SMART WATER MANAGEMENT

DEVELOPMENT PART 1

IOT PHASE 3

INSTRUCTIONS

- In this project, we will explore how to implement smart water detection over the WOWKI simulator, a versatile platform for simulating IoT environments.
- By combining the capabilities of IoT sensors, data analytics, and automation, we can proactively detect and respond to water-related issues, ensuring efficient water resource management and environmental conservation.
- This introduction sets the stage for a comprehensive exploration of the technology and its application in safeguarding our precious water resources.

SOFTWARE WEBSITE

- Wokwi

WOKWI:

- The WOWKI simulator is a versatile platform designed to simulate IoT (Internet of Things) environments for testing and development purposes.
- It provides a controlled and realistic virtual environment to mimic real-world scenarios and interactions that involve IoT devices and sensors.
- Here are some key points related to the WOWKI simulator and its use in

IOT APPLICATIONS :

1. IOT SIMULATION:

- WOWKI simulator is dedicated to simulating IoT devices, networks, and interactions.
- It allows developers and researchers to create, test, and analyze IoT applications without the need for physical devices or real-world environments.

- You can use it to show real-time measurements, alerts, or any relevant information from the monitoring system, making it easier for users or operators to access and understand the data.
- It provides a compact and clear display, which is particularly useful for quick on-site analysis and decision-making in water monitoring applications.

2.DEVICE EMULATION:

- WOWKI provides the capability to simulate real-world scenarios, such as smart cities, industrial automation, agriculture, and environmental monitoring.
- This enables the testing of IoT applications in contextually relevant environments..

3.REALISTIC SCENARIOS:

- WOWKI provides the capability to simulate real-world scenarios, such as smart cities, industrial automation, agriculture, and environmental monitoring.

- This enables the testing of IoT applications in contextually relevant environments.

4.DATA GENERATION:

- It can generate realistic data streams, which is crucial for testing data analytics, machine learning algorithms, and the responsiveness of IoT systems under different conditions.

5.SECURITY TESTING:

- Security is a critical aspect of IoT, and the WOWKI simulator allows for the testing of IoT security measures in a controlled environment. This helps identify vulnerabilities and develop strategies to protect IoT networks.
- Overall, the WOWKI simulator plays a significant role in accelerating the development and testing of IoT applications, ensuring they are robust, efficient, and ready for real-world deployment. It offers a valuable playground for IoT innovation and research.

PROGRAMMING LANGUAGE:

PYTHON:

- Python is a versatile and high-level programming language known for its simplicity and readability.
- It is widely used for web development, data analysis, artificial intelligence, and more.
- Python's clean syntax and extensive libraries make it a popular choice for both beginners and experienced developers.

PROGRAM :

Water level Detection Program

```
/* Fill-in your Template ID (only if using Blynk.Cloud) */
#define BLYNK_TEMPLATE_ID "TMPLIcLQu4bQ"
#define BLYNK_TEMPLATE_NAME "water monitor"
#define BLYNK_AUTH_TOKEN "OgvenxCWu9sG7-9deFGLFCLE4rWCGW7N"
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Wokwi-GUEST";   //WiFi Name
char pass[] = "";   //WiFi Password
//Set Water Level Distance in CM
int emptyTankDistance = 150 ;  //Distance when tank is empty
int fullTankDistance =  40 ;  //Distance when tank is full (must be greater than 25cm)
//Set trigger value in percentage
int triggerPer =   10 ;  //alarm/pump will start when water level drop below triggerPer
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <AceButton.h>
using namespace ace_button;
// Define connections to sensor
#define TRIGPIN    27  //D6
#define ECHOPIN    26  //D7
#define wifiLed    2  //D0
#define BuzzerPin  13  //D3
#define RelayPin   14 //D5
#define ButtonPin1 12   //RX   //Mode
#define ButtonPin2 33  //SD3  //Relay
#define ButtonPin3 32  //D4   //STOP Buzzer
#define fullpin    25
//Change the virtual pins according the rooms
#define VPIN_BUTTON_1    V1
#define VPIN_BUTTON_2    V2
#define VPIN_BUTTON_3    V3
#define VPIN_BUTTON_4    V4
#define VPIN_BUTTON_5    V5
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
```

```
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
float duration;
float distance;
int   waterLevelPer;
bool  toggleBuzzer = HIGH; //Define to remember the toggle state
bool toggleRelay = false; //Define the toggle state for relay
bool modeFlag = true;
bool conection = 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() { // called every 3 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    //Serial.println("Blynk Not Connected");
    digitalWrite(wifiLed, LOW);
    conection = true;

  }
  if (isconnected == true) {
    digitalWrite(wifiLed, HIGH);
    //Serial.println("Blynk Connected");
    conection = false;
  }
}
// When App button is pushed - switch the state
BLYNK_WRITE(VPIN_BUTTON_3) {
  modeFlag = param.asInt();
  if(!modeFlag && toggleRelay){
      digitalWrite(RelayPin, LOW);  //turn off the pump
```

```
      toggleRelay = false;
    }
    controlBuzzer(500);
    currMode = modeFlag ? "AUTO" : "MANUAL";
}
BLYNK_WRITE(VPIN_BUTTON_4) {
  if(!modeFlag){
    toggleRelay = param.asInt();
    digitalWrite(RelayPin, toggleRelay);
    controlBuzzer(500);
  }
  else{
    Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  }
}
BLYNK_WRITE(VPIN_BUTTON_5) {
  toggleBuzzer = param.asInt();
  digitalWrite(BuzzerPin, toggleBuzzer);
}
BLYNK_CONNECTED() {
  Blynk.syncVirtual(VPIN_BUTTON_1);
  Blynk.syncVirtual(VPIN_BUTTON_2);
  Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
  Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
}
void displayData(){
  display.clearDisplay();
  display.setTextSize(3);
  display.setCursor(30,0);
  display.print(waterLevelPer);
  display.print(" ");
  display.print("%");
  display.setTextSize(1);
  display.setCursor(0,25);
  display.print(conection ? "OFFLINE" : "ONLINE");
  display.setCursor(60,25);
  display.print(currMode);
  display.setCursor(110,25);
  display.print(toggleRelay ? "! ON" : "OFF");
  display.display();
}
```

```
void measureDistance(){
  // Set the trigger pin LOW for 2uS
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);
  // Set the trigger pin HIGH for 20us to send pulse
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(20);
  // Return the trigger pin to LOW
  digitalWrite(TRIGPIN, LOW);
  // Measure the width of the incoming pulse
  duration = pulseIn(ECHOPIN, HIGH);
  // Determine distance from duration
  // Use 343 metres per second as speed of sound
  // Divide by 1000 as we want millimeters
  distance = ((duration / 2) * 0.343)/10;
  if (distance > (fullTankDistance - 10)  && distance < emptyTankDistance ){
    waterLevelPer = map((int)distance ,emptyTankDistance, fullTankDistance, 0, 100);
    Blynk.virtualWrite(VPIN_BUTTON_1, waterLevelPer);
    Blynk.virtualWrite(VPIN_BUTTON_2, (String(distance) + " cm"));
    // Print result to serial monitor
//    Serial.print("Distance: ");
//    Serial.print(distance);
//    Serial.println(" cm");
    if (waterLevelPer < triggerPer){
      if(modeFlag){
        if(!toggleRelay){
          controlBuzzer(500);
          digitalWrite(RelayPin, HIGH); //turn on relay
          toggleRelay = true;
          Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
        }
      }
      else{
        if (toggleBuzzer == HIGH){
          digitalWrite(BuzzerPin, HIGH);
          Serial.println(" BuzzerPin high");
        }
      }
    }
    if (distance < fullTankDistance){
      digitalWrite(fullpin, HIGH);
      if(modeFlag){
```

```
        if(toggleRelay){
          digitalWrite(RelayPin, LOW); //turn off relay
          toggleRelay = false;
          Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
          controlBuzzer(500);
        }
      }
      else{
        if (toggleBuzzer == HIGH){
        digitalWrite(BuzzerPin, HIGH);
        }
      }
    }
    if (distance > (fullTankDistance + 5) && waterLevelPer > (triggerPer + 5)){
      toggleBuzzer = HIGH;
      Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
      digitalWrite(BuzzerPin, LOW);
    }
    if (distance = fullTankDistance){
    Serial.println(" udh bang ");
    }
  }
  displayData();
  delay(100);
}
void controlBuzzer(int duration){
  digitalWrite(BuzzerPin, HIGH);
  Serial.println(" BuzzerPin HIT");
  delay(duration);
  digitalWrite(BuzzerPin, LOW);
}
void setup() {
  // Set up serial monitor
  Serial.begin(9600);
  // Set pinmodes for sensor connections
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(wifiLed, OUTPUT);
  pinMode(RelayPin, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);
  pinMode(fullpin, OUTPUT);
  pinMode(ButtonPin1, INPUT_PULLUP);
```

```
  pinMode(ButtonPin2, INPUT_PULLUP);
  pinMode(ButtonPin3, INPUT_PULLUP);
  digitalWrite(wifiLed, HIGH);
  digitalWrite(RelayPin, LOW);
  digitalWrite(BuzzerPin, LOW);
  config1.setEventHandler(button1Handler);
  config2.setEventHandler(button2Handler);
  config3.setEventHandler(button3Handler);
  button1.init(ButtonPin1);
  button2.init(ButtonPin2);
  button3.init(ButtonPin3);
  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); // check if Blynk server is connected every 2
seconds
  timer.setInterval(1000L,  measureDistance); // measure water level every 1 seconds
  Blynk.config(auth);
  delay(1000);
  Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
  Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
  delay(500);
}
void loop() {
  Blynk.run();
  timer.run(); // Initiates SimpleTimer
  button1.check(); //mode change
  button3.check(); //buzzer reset
  if(!modeFlag){  //if in manual mode
    button2.check();
  }
}
void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT1");
```

```
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      if(modeFlag && toggleRelay){
        digitalWrite(RelayPin, LOW);  //turn off the pump
        toggleRelay = false;
        controlBuzzer(500);
      }
      modeFlag = !modeFlag;
      currMode = modeFlag ? "AUTO" : "MANUAL";
      Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
      controlBuzzer(200);
      break;
  }
}
void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT2");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      if(toggleRelay){
        digitalWrite(RelayPin, LOW);  //turn off the pump
        toggleRelay = false;
      }
      else{
        digitalWrite(RelayPin, HIGH);  //turn on the pump
        toggleRelay = true;
      }
      Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
      controlBuzzer(500);
      delay(1000);
      break;
  }
}
void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT3");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      digitalWrite(BuzzerPin, LOW);
      toggleBuzzer = LOW;
      Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
```

```
    break;
  }
}
```

CONCLUSION:

- In conclusion, the utilization of the WOKWI simulator for smart water management presents a promising approach to safeguarding this invaluable resource.
- By harnessing the power of IoT technology within this virtual environment, we can proactively monitor, analyze, and optimize water-related processes.
- This innovation not only aids in the efficient use of water resources but also contributes to environmental conservation and sustainability.
- As we continue to face challenges related to water scarcity and quality, the WOKWI simulator offers a practical and effective means to develop, test, and implement smart solutions that can have a lasting impact on water management practices.
- In upcoming phases, we will discuss about the output for this program and circuit for water level detection

THANK YOU