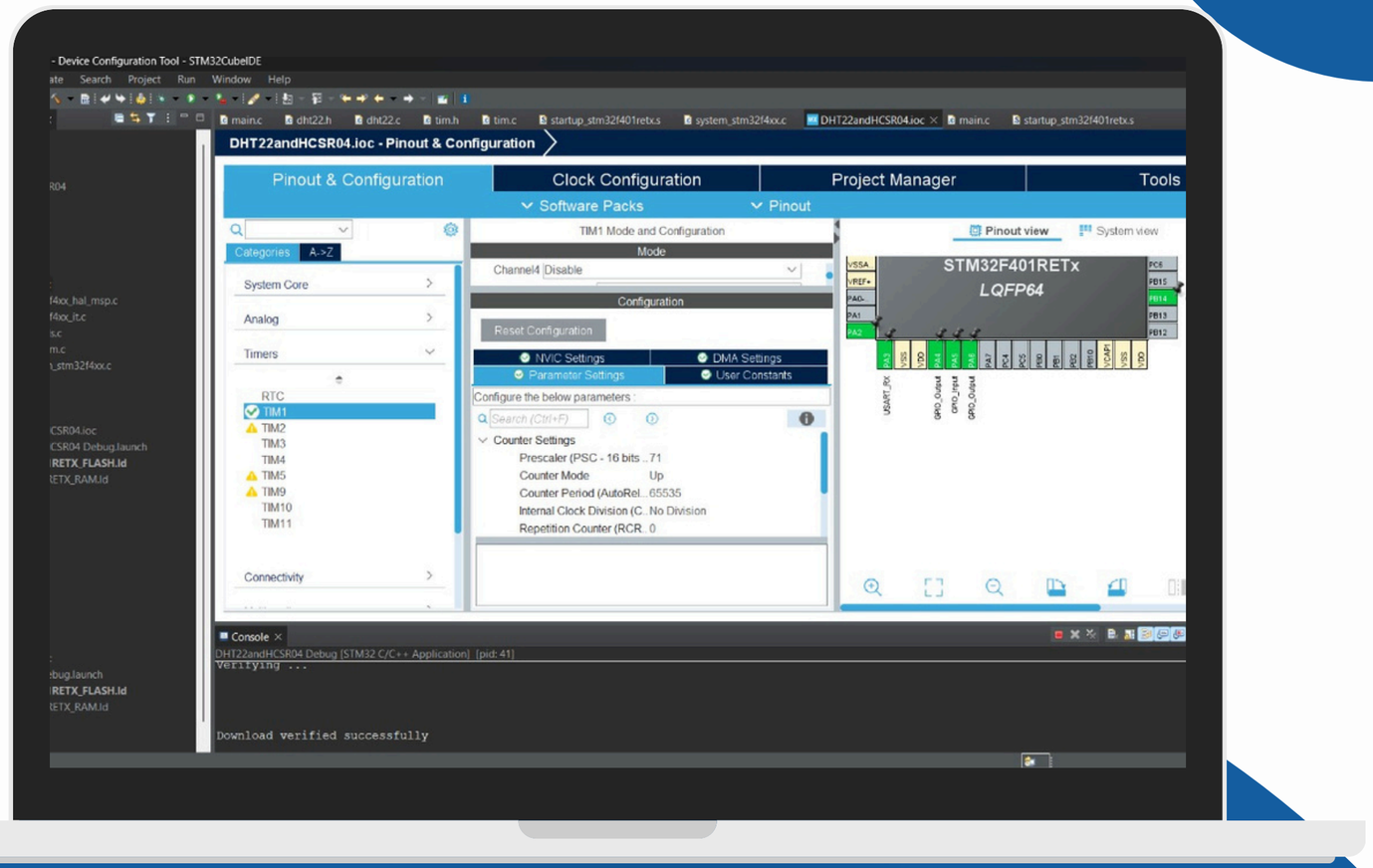
# IOT Enabled Water Monitoring and Controlling System

By: MD Musharaf Ali ( 1604-21-735-116)
MD Afhaam Ismail ( 1604-21-735-102)
MD Riyan (1604-21-735-117)

GUIDE:
Dr.MA RAHEEM




IOT Enabled Water Monitoring
and Controlling System

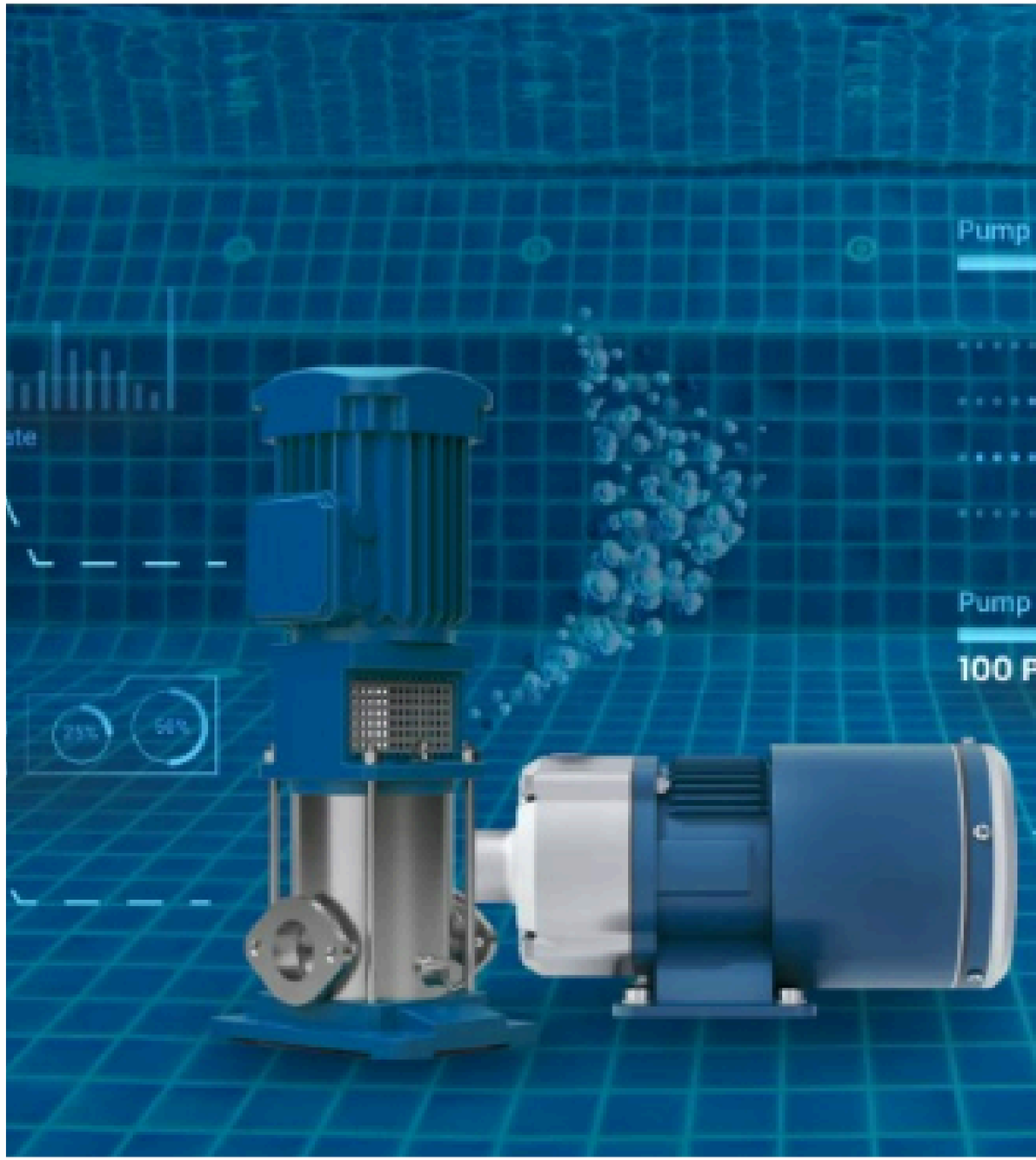
# INDEX

# Introduction

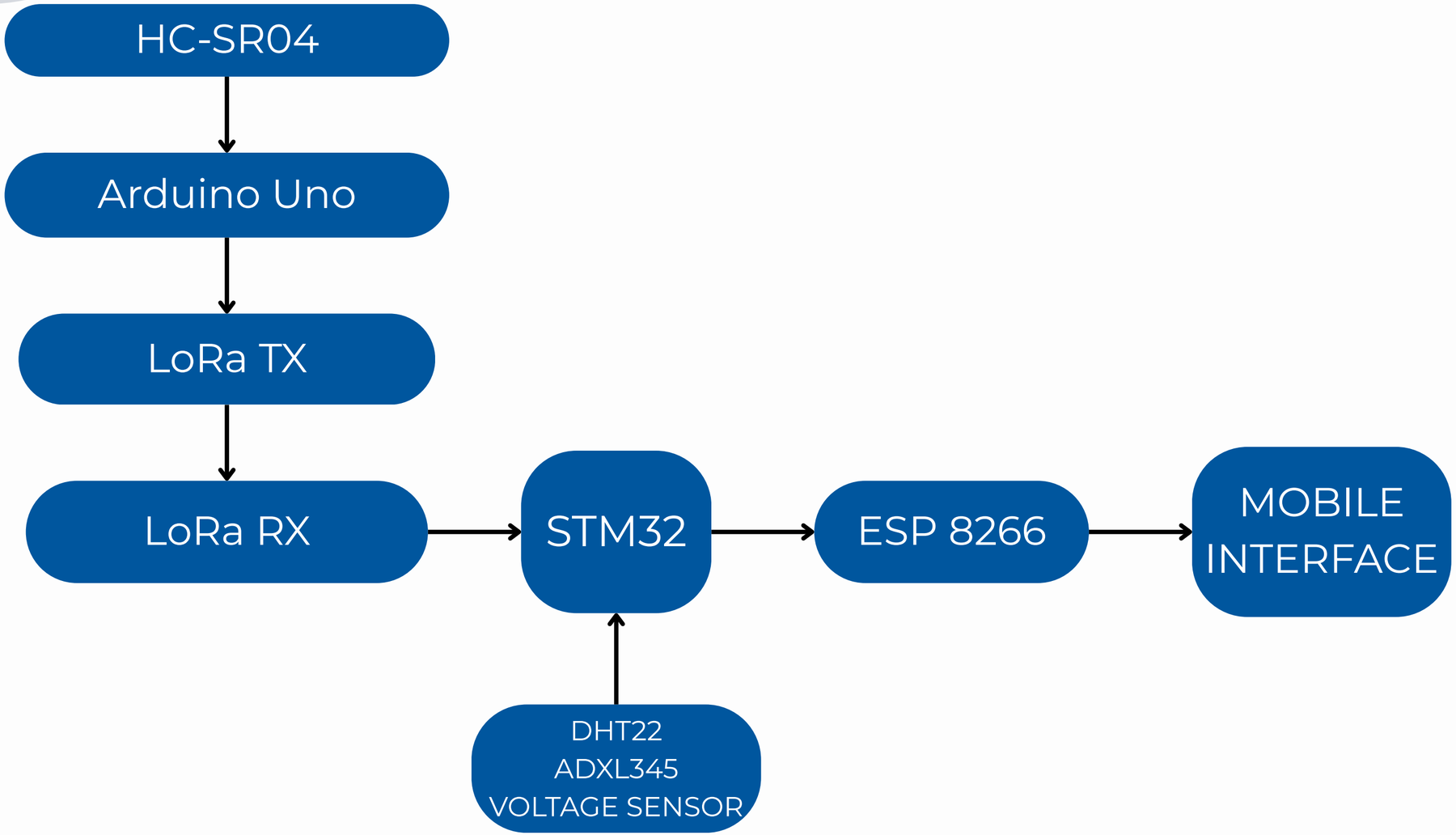
- **The project aims to automate and monitor water usage and motor control using embedded systems and IoT technologies.**
- **It combines sensor data acquisition, data processing, and wireless transmission to offer a comprehensive water management solution.**
- **Users can view real-time data such as water levels, temperature, and motor condition on a cloud-based dashboard.**
- **The system also enables remote control of a water pump via the internet, improving efficiency and reducing manual intervention**.

# Project Overview

- Arduino measures water levels.
- STM32 gathers sensor data and handles logic.
- ESP8266 sends this data to the cloud for storage and display.
- A custom-built website allows users to monitor data and toggle the relay.
- It gives a complete picture of the system’s ecosystem.



# Block Diagram



# Hardware Used

## Device-1:

- Arduino-uno
- HC-SR04 (Ultra-sonic sensor)
- LoRa Transmitter

## Device-2:

- LoRa Receiver
- STM32 NUCLEO F401RE
- DHT22 (temp sensor)
- ADXL 345
- PZEM-OO4T (voltage sensor)
- ESP8266

# Working Principle

- The HC-SR04 sensor on the Arduino Uno detects the water level and sends the data wirelessly using the LoRa transmitter.
- The LoRa receiver connected to the STM32 receives this data and integrates it with readings from other sensors like DHT22, ADXL345, and a voltage sensor.
- The STM32 processes the collected data and transmits it to the ESP8266 module using UART communication.
- The ESP8266 sends the data to a web-based interface, allowing users to monitor the system remotely.
- Commands from the mobile/web interface are received by ESP8266 and relayed back to the STM32 to switch the motor ON or OFF via the relay module.
- The system ensures reliable real-time monitoring and control of water levels and motor health from any location.

arduino | Arduino IDE 2.3.5

File Edit Sketch Tools Help

Arduino Uno

arduino.ino

```
#define TRIG_PIN 7
#define ECHO_PIN 8
void setup() {
    Serial.begin(115200);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
}
void loop() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    long duration = pulseIn(ECHO_PIN, HIGH);
    float distance = duration * 0.034 / 2;
    if (distance >= 0.0 && distance <= 200.0) {
        Serial.print("\n");
        Serial.print(distance, 2);
        Serial.print("\n");
    }
    delay(2000);
}
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on 'COM16')

Both NL & CR 115200 baud

```
114.00

114.14

114.39
```

Ln 22, Col 18 Arduino Uno on COM16 3

Gold +0.71%

Search

ENG IN 02:10 AM 09-04-2025

workspace_1.17.0 - Device Configuration Tool - STM32CubeIDE

File Edit Navigate Search Project Run Window Help

Project Explorer

- adxl
- code1
- DHT22
- DHT22andHCSR04
- esp8266
- final_proto1
- final_proto2
- HCSR04
- led blik
- lora
- major_proto1
  - Binaries
  - Includes
  - Core
    - Inc
    - Src
      - main.c
      - stm32f4xx_hal_msp.c
      - stm32f4xx_it.c
      - syscalls.c
      - sysmem.c
      - system_stm32f4xx.c
    - Startup
  - Drivers
  - Debug
  - major_proto1.ioc
  - major_proto1 Debug.launch
  - STM32F401RETX_FLASH.ld
  - STM32F401RETX_RAM.ld
- OLED
- project_1
- project_2
- project 1
- receive
- relay

main.c | stm32f4xx_hal.c | major_proto1.ioc | main.c

major_proto1.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

Analog

Timers

Connectivity

- I2C1
- I2C2
- I2C3
- SDIO
- SPI1
- SPI2
- SPI3
- USART1
- USART2
- USART6
- USB_OTG_FS

USART1 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Configuration

Reset Configuration

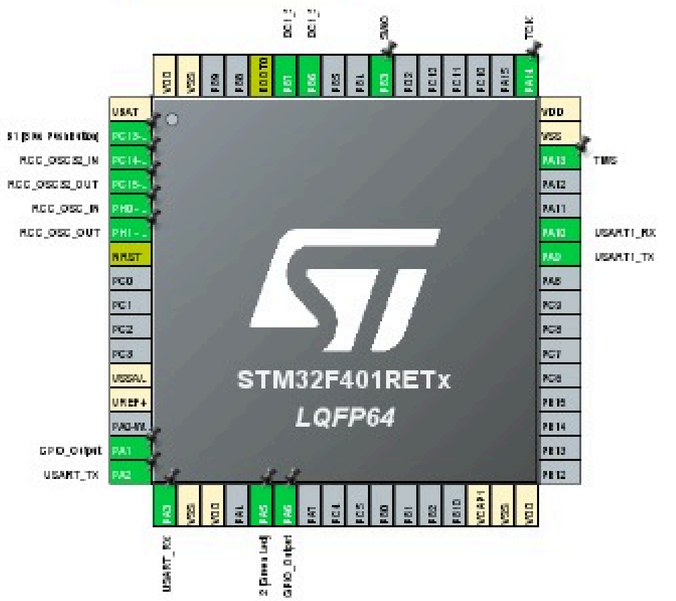
DMA Settings | GPIO Settings | User Constants | NVIC Settings | Parameter Settings

Configure the below parameters :

Search (Ctrl+F)

| | |
|---|---|
| Baud Rate | 115200 Bits/s |
| Word Length | 8 Bits (including Parity) |
| Parity | None |

Pinout view | System view




Console

<terminated> esp8266 Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Apr 8, 2025, 7:20:57 PM) [pid: 7]

```
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Shutting down...
```

Gold +0.71%

Search

ENG IN

01:52 AM 09-04-2025

workspace_1.17.0 - major_proto1/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

stm32f4xx_hal.c | main.c | startup_stm32f401retx.s

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */
    if (HAL_UART_Receive(&huart1, (uint8_t *)&rxChar, 1, 100) == HAL_OK) {
        if (rxChar == '\n' || rxChar == '\r') {
          rxBuffer[rxIndex] = '\0';
          float temp = atof(rxBuffer);
          if (temp > 0 && temp < 1000) {
            distance = temp;
          }
          rxIndex = 0;
        } else {
          if (rxIndex < RX_BUFFER_SIZE - 1) {
            rxBuffer[rxIndex++] = rxChar;
          } else {
            rxIndex = 0;
          }
        }

        if (distance >= 150.0f) {
          relayState = true;
        } else if (distance <= 50.0f) {
          relayState = false;
        }

        HAL_GPIO_WritePin(RELAY_PORT, RELAY_PIN, relayState ? GPIO_PIN_RESET : GPI

        if (HAL_GetTick() - lastUpdate >= 2000) {
          DHT22_Read();
```

Variables | Breakpoints | Expressions | Registers | Live Expressions | SFRs

| Expression | Type | Value |
|---|---|---|
| rxBuffer | char [16] | [16] |
| rxBuffer[0] | char | 49 '1' |
| rxBuffer[1] | char | 49 '1' |
| rxBuffer[2] | char | 49 '1' |
| rxBuffer[3] | char | 46 '.' |
| rxBuffer[4] | char | 54 '6' |
| rxBuffer[5] | char | 54 '6' |
| rxBuffer[6] | char | 0 '\000' |
| rxBuffer[7] | char | 0 '\000' |
| rxBuffer[8] | char | 0 '\000' |
| rxBuffer[9] | char | 0 '\000' |
| rxBuffer[10] | char | 0 '\000' |
| rxBuffer[11] | char | 0 '\000' |
| rxBuffer[12] | char | 0 '\000' |
| rxBuffer[13] | char | 0 '\000' |
| rxBuffer[14] | char | 0 '\000' |
| rxBuffer[15] | char | 0 '\000' |
| distance | volatile float | 111.660004 |
| relayState | _Bool | false |
| temperature | float | 27 |
| humidity | float | 39.0999985 |
| raw_x | | Failed to evaluate expression |
| raw_y | | Failed to evaluate expression |
| raw_z | | Failed to evaluate expression |
| x | | Failed to evaluate expression |
| y | | Failed to evaluate expression |

Console | Problems | Executables | Debugger Console | Memory

major_proto1 Debug [STM32 C/C++ Application] [pid: 10]

```
Verifying ...


Download verified successfully
```

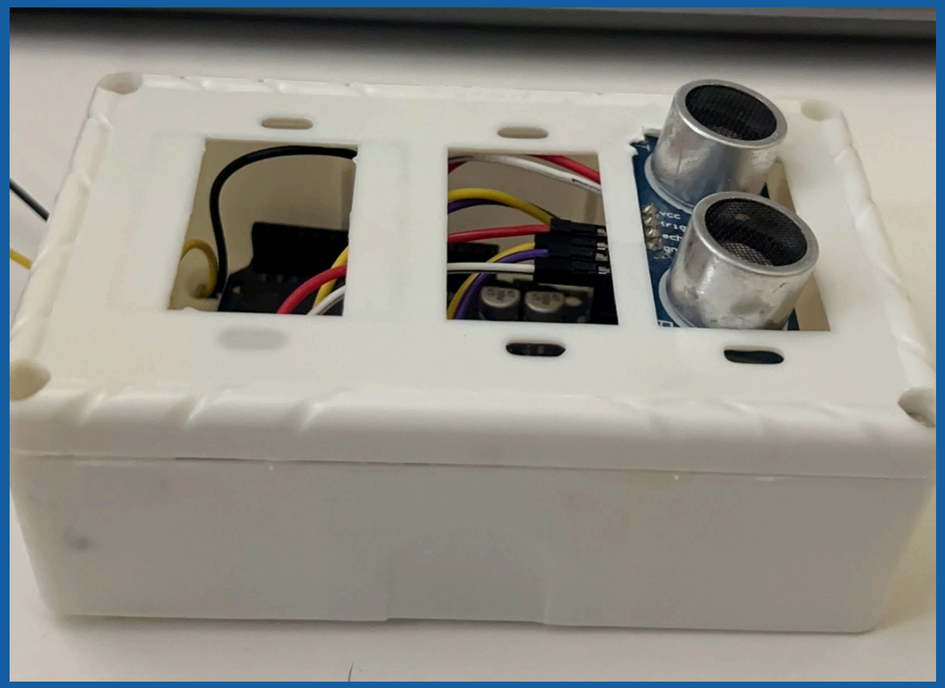
78°F Mostly clear

Search

ENG IN

02:25 AM 09-04-2025

# Data Flow-



01 Sensor Data Collection: STM32 collects readings from connected sensors like DHT22, ADXL345, voltage sensor, and LoRa receiver.

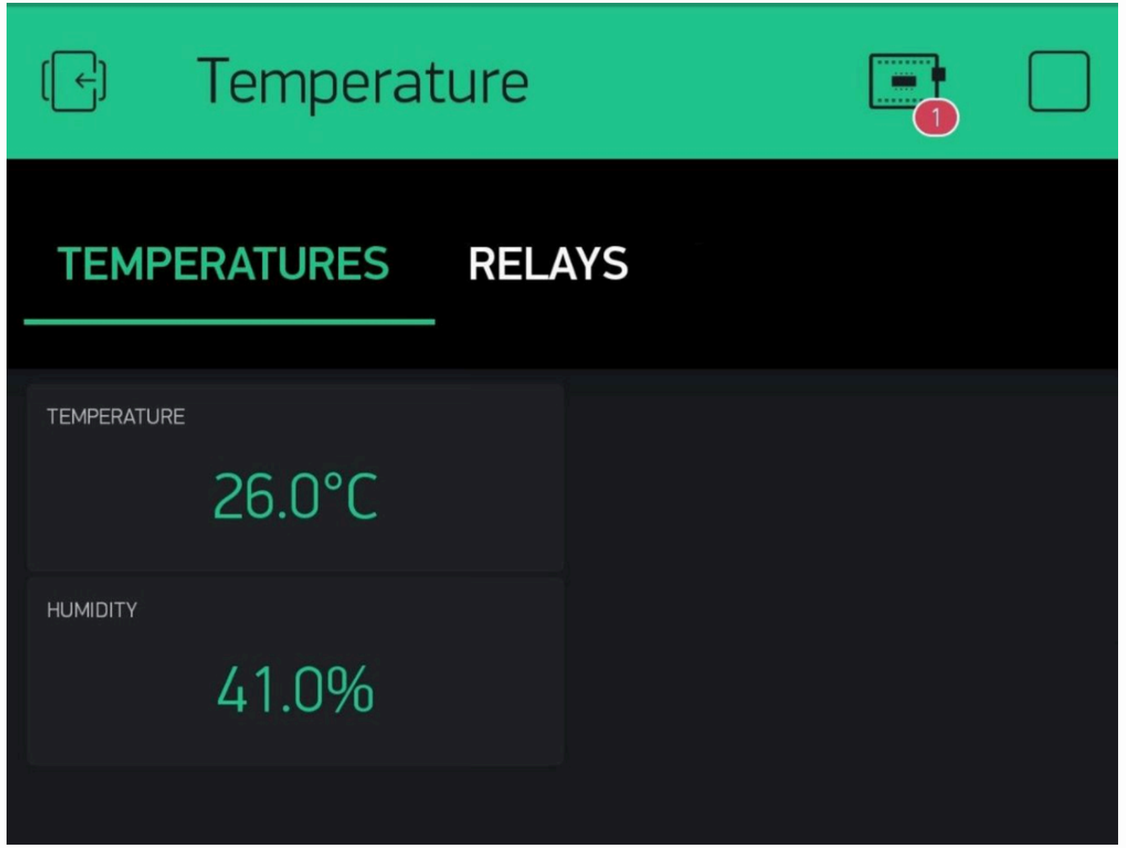
02 Data Transmission: STM32 sends this sensor data to the ESP8266 via UART communication.

03 Cloud Upload: ESP8266 uploads the received data to the cloud using HTTP or MQTT protocols.

04 Remote Control: The website displays the data in real-time, and users can send control commands back to operate the relay remotely.

# Website Interface

The site shows live values like distance, temperature, and voltage.
It includes a simple switch/button to turn the motor on or off.
The interface is responsive for use on both phones and desktops.
It bridges the gap between hardware and user.



# BENEFITS

- Remote Monitoring & Control:

Users can conveniently monitor water levels, motor status, and environmental data from anywhere through an internet-connected device, ensuring full control without being physically present.

- Prevention of Motor Damage:

The system helps prevent dry run conditions by accurately monitoring water levels, thus protecting the motor from damage and reducing unnecessary power consumption.

- Cost-Effective & Scalable Design:

Designed using affordable and readily available components, the system can be easily expanded or upgraded to meet growing or changing requirements.

- Versatile Applications:

Suitable for use in agricultural farms, domestic water tanks, and industrial setups—making it a flexible solution for various water management needs.

# Upcoming Enhancements

- **Integration of ESP8266 for Cloud Connectivity:**

**Plan to integrate ESP8266 to push real-time sensor data to the cloud and allow remote monitoring through a mobile-friendly web interface.**

- **Bi-directional Communication:**

**Enable command reception from the cloud to control the motor relay remotely, giving full two-way control of the water system.**

- **Addition of Voltage and Vibration Monitoring:**

**Implement PZEM-004T and ADXL345 sensors to monitor power consumption and motor vibration, improving reliability and enabling predictive maintenance**

# Conclusion

- The project successfully enabled remote monitoring of water levels and environmental conditions using a combination of STM32, sensors, and wireless communication.

- Real-time data collection and control were achieved through a user-friendly website, allowing users to view sensor readings and operate the motor from anywhere.

- The system has proven to be highly cost-effective, scalable, and reliable, making it suitable for a wide range of applications including farms, households, and small industries.

- Overall, this project demonstrates a practical and impactful use of IoT technology, with strong potential for future upgrades such as mobile integration, predictive maintenance, and solar-powered operation.

# THANK YOU!