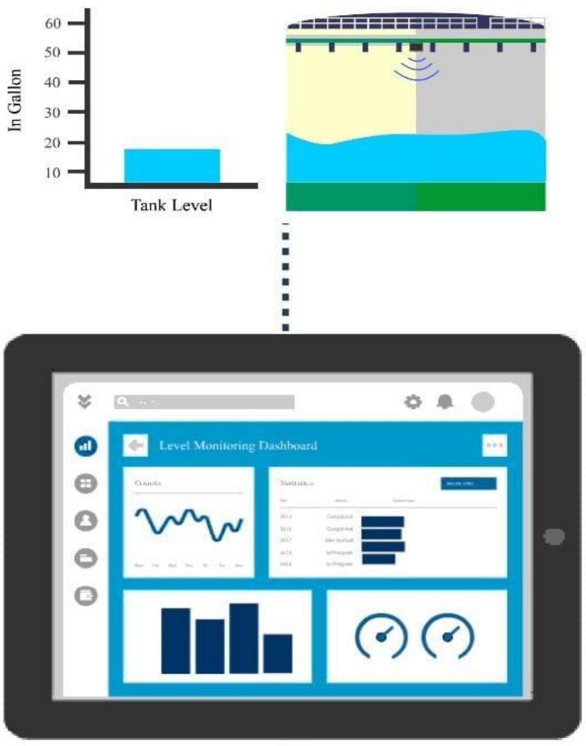
Smart Water Management using IoT

1.Objective:

The objective of this project is to create a Smart Water Management system that leverages Internet of Things (IoT) technology to monitor and manage water resources efficiently. The project aims to reduce water wastage, prevent leaks, and ensure the sustainable use of water in urban and rural environments.



IoT Device Setup:

The project involves the deployment of various IoT devices for data collection, control, and communication. Here are the key components of the IoT device setup:

Water Level Sensors:

These sensors are deployed in water storage tanks and reservoirs to monitor water levels and trigger refill requests when necessary.

IoT Gateway:

A central IoT gateway collects data from all the sensors, preprocesses the data, and sends it to a cloud-based platform for analysis and control.

2.Platform Development:

The project's platform development involves creating a cloud-based system for data analysis, visualization, and control. The platform consists of the following components:

Cloud Data Storage:

Data collected from the IoT devices is stored in a cloud database for real-time and historical analysis. Services like AWS, Azure, or Google Cloud can be used for this purpose.

Data Analysis and Prediction:

Machine learning models are developed to analyze the data and predict water quality, consumption trends, and leak detection. These models help in making informed decisions.

Control System:

The platform can send commands back to the IoT devices to control water flow, shut off supply in case of emergencies, or trigger maintenance alerts.

Dashboard:

User-friendly dashboards are created for water authorities, environmental agencies, and consumers to access real-time information about water quality, consumption, and alerts.

3.Code Implementation:

The code for this project will be written in various programming languages, depending on the component:

IoT Device Code:

Each IoT device has its own code to read sensor data and transmit it to the IoT gateway. This code may be written in languages like C, Python, or platforms like Arduino.

IoT Gateway Code:

The gateway has code to aggregate data, perform preprocessing, and securely transmit it to the cloud. Communication protocols like MQTT or HTTP may be used.

Cloud Data Storage Code:

Setting up and managing databases can be done using cloud providers' services and APIs.

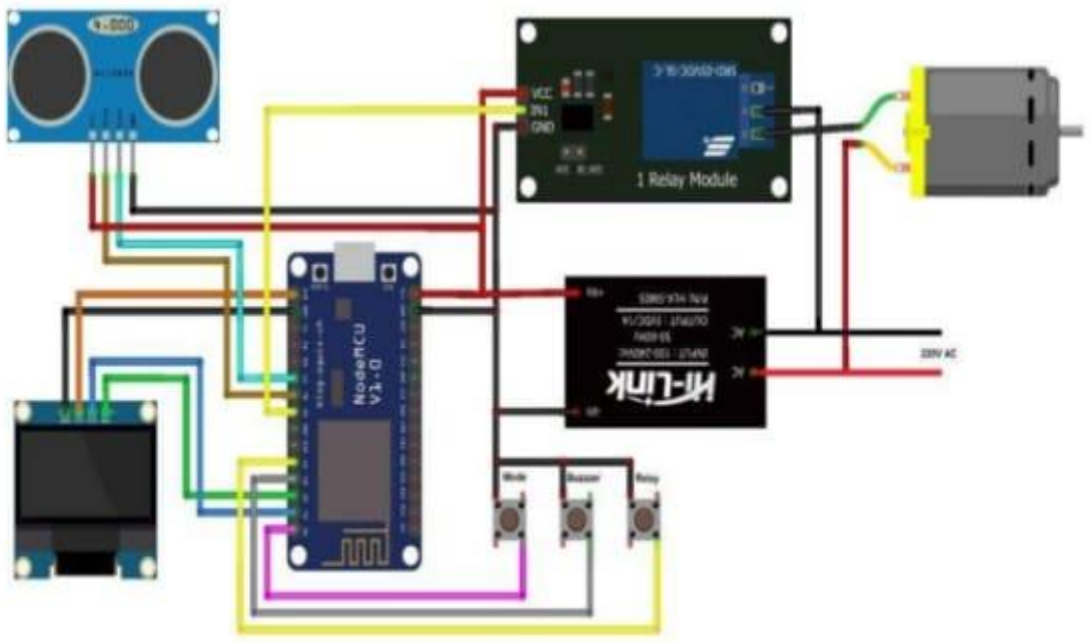
Data Analysis and Prediction Code:

Machine learning models are implemented using Python, along with libraries like TensorFlow, scikit-learn, or specific water quality analysis tools.

4.Explanation in Detail:

The project monitors water quality, flow rates, and levels to ensure efficient water management. Machine learning models help in early detection of water quality issues and leaks, allowing for preventive actions. Dashboards provide easy access to real-time data and alerts for decision-makers, enabling them to take timely actions to conserve water resources, prevent contamination, and ensure sustainable water usage.

This project's success contributes to better water quality, reduced water wastage, and more efficient water distribution, benefiting both the environment and the community.



Code for Running above Circuit:

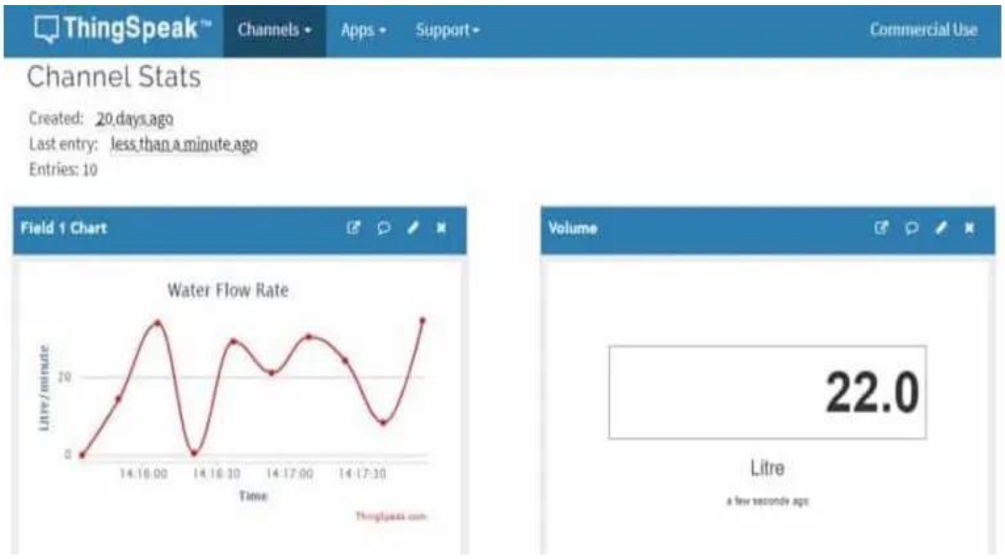
```
#include <Adafruit_SSD1306.h>

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>
```

```
#include <AceButton.h>

#define BLYNK_TEMPLATE_ID “*****”

#define BLYNK_TEMPLATE_NAME “********”

#define BLYNK_AUTH_TOKEN “*********”

Char ssid[] = “******”;

Char pass[] = “******”;

Int emptyTankDistance = 160;

Int fullTankDistance = 20;

Int triggerPer = 20;

Using namespace ace_button;

#define TRIG 12 //D6

#define ECHO 13 //D7

#define Relay 14 //D5

#define BP1 2 //D0

#define BP2 13 //D3

#define BP3 15 //D4

#define V_B_1 V1

#define V_B_3 V3

#define V_B_4 V4

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 32

#define OLED_RESET -1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

Float duration;

Float distance;

Int waterLevelPer;

Bool toggleRelay = false;

Bool modeFlag = true;

String currMode;
```

```
Char auth[] = BLYNK_AUTH_TOKEN;
ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);
ButtonConfig config3;
AceButton button3(&config3);
Void handleEvent1(AceButton*, uint8_t, uint8_t);
Void handleEvent2(AceButton*, uint8_t, uint8_t);
Void handleEvent3(AceButton*, uint8_t, uint8_t);
BlynkTimer timer;
Void checkBlynkStatus() {
 Bool isconnected = Blynk.connected();
 If (isconnected == false) {
 }
 If (isconnected == true) {
 }
}
BLYNK_WRITE(VPIN_BUTTON_3) {
 modeFlag = param.asInt();
 if (!modeFlag && toggleRelay) {
 digitalWrite(Relay, LOW);
 toggleRelay = false;
 }
 currMode = modeFlag ? "AUTO" : "MANUAL";
}
BLYNK_WRITE(VPIN_BUTTON_4) {
 If (!modeFlag) {
 toggleRelay = param.asInt();
 digitalWrite(Relay, toggleRelay);
 } else {
 Blynk.virtualWrite(V_B_4, toggleRelay);
 }
}
BLYNK_CONNECTED() {
 Blynk.syncVirtual(V_B_1);
 Blynk.virtualWrite(V_B_3, modeFlag);
 Blynk.virtualWrite(V_B_4, toggleRelay);
}
Void displayData() {
 Display.clearDisplay();
 Display.setTextSize(3);
 Display.setCursor(30, 0);
 Display.print(waterLevelPer);
 Display.print(“ “);
 Display.print(“%”);
 Display.setTextSize(1);
 Display.setCursor(20, 25);
 Display.print(currMode);
 Display.setCursor(95, 25);
 Display.print(toggleRelay ? “ON” : “OFF”);
 Display.display();
}
Void measureDistance() {
 digitalWrite(TRIG, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG, HIGH);
 delayMicroseconds(20);
 digitalWrite(TRIG, LOW);
 duration = pulseIn(ECHO, HIGH);
 distance = ((duration / 2) * 0.343) / 10;
 if (distance > (fullTankDistance – 15) && distance < emptyTankDistance) {
 waterLevelPer = map((int)distance, emptyTankDistance, fullTankDistance, 0, 100);
 Blynk.virtualWrite(V_B_1, waterLevelPer);
 If (waterLevelPer < triggerPer) {
 If (modeFlag) {
 If (!toggleRelay) {
 digitalWrite(Relay, HIGH);
 toggleRelay = true;
 Blynk.virtualWrite(V_B_4, toggleRelay);
 }
 }
 }
 If (distance < fullTankDistance) {
 If (modeFlag) {
 If (toggleRelay) {
 digitalWrite(Relay, LOW);
 toggleRelay = false;
 Blynk.virtualWrite(V_B_4, toggleRelay);
 }
 }
 }
 }
 displayData();
 delay(100);
}
Void setup() {
```

```
Serial.begin(9600);
pinMode(ECHO, INPUT);
pinMode(TRIG, OUTPUT);
pinMode(Relay, OUTPUT);
pinMode(BP1, INPUT_PULLUP);
pinMode(BP2, INPUT_PULLUP);
pinMode(BP3, INPUT_PULLUP);
digitalWrite(Relay, HIGH);
config1.setEventHandler(button1Handler);
config2.setEventHandler(button2Handler);
config3.setEventHandler(button3Handler);
button1.init(BP1);
button2.init(BP2);
button3.init(BP3);
currMode = modeFlag ? “AUTO” : “MANUAL”;
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
Serial.println(F(“SSD1306 allocation failed”));
For (;;)
;
}
Delay(1000);
Display.setTextSize(1);
Display.setTextColor(WHITE);
Display.clearDisplay();
WiFi.begin(ssid, pass);
Timer.setInterval(2000L, checkBlynkStatus);
Timer.setInterval(1000L, measureDistance);
Blynk.config(auth);
Delay(1000);
```

```
 Blynk.virtualWrite(V_B_3, modeFlag);
 Blynk.virtualWrite(V_B_4, toggleRelay);
 Delay(500);
}
Void loop() {
 Blynk.run();
 Timer.run();
 Button1.check();
 Button3.check();
 If (!modeFlag) {
 Button2.check();
 }
}
Void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
 Serial.println("EVENT1");
 Switch (eventType) {
 Case AceButton::kEventReleased:
 If (modeFlag && toggleRelay) {
 digitalWrite(Relay, LOW);
 toggleRelay = false;
 }
 modeFlag = !modeFlag;
 currMode = modeFlag ? "AUTO" : "MANUAL";
 Blynk.virtualWrite(V_B_3, modeFlag);
 Break;
 }
}
Void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
 Serial.println("EVENT2");
 Switch (eventType) {
 Case AceButton::kEventReleased:
 If (toggleRelay) {
 digitalWrite(Relay, LOW);
 toggleRelay = false;
 } else {
 digitalWrite(Relay, HIGH);
 toggleRelay = true;
 }
 Blynk.virtualWrite(V_B_4, toggleRelay);
 Delay(1000);
 Break;
 }
}
Void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
 Serial.println("EVENT3");
 Switch (eventType) {
 Case AceButton::kEventReleased:
 Break;
 }
}
```

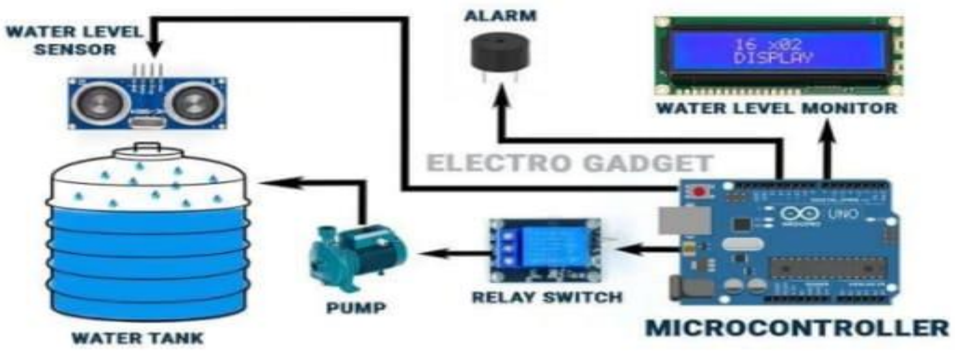


Channel Stats

Created: 20 days ago

Last entry: less than a minute ago

Entries: 10



CONCLUSION:

An internet-based approach on smart water level monitoring on a real-time basis. The results of the water level are verified that the system achieved the reliability and feasibility of using it for the actual monitoring purposes. The WSN network will be developed in the future comprising of more number of nodes to extend the coverage range. In our proposed system, water level can be monitored continuously from anywhere using web browser. Motor can be controlled automatically full smart automation is achieved. It is a robust system & small in size. This Project use ultrasonic sensors which provide more accurate and calibrated information for water level in tank.