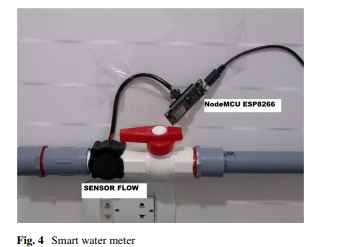
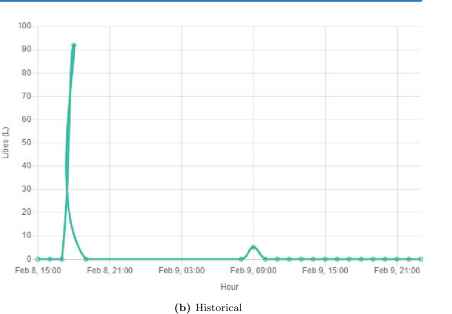
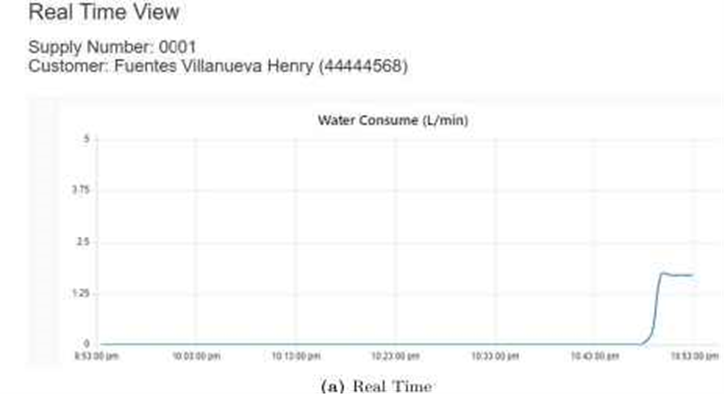
Smart Water System

**Indruction**:

A smart water system refers to a technology-driven approach to managing and optimizing water resources, consumption, and distribution. These systems leverage various sensors, data analytics, and automation to monitor, control, and improve the efficiency and sustainability of water-related processes. Here are some key components and features of a smart water system:

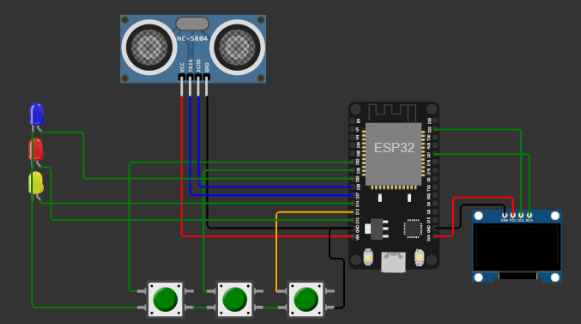
1. **Sensor Network:** Smart water systems often deploy a network of sensors throughout the water infrastructure. These sensors can monitor various parameters such as water quality, water flow, pressure, and even the condition of pipes and equipment.
2. **Data Collection:** The data collected by the sensors is transmitted in real-time to a central control system. This data helps in monitoring the current state of the water system and identifying potential issues.
3. **Data Analytics:** Advanced data analytics and algorithms are used to process the information from the sensors. This allows for the detection of anomalies, leakages, and other problems, as well as predictive maintenance to prevent failures.
4. **Remote Monitoring:** Smart water systems often provide the capability for remote monitoring, allowing operators to access real-time data and control systems remotely through digital interfaces.
5. **Water Quality Control:** By continuously monitoring water quality, smart water systems can help ensure that the water supplied to consumers meets safety and quality standards. 
6. **Leak Detection:** Smart water systems can detect leaks in the distribution network early on, minimizing water loss and preventing damage to infrastructure.
7. **Energy Efficiency:** These systems can optimize the energy consumption of water pumps and treatment facilities, reducing operational costs and environmental impact.
8. **Customer Engagement:** Some smart water systems provide consumers with tools to monitor their water usage, helping them to reduce consumption and save money.
9. **Asset Management:** Smart water systems assist in the maintenance and management of water infrastructure assets, extending their lifespan and reducing repair costs.



1. **Integration with IoT:** Integration with the Internet of Things (IoT) allows for more comprehensive control and automation of water processes. For example, adjusting water treatment processes based on real-time data.
2. **Environmental Sustainability:** By reducing water waste and energy consumption, smart water systems contribute to environmental sustainability and the conservation of water resources.
3. **Compliance and Reporting:** These systems can generate reports and alerts to help ensure compliance with water quality and environmental regulations.

Smart water systems are crucial for urban planning, water utilities, and industries where efficient water management is essential. They play a significant role in addressing the growing challenges of water scarcity, aging infrastructure, and the need for sustainability in water resource management.

\*  TITLE: IoT-based Water Level Indicator using NodeMCU, Ultrasonic Sensor & Blynk with 0.96" OLED



 \*  Click on the following links to learn more.

 \*  YouTube Video: https://youtu.be/NHxnWQF2504

 \*  Related Blog : https://iotcircuithub.com/esp8266-projects/

 \*

 \*  This code is provided free for project purpose and fair use only.

 \*  Please do mail us to techstudycell@gmail.com if you want to use it commercially.

 \*  Copyrighted © by Tech StudyCell

 \*

 \*  Preferences--> Aditional boards Manager URLs :

 \*  https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_dev\_index.json, http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

 \*

 \*  Download Board ESP8266 (3.1.1) : https://github.com/espressif/arduino-esp32

 \*

 \*  Download the libraries

 \*  Blynk Library (1.1.0):  https://github.com/blynkkk/blynk-library

 \*  Adafruit\_SSD1306 Library (2.5.7): https://github.com/adafruit/Adafruit\_SSD1306

 \*  AceButton Library (1.9.2): https://github.com/bxparks/AceButton

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Fill-in your Template ID (only if using Blynk.Cloud) \*/

#define BLYNK\_TEMPLATE\_ID "TMPLlcLQu4bQ"

#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;

  }

}