Statement of Purpose

Project-Based Submission

Project Title: Smart Water Management System

Applicant: Doddipatla Siddeswar

Introduction

- I am a passionate electronics and communication engineering student with a deep interest in embedded systems. From my childhood, I started doing projects that brought me to this stage. When I saw many government hostels and schools with improper maintenance or improper water management systems leading to a huge amount of water wastage, which leads to water scarcity, I was inspired. Water is the most important natural resource, so we have to use it properly. That inspired me to make this project and provide it to everyone at an affordable cost. I started to build a smart system to monitor and manage water tank levels efficiently using IoT. This project not only highlights my technical skills but also reflects my commitment to solving real-world environmental issues.

- Water over flow leads to:

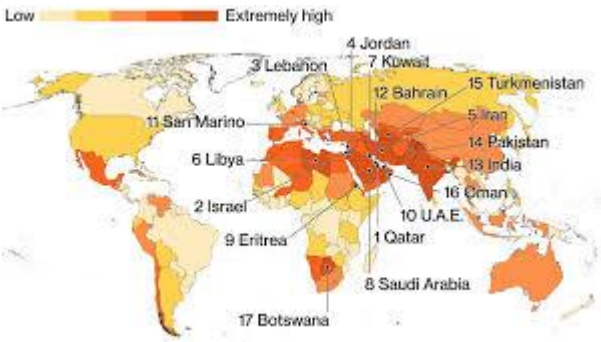
The issue of a scarcity.

The construction damage due water over flow.

The formation of diseases like malaria dengue is occurred due water stagnant.

Problem Statement

Water is the most important resource. Many countries are facing a water scarcity problem, including Qatar, Israel, Lebanon, Iran, Jordan, Libya, Kuwait, Saudi Arabia, Eritrea, the United Arab Emirates, San Marino, Bahrain, India, Pakistan, Turkmenistan, Oman, and Botswana. I want to save those countries from that situation.



Objective

- The goal of this project is to create a low-cost, real-time water monitoring system using microcontroller boards and sensors. The system alerts users through audio, displays the water level on a screen, and automatically controls the motor to prevent overflow or dry running.
- It monitors the amount of water consumed and the flow rate, which helps us know if there is a water leakage or not.
- We receive every single piece of information on our display, including water temperature; the water level is displayed with the help of LEDs, and the percentage announcement helps us always know the water level.
- It is fully automated, so there is no human error or water wastage issue. My project is compactable and less power consumption and affordable cost.

Technology Used:

- **Water level and pump control**

1. Arduino Uno

Arduino Uno

- **Type**: ATmega328P microcontroller board
- **Purpose**: Central controller that:
  - Reads water level via digital pins
  - Measures temperature using analog input
  - Drives the OLED display via I2C
  - Controls a relay for pump operation
  - Plays audio using DFPlayer Mini via Software Serial
- **Pins Used**:
  - **D2–D6**: Water level sensors
  - **D7–D11**: LEDs for level indication
  - **A0**: Analog input from thermistor
  - **A1**: Relay control
  - **D12,D13**: Serial connection to DFPlayer Mini

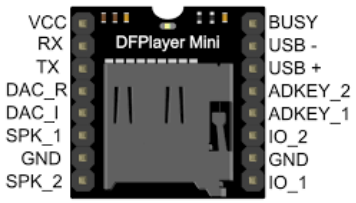
2. OLED Display (SSD1306, 128x64, I2C)

- **Pins**:
  - **SDA - A4**
  - **SCL - A5**
  - **VCC- 5V**
  - **GND-GND**

I2C OLED Display (SSD1306)

- **Purpose**: Graphically displays:
  - Water level and percentage
  - Water tank fill graphic
  - Pump ON/OFF status
  - Temperature (°C) with thermometer icon
  - High temperature warnings (above 45°C)

3. DFPlayer Mini MP3 Module



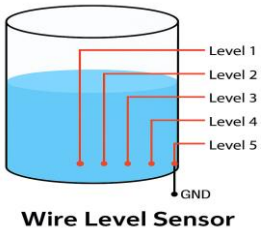
DFPlayer Mini with speaker

- **Pins**:
  - **RX - Arduino Pin 12**
  - **TX - Arduino Pin 13**
- **Purpose**:
  - Plays voice/audio alerts stored on microSD card:
    - Level 1 to Level 5 (e.g., "Level 1 detected")
    - Temperature High Warning (e.g., "Temperature too high!")
- **File Names on SD Card**: 001.mp3 to 006.mp3

4. 8Ω 1W Speaker

- **Connection**: To DFPlayer's **SPK_1 and SPK_2** pins
- **Purpose**: Plays audio alerts loudly enough for nearby users

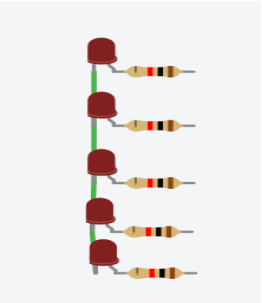
5. Water Level Detection Wires (x5)



Wire Level Sensor

- **Pins Used: D2, D3, D4, D5, D6**
- **Connection:**
  - **(One wire at bottom) connected to GND inside the tank**
  - **Five wires placed at increasing heights in tank connected to pins D2–D6**
- **Logic: When water touches a wire, circuit is completed to GND- Pin reads LOW**

6. LEDs (5x for levels)



- **Pins Used**: D**7, D8, D9,D10,D11**
- **Purpose**:
    - Indicate active water level visually
    - Only one LED is ON at a time, corresponding to current detected level

7. Relay Module (Pump Control)

- **Pin Used**: **A1**
- **Purpose**:
    - Turns ON pump when level is above Level 1
    - Turns OFF pump when tank is nearly empty
    - **Logic**: digitalWrite(A1, HIGH) → pump ON
    - digitalWrite(A1, LOW) → pump OFF

8. HT-NTC100K Thermistor (350°C)

- **Pin Used**: **A0 (Analog Input)**
- **Connection**:
    - One side - **5V**
    - Other side - **A0** with **10KΩ resistor** to **GND**
- **Purpose**:
    - Measures water temperature using voltage divider
    - Code calculates Celsius temperature using Steinhart-Hart formula
    - Triggers audio and OLED warning if temp > 45°C

NOTE:

Add 10k resistor acts as a pull-down resistor in the **NTC thermistor voltage divider** -Converts resistance to measurable voltage

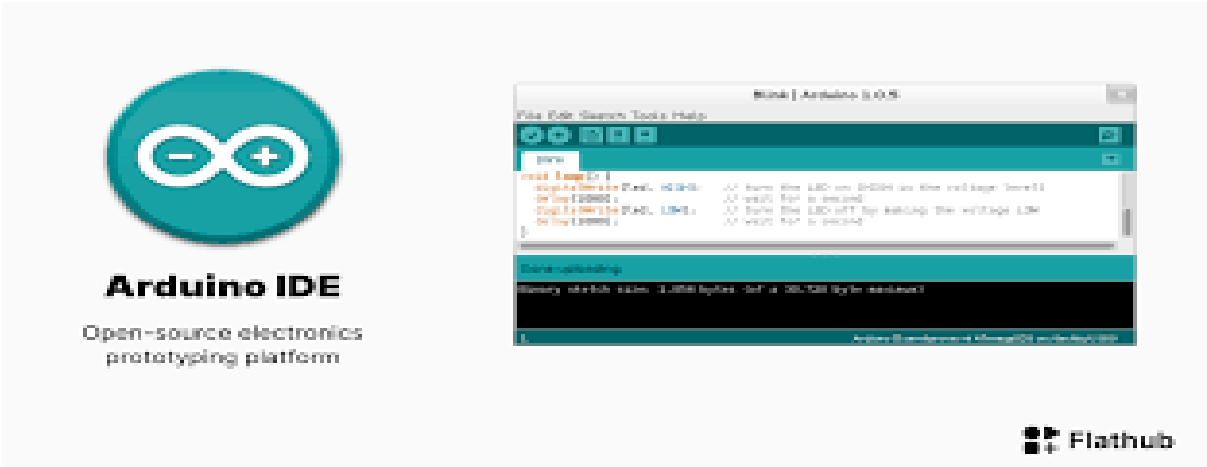
SOFTWARE CODE FOR WATER LEVEL:

```
 #include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SoftwareSerial.h>
#include <DFRobotDFPlayerMini.h>
#include <math.h>

// OLED setup
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Water level input pins
const int levelPins[5] = {2, 3, 4, 5, 6};
bool levelStatus[5] = {false};
const int ledPins[5] = {7, 8, 9, 10, 11};

// Relay pin
const int relayPin = A1;

// DFPlayer Mini
SoftwareSerial mySerial(12, 13); // RX, TX
DFRobotDFPlayerMini dfplayer;

// Temperature sensor
const int tempPin = A0;

// State tracking
int currentLevel = -1;
int lastLevel = -1;
bool pumpOn = false;
unsigned long lastTempPlayTime = 0;
const unsigned long tempPlayInterval = 60000;

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  // OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("OLED not found"));
    while (1);
  }
  display.clearDisplay();
  display.display();

  // Pins
  for (int i = 0; i < 5; i++) {
    pinMode(levelPins[i], INPUT_PULLUP);
    pinMode(ledPins[i], OUTPUT);
  }
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);

  // DFPlayer
  if (!dfplayer.begin(mySerial)) {
    Serial.println(F("DFPlayer not found"));
  } else {
    dfplayer.volume(25);
  }
}

void loop() {
  // Read level
  for (int i = 0; i < 5; i++) {
    levelStatus[i] = !digitalRead(levelPins[i]);
  }
  if (levelStatus[0]) currentLevel = 1;
  else if (levelStatus[1]) currentLevel = 2;
  else if (levelStatus[2]) currentLevel = 3;
  else if (levelStatus[3]) currentLevel = 4;
  else if (levelStatus[4]) currentLevel = 5;
  else currentLevel = -1;
```

```
// LEDs
for (int i = 0; i < 5; i++) {
  digitalWrite(ledPins[i], (i == currentLevel - 1) ? HIGH : LOW);
}

// Relay logic
if (currentLevel != -1 && currentLevel > 1) {
  digitalWrite(relayPin, HIGH);
  pumpOn = true;
} else {
  digitalWrite(relayPin, LOW);
  pumpOn = false;
}

// DFPlayer level audio (play only on change)
if (currentLevel != lastLevel && currentLevel != -1) {
  dfplayer.play(currentLevel); // 1 to 5 mapped to levels
  lastLevel = currentLevel;
}

// Temperature
int raw = analogRead(tempPin);
float voltage = raw * (5.0 / 1023.0);
float resistance = (5.0 - voltage) * 10000 / voltage;
float temperature = 1.0 / (log(resistance / 100000.0) / 3950.0 + 1.0 / 298.15) - 273.15;

if (temperature >= 45.0 && millis() - lastTempPlayTime > tempPlayInterval) {
  dfplayer.play(6); // High temp alert
  lastTempPlayTime = millis();
}

// OLED Display
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(40, 0);
display.print("Water Level: ");
display.println(currentLevel == -1 ? 0 : currentLevel);
display.setCursor(40, 10);
display.print("Level ");
display.print(currentLevel == -1 ? 0 : currentLevel);
display.println(" Detected");

display.setCursor(40, 58);
display.print("Temp: ");
display.print(temperature, 1);
display.print(" C");

// Tank Display
int tankX = 10, tankY = 8, tankWidth = 40, tankHeight = 50;
display.drawRoundRect(tankX, tankY, tankWidth, tankHeight, 6, SSD1306_WHITE);
display.drawRect(tankX + (tankWidth - 12) / 2, tankY - 8, 12, 8, SSD1306_WHITE);

int fillHeight = 0;
if (currentLevel != -1)
  fillHeight = (tankHeight / 5) * (6 - currentLevel);

if (fillHeight > 0) {
  int fillX = tankX + 3;
  int fillY = tankY + tankHeight - fillHeight;
  int fillW = tankWidth - 6;
  int fillAdj = fillHeight - 2;
  if (fillAdj < 0) fillAdj = 0;
  display.fillRoundRect(fillX, fillY, fillW, fillAdj, 4, SSD1306_WHITE);
  display.drawLine(fillX, fillY, fillX + fillW, fillY, SSD1306_BLACK);
}

// Percentage text
int percent = currentLevel == -1 ? 0 : 100 - (currentLevel - 1) * 25;
String percentText = String(percent) + "%";
display.setTextSize(2);
int16_t x1, y1;
uint16_t w, h;
display.getTextBounds(percentText, 0, 0, &x1, &y1, &w, &h);
```

```
  int textY = tankY + tankHeight - (tankHeight / 5) * (6 - currentLevel) + ((tankHeight / 5 - h) / 2);
  int textX = tankX + (tankWidth - w) / 2;
  bool insideFill = currentLevel != -1 && (textY + h > tankY + tankHeight - fillHeight);
  display.setTextColor(insideFill ? SSD1306_BLACK : SSD1306_WHITE);
  display.setCursor(textX, textY);
  display.print(percentText);
  display.setTextColor(SSD1306_WHITE);

  // Pump status
  if (pumpOn) {
    display.setTextSize(2);
    display.setCursor(40, 25);
    display.setTextColor(SSD1306_WHITE, SSD1306_BLACK);
    display.println("PUMP ON");
  }

  // High temp warning
  if (temperature >= 45.0) {
    display.setTextSize(1);
    display.setCursor(40, 40);
    display.println("!!! TEMP HIGH !!!");
    display.fillTriangle(10, 50, 20, 30, 30, 50, SSD1306_WHITE);
  }

  // Thermometer icon
  int thermoX = 90, thermoY = 20, thermoWidth = 20, thermoHeight = 40;
  display.drawCircle(thermoX + thermoWidth / 2, thermoY + thermoHeight, 8, SSD1306_WHITE);
  display.drawRoundRect(thermoX + thermoWidth/2 - 5, thermoY, 10, thermoHeight, 5, SSD1306_WHITE);

  float tempBar = constrain(temperature, 0, 60);
  fillHeight = map(tempBar, 0, 60, 0, thermoHeight - 4);
  display.fillCircle(thermoX + thermoWidth / 2, thermoY + thermoHeight, 7, SSD1306_WHITE);
  display.fillRoundRect(thermoX + thermoWidth/2 - 4, thermoY + thermoHeight - fillHeight, 8, fillHeight, 4, SSD1306_WHITE);

  // Temp number
  display.setTextSize(2);
  display.setCursor(thermoX - 20, thermoY + thermoHeight + 4);
  display.print(temperature, 1);
  display.print(" C");

  if (temperature >= 45.0) {
    display.setTextSize(1);
    display.setCursor(thermoX - 30, thermoY + thermoHeight + 24);
    display.println("!!! TEMP HIGH !!!");
    display.fillTriangle(thermoX - 10, thermoY + thermoHeight + 18,
                         thermoX, thermoY + thermoHeight + 38,
                         thermoX + 10, thermoY + thermoHeight + 18, SSD1306_WHITE);
  }

  display.display();
  delay(500); // smooth refresh
}
```

Use Arduino ide software coding the Arduino uno



- **Water flow monitoring**

1. ESP8266 Wi-Fi Development Board

- **Function: Acts as the central microcontroller. It reads data from the water flow sensor, processes it, updates the LCD, connects to Wi-Fi, and uploads data to the Blynk IoT cloud.**
- **Why Chosen: It is compact, affordable, and comes with built-in Wi-Fi, making it ideal for IoT applications.**
- **Technical Features:**
  - **Based on the ESP8266 chip**
  - **80/160 MHz processor**
  - **4 MB Flash memory**
  - **GPIO support and hardware interrupt capabilities**

ESP82

- **Connection: Connected to the flow sensor (D5), I2C LCD (D1, D2), and powered via micro-USB (5V).**

2. YF-S201 Hall Effect Water Flow Sensor

- **Function: Measures the flow rate and total water usage by outputting pulses proportional to the flow of water.**
- **Why Chosen: Accurate and cost-effective for water flow monitoring, commonly used in residential IoT systems.**
- **Technical Features:**
  - **1–30 L/min flow range**
  - **Operating voltage: 5–18V**
  - **Pulse Frequency: (7.5 × Flow Rate in L/min)**

- **Connection:**
  - **VCC to 5V**
  - **GND to GND**
  - **OUT to digital pin D5 (interrupt pin on NodeMCU)**

3. 16x2 I2C LCD Display Module (LiquidCrystal_I2C)

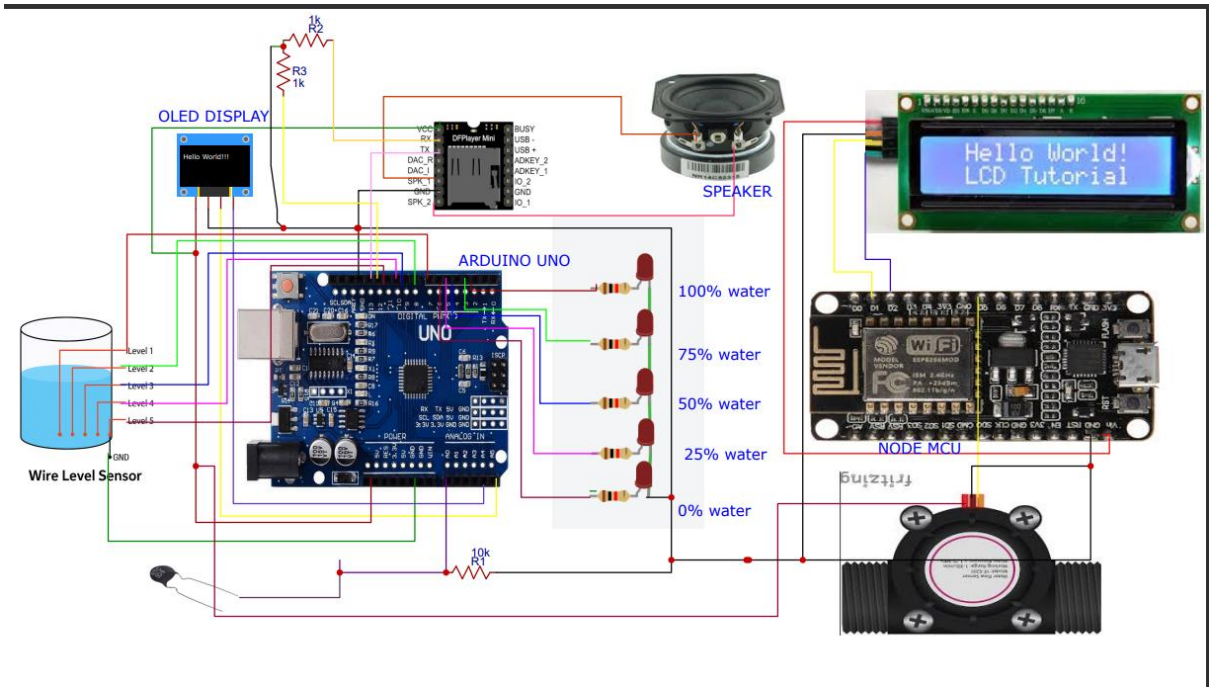
- **Function**: Displays real-time flow rate, total daily water usage, saved water, and current time/date.
- **Why Chosen**: I2C reduces the number of pins used, allowing for simple two-wire communication with ESP8266.
- **Technical Features**:
    - 16 characters × 2 lines
    - I2C interface (uses only SDA and SCL lines)
    - Backlight with adjustable contrast
- **Connection**:
    - VCC to 3.3V or 5V
    - GND to GND
    - SDA to D2 (GPIO4)
    - SCL to D1 (GPIO5)

SOFTWARE CODE FOR WATER FLOW:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define FLOW_SENSOR_PIN D5  // GPIO14

volatile int flowPulseCount = 0;
unsigned long lastMillis = 0;
float flowRate = 0.0;
float totalLiters = 0.0;

// Create LCD: If your scanner showed 0x27, keep this. If 0x3F, change it.
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Interrupt service routine for flow sensor
void IRAM_ATTR flowISR() {
  flowPulseCount++;
}

void setup() {
  Serial.begin(115200);

  // I2C LCD setup on NodeMCU (D1=SDA, D2=SCL)
  Wire.begin(D1, D2);
  lcd.begin(16, 2);
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Water Monitor");

  // Flow sensor setup
  pinMode(FLOW_SENSOR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), flowISR, RISING);
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - lastMillis >= 1000) {  // every 1 second
    detachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN));

    // Calculate flow rate in L/min
    flowRate = (flowPulseCount / 7.5);  // For YF-S201: 7.5 pulses = 1 L/min
    float litersPerSecond = flowRate / 60.0;
    totalLiters += litersPerSecond;

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Flow: ");
    lcd.print(flowRate, 1);
    lcd.print(" L/m");

    lcd.setCursor(0, 1);
    lcd.print("Total: ");
    lcd.print(totalLiters, 2);
    lcd.print(" L");

    Serial.print("Flow: ");
    Serial.print(flowRate);
    Serial.print(" L/m, Total: ");
    Serial.println(totalLiters);

    flowPulseCount = 0;
    lastMillis = currentMillis;

    attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), flowISR, RISING);
  }
}
```

CIRCUIT DIGRAM:



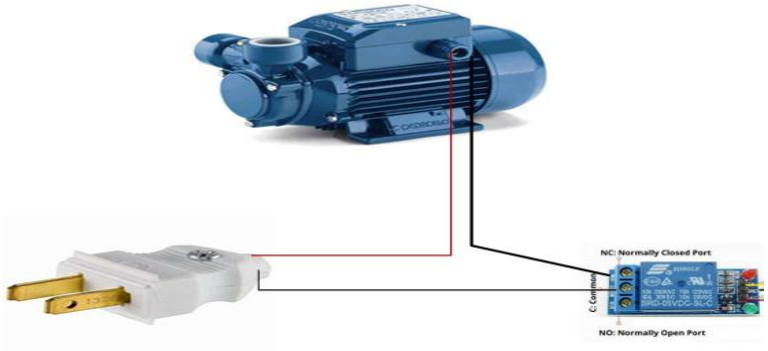
INSTALLATION STEPS:

STEP 1:

- When we provide a constant USB power supply of 5V to our compactwatermanagement box and connect all the sensors to the box or system with the help of the port at the side of the box.

STEP 2:

- Connect relay connection com, no and to the pump.



STEP 3:

- Hang the system to your convenient place and observe readings in display.

Methodology

The water level is detected using multiple sensors at different tank heights. The readings are processed by a microcontroller (Arduino/ESP32), which drives an OLED display for visual feedback and activates DFPlayer Mini for audio alerts. The system can be expanded with IoT connectivity using Blynk for remote monitoring and pump control. A relay module is used to automate pump switching

Expected Outcome

- The project is expected to reduce water wastage, alert users when the tank is full or empty, and automate motor control.
- Saving water resource.
- Protecting plants and animals from droughts.
- Green and clean disease-free country.
- We can monitor our daily usage of water which helps us to save water and we can also detect the leaks in our house, collage, apartment, and many areas.

Application in Germany

As a German applicant, I am very interested in researching at universities like RWTH Aachen, TU Chemnitz, TU Munich, etc. I want to pursue a Master's in Embedded Systems; I think my project is closely related. Through this project, I want to bring about change in every single family or community to monitor and save water regularly. Germany is a country with advanced technology. I want to seek knowledge with the help of research universities and develop more such projects to assist in the development of the country.

Skills Gained

- This project helped me enhance my circuit design, coding (C/C++), IoT integration, and troubleshooting skills. I also learned the importance of user-friendly design and energy efficiency in embedded systems.
- I learned how to innovate the idea into a real-time project.
- I have learned the calibration and integration of IoT sensors.
- Now I am capable of performing any activity with the respective IoT sensors.

Conclusion

My Smart Water Level Monitoring System represents my dedication to applying engineering skills for environmental protection and safety. This project lays a strong foundation for my graduate studies in embedded systems and IoT automation. I am eager to continue this journey in Germany by studying for my MS in embedded systems and continuing to pursue a PhD, a country known for its leadership in engineering and environmental consciousness and care among international students.