Environmental monitoring

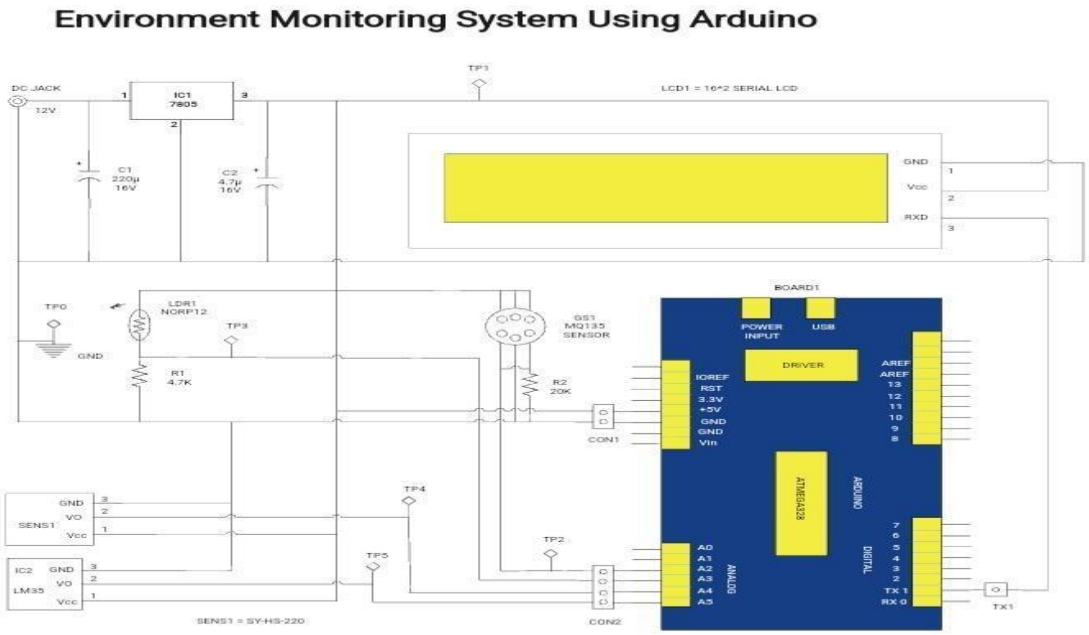
Developing of environmental monitoring :

Environmental monitoring solutions have evolved over the years into Smart Environmental Monitoring (SEM) systems that now incorporate modern sensors, Machine Learning (ML) techniques, and the Internet of Things (IoT). Technologies such as IoT devices and wireless sensor networks have made advanced environmental monitoring using IoT a more streamlined and Artificial Intelligence-controlled process.

Data captured by IoT environmental monitoring sensors from a wide variety of environmental conditions can be integrated via the Wireless Sensor Network (WSN) into one, cloud-based environmental system, in which IoT devices embedded with ML can record, characterize, monitor, and analyze elements in a specific environment.

IoT for environmental monitoring facilitates the development of wireless, remote environmental monitoring systems, which enable operations to remove much of the human interaction in system cfunction, which reduces human labor, increases the range and frequency of sampling and monitoring, facilitates husophisticated on-site testing, provides lower latency, and connects detection systems to response teams, ultimately resulting in higher rates of significant disaster and contamination prevention.

Circuit diagram:



In this circuit diagram Arduino UNO is connected to transmitter and receiver in connected to NodeMCU, then The data of the sensors which are collected from the sensors are transmitted from the pH sensor, soil moisture Sensor and Temperature sensor to connected NodeMCU.

Signal Conversion Module:
Supply Voltage: 3.3~5.5V• BNC Probe Connector• High Accuracy: ±0.1@25°C• Detection Range: 0~14

Coding:

```
# Arduino Code
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ArduinoJson.h>
OneWire oneWire(2);
DallasTemperaturetemp_sensor(&oneWire);
Float calibration_value = 21.34;
Int phval = 0;
Unsigned long int avgval;
Int buffer_arr[10], temp;
Void setup()
{
Serial.begin(9600);
Temp_sensor.begin();
}
StaticJsonBuffer<1000>jsonBuffer;
JsonObject&root=jsonBuffer.createObject();
Void loop()
{
For (int I = 0; I < 10; i++)
{
Buffer_arr[i] = analogRead(A0);
Delay(30);
}
For (int I = 0; I < 9; i++)
{
For (int j = I + 1; j < 10; j++)
{
If (buffer_arr[i] > buffer_arr[j])
{
Temp = buffer_arr[i];
Buffer_arr[i]=buffer_arr[j];
Buffer_arr[j] = temp;
```

```
}
}
}
Avgval = 0;
For (int I = 2; I < 8; i++)
Avgval += buffer_arr[i];
Float volt = (float)avgval * 5.0 / 1024 / 6;
Floatph_act=-5.70*volt+calibration_value;
Temp_sensor.requestTemperature();
Intmoisture_analog=analogRead(A1);
Intmoist_act=map(moisture_analog,0,1023,100,0);
Root[“a1”] = ph_act;
Root[“a2”]=temp_sensor.getTempByIndex(0);
Root[“a3”] = moist_act;
Root.printTo(Serial);
Serial.println(“”);
}
```

NodeMCU Code:

```
#include<ESP8266WiFi.h>
#include<WiFiClient.h>
#include<ESP8266WebServer.h>
#include <ArduinoJson.h>
Const char* ssid=“admin”;
//Replace with your network SSIDConst char* password=“12345678”;
//Replace with your network passwordESP8266WebServer server(80);
String page = “”;Int data1, data2, data3;
Void setup()
{
Serial.begin(9600);
WiFi.begin(ssid, password);
While (WiFi.status() != WL_CONNECTED)
{
Delay(500);Serial.print(“.”);
}
```

```
Serial.println(WiFi.localIP());
Server.on(“/”, []()
{
Page = “<html><head><title>IoT Design</title></head><style
type=\”text/css\”>”;
67Page += “table{border-collapse: collapse;}th {background-color: green
;color: white;
}
table,td {border: 4px solid black;
font-size: x-large;”;
Page += “text-align:center;border-style: groove;border-color: rgb(255,0,0);
}
</style><body><center>”;
Page += “<h1>Smart Aquaculture Monitoring using IoT</h1><br><br><table
style=\”width: 1200px;
height: 450px;\”><tr>”;
Page +=
“<th>Parameters</th><th>Value</th><th>Units</th></tr><tr><td>PH
Value</td><td>”+String(data1)+”</td><td>N/A</td></tr>”;
Page+=“<tr><td>Temperature</td><td>”+String(data2)+”</td><td>Centigra
de</td></tr><tr><td>Moisture</td><td>”+String(data3)+”</td><td>%</td>”;
Page += “<meta http-equiv=\”refresh\” content=\”3\”>”;
Server.send(200, “text/html”, page);});
Server.begin();}Void loop()
{
StaticJsonBuffer<1000> jsonBuffer;
JsonObject&root=jsonBuffer.parseObject(Serial);
If (root == JsonObject::invalid())
{
Return;
8Serial.println(“invalid”);
}
Data1 = root[“a1”];
Data2 = root[“a2”];
```

```
Data3 = root[“a3”];
Serial.println(data1);
Serial.println(data2);
Serial.println(data3);
Server.handleClient();
}
```

Coding definition:

First of all, include all the header files, which will be required throughout the code. Here we are using onwire.h and DallasTemperature.h library for a DS18B20 temperature sensor. This can be downloaded from the links given and included in the Arduino library. Similarly, we are using ArduinoJson.h library for sending JSON data from the transmitter to the receiver side.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ArduinoJson.h>
```

Next, define the connection pin of Arduino, where the output pin of the DS18B20 sensor will be connected, which is digital pin 2 in my case. Then, objects for onewire class and DallasTemperature classare defined which will be required in the coding for temperature measurement.

```
OneWire oneWire(2);

DallasTemperature temp_sensor(&oneWire);
```

Next, the calibration value is defined, which can be modified as required to get an accurate

pH value of solutions.

Float calibration_value = 21.34;

**Then a JSON Object is defined which will be required for sending parameters from the
Transmitter part to the Receiver part.**

StaticJsonBuffer<1000> jsonBuffer;

JsonObject& root = jsonBuffer.createObject();

**Inside loop(), read 10 sample Analog values and store them in an array. This is required to
smooth the output value.**

For(int i=0;i<10;i++)

{

Buffer_arr[i]=analogRead(A0);

Delay(30);

}

Then, we have to sort the Analog values received in ascending order. This is required because we need to calculate the running average of samples in the later stage.

For(int i=0;i<9;i++)

{

```
For(int j=i+1;j<10;j++)

{

If(buffer_arr[i]>buffer_arr[j])

{

Temp=buffer_arr[i];

Buffer_arr[i]=buffer_arr[j];

Buffer_arr[j]=temp;
}

}

}
```

Finally, calculate the average of a 6 centre sample Analog values. Then this average value is converted into actual pH value and stored in a variable.

```
For(int i=2;i<8;i++)

Avgval+=buffer_arr[i];

Float volt=(float)avgval*5.0/1024/6;

Float ph_act = -5.70 * volt + calibration_value;
```

To send a command to get the temperature values from the sensor, requestTemperatures() function is used.

```
Temp_sensor.requestTemperatures();
```

Now, analog values from the soil moisture sensor are read and this is mapped to percentage using map() function as shown below:

```
Int moisture_analog=analogRead(A1);

Int moist_act=map(moisture_analog,0,1023,100,0);
```

Finally, the parameters which are to be sent to NodeMCU, are inserted into JSON objects and they are sent via serial communication using root.printTo(Serial) command.

```
Root[“a1”] = ph_act;

Root[“a2”] = temp_sensor.getTempCByIndex(0);

Root[“a3”] = moist_act;

Root.printTo(Serial);

Serial.println(“”);
```

Programming NodeMCU:

After the successful completion of the hardware setup and Arduino programming, now it’s time to program the NodeMCU.

Note: Remove the Transmitter and Receiver connections between Arduino and NodeMCU before uploading the code.

To upload code in NodeMCU, follow the steps below:

1. Open Arduino IDE, then go to File–>Preferences–>Settings.
2.Typehttps://arduino.esp8266.com/stable/package_esp8266com_index.json in the 'Additional Board Manager URL' field and click 'Ok'.
3. Now go to Tools > Board > Boards Manager. In the Boards Manager window, Type
ESP8266 in the search box, select the latest version of the board, and click on install.
4. After installation is complete, go to Tools ->Board -> and select NodeMCU 1.0(ESP-
12E Module). Now you can program NodeMCU with Arduino IDE.

First, we have to include all the required library files in our code. ESP8266 library is included for both client and server configurations and Arduinojson.h for receiving the JSON data.

```
#include<ESP8266WiFi.h>

#include<WiFiClient.h>

#include<ESP8266WebServer.h>

#include <ArduinoJson.h>
```

Then create the ESP8266WebServer class object with the name server and default port number 80.

ESP8266WebServer server (80);

Now, declare network credentials i.e SSID and password. It is required to connect our NodeMCU to the internet.

Const char* ssid = “admin”;

Const char* password = “12345678”;

Then, to connect NodeMCU to the internet, call WiFi.begin and pass network SSID and password as its arguments. Check for the successful network connection using WiFi.status() and after a successful connection, print a message in Serial Monitor with IP address.

```
Serial.beg
in(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
 {
 delay(500);
 Serial.print(".");
 }
 Serial.print(WiFi.localIP());
```

In the next step, an HTML page for this Water Monitoring System using IOT is created as shown below, which has an HTML table to show pH value, Temperature, and soil moisture.

The HTML page is stored in a string variable so that it can be sent back on client request using the server.send() function.

```
page="<html><head><title>IoTDesignPro</title></head><styletype=\x/
css\">";
 page += "table{border-collapse: collapse;}
th {background-color: green ;color: white;table,td {border: 4px solid
black;font-size: x-large;";

 page += "text-align:center;border-style: groove;border-color:
rgb(255,0,0);}</style><body><
center>";

 page += "<h1>Smart Aquaculture Monitoring using
IoT</h1><br><br><table style=\"width: 1200p
x;height: 450px;\"><tr>";

 page +=
"<th>Parameters</th><th>Value</th><th>Units</th></tr><tr><td>PH
Value</td><td>"+Str
ing(data1)+"</td><td>N/A</td></tr>";

 page
+="<tr><td>Temperature</td><td>"+String(data2)+"</td><td>Centigra
de</td></tr><tr><td>
Moisture</td><td>"+String(data3)+"</td><td>%</td>";

 page += "<meta http-equiv=\"refresh\" content=\"3\">";
 server.send(200, "text/html", page);
```

Now, check for valid JSON data reception, from the Transmitter side. If no valid data is found, a message will be displayed on the serial monitor.

```
if (root == JsonObject::invalid())

{
 return;
 Serial.println("invalid");
 }
```

Otherwise, data received are assigned to individual variables andappended in the HTML webpage for real-time display.

```
root["a1"] = ph_act;

root["a2"] = temp_sensor.getTempCByIndex(0);

root["a3"] = moist_act;
```

At the end of the loop, to handle the client request, we have to call server.handleClient().

It will handle new requests and check them.

```
server.handleClient();
```

Below is the HTML page will be shown in a web browser:

Research	Purpose	Findings and Challenges	Method/Device Used
OEM [40]	Oceanic environment monitoring	Light weight; costly and invasive sensory networks	Wireless Sensors
IOT Based SM [46]	Soil monitoring for farming	Efficient vegetable crop monitoring; Greenhouse gases pose challenges on health of vegetables like tomato	Wireless sensors
IoT Protocols for MEM [42]	Marine environment acoustic monitoring	Lower latency; low power consumption; installation and coverage issues	WSN and IoT
IoT for air pollution [47]	Air pollution monitoring system	Mobile kit "IoT-Mobair" for prediction; inferior precision; low sensitivity; computationally complex	Gas sensor and IoT
[5]	Air quality monitoring	Scalable and high-density air quality monitoring with interconnection of heterogeneous sensors; computational complexity due to huge data captured and processed	Mobile sensor network and WSN
IoT based SEM [7]	Environmental monitoring	W3C standard for interoperability; interoperability issues of heterogeneous sensors	Heterogeneous sensors
Air quality [12]	Air quality monitoring	Large area monitoring; noisy data; accuracy and cost issues	Geomatics sensors and IoT
Pollution monitoring [16]	Air pollution monitoring System	Real time monitoring; accuracy issues	Sensors with MQ3 Model, Raspberry Pi and IoT
Sensor based AQM [37]	Air pollution monitoring system	Efficient for low coverage area; low cost; easy to install; less number of pollutants are covered	Gas sensor and LASER sensor
SEM [48]	Dust and humidity monitoring	Wide coverage and efficiency; low cost and small size	IoT
Radiation [36]	Radiation monitoring	High cost and low stability against temperature variation	HPXe chamber
Aqua farming and energy conservation [49]	Aqua Farming	Water quality and quantity control; higher carbon emission and energy requirement	Odor, pH, conductance and temperature sensor

Conclusion:

Environmental monitoring of air quality (1-hour TSP and 24-hour TSP) for the Project was performed in February 2003. All the monitoring results complied with the AL levels except four 24-hr TSP exceedances. Since all the 1-hr TSP monitoring results complied with the AL levels, it indicated an acceptable air quality during the operation hours of the Fill Bank. However, a poorer ambient air quality in the Fill Bank could be interpreted from the 24-hr TSP monitoring results. The Contractor was required to follow up all the mitigation measures as recommended in the EIA Report, EP and EM&A Manual. At this stage, provision of covers or hydroseeding on the exposed slopes and more frequent water spraying on the stockpiles and main haul roads are recommended.