i++) {

byte discardByte = mySerial.read();

// Serial.println(discardByte);

}

moistmsb=mySerial.read();

moistlsb=mySerial.read();

moist=moistlsb + (moistmsb \* 256);

Serial.println("moist");

Serial.println(moist);

//Serial.println("moisture");

delay(2000);

}

}

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t))

{

Serial.println("Failed to read from DHT sensor!");

delay(1000);

return;

}

PublishData(t,h,moist);

delay(1000);

if(moist>500)

{

digitalWrite(D4,HIGH);

}

else

{

digitalWrite(D4,LOW);

}

}

void PublishData(float temp, float humid, int moist){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"temperature\":";

payload += temp;

payload+="," "\"humidity\":";

payload += humid;

payload+="," "\"moisture\":";

payload += moist;

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");

}

}

**XCTU INSTALLATION:**

Download the XCTU software from the link below by choosing the bit size of the computer

<http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>

Connect the XBee Explorer USB board along with the XBee module to the computer.

Click on the "Discover devices" icon and select the port name to add the XBee modules in the XCTU software.

Now select one XBee module and go to radio configuration and change the ID field to e.g. "1001". These values must be the same to all xBee modules to communicate with each other. Let this XBee module be the transmitter and set the CE field as "Router". If the baud rate isn't set to 9600bps, change it to this value. Select the sampling rate as 1000m/s.

Now click the "Write" button to save the changes in the XBee module.

Disconnect the xBee explorer board from the computer and connect the other XBee module on it.

Connect the explorer board to the computer again and follow the same procedure but this time set the CE field as "Co-ordinator".

After making these changes, go to co-ordinator XBee module terminal and check for the starting analog value which is received from the router.