12.01.2026

WATER LEAKAGE DETECTOR

IOT FINAL PROJECT GROUP 5

ABDULKADİR KÖSEOĞLU

NO	NAME	STUDENT ID
1	NUR ADRIANA BINTI YUSERIZAL	2513011035
2	AINUR BATRISYIA BINTI MUHAMAD ZAIRUL	2513011034
3	KÜBRA CERİT	2111051012
4	NOMAN TANVEER	2511011089

Table of Contents

1.0 Introduction

Water leakage is one of the most common yet underestimated problems in residential and urban environments. Undetected leaks can lead to significant water waste, infrastructure damage, and increased maintenance costs. With the rapid growth of smart city and smart home technologies, Internet of Things (IoT) based monitoring systems have become an effective solution for early detection and prevention of such issues.

The objective of this project is to design and implement a Smart Water Leakage Detection System using an ESP32 microcontroller, multiple water leakage sensors, and a cloud-based monitoring platform (Blynk IoT). The system continuously monitors moisture levels at different locations, analyses the data using a TinyML-inspired intelligent classification approach, and notifies users in real time through a live dashboard and event-based alerts.

The project aims to develop an IoT-based smart water leakage detection system capable of:

- Early leakage detection
- Severity classification
- Cloud-based monitoring and alerting

This project supports SDG 6 (Clean Water and Sanitation) and SDG 11 (Sustainable Cities and Communities) by enabling early detection of water leaks, reducing water wastage, and promoting efficient water management. It also encourages smarter and safer living environments using IoT-based monitoring and control. The system is designed for home use, with two sensors placed in the kitchen and living room. When a leak is detected, a real-time notification is sent through the Blynk app, allowing quick action to prevent damage, improve safety, and reduce maintenance costs.

2.0 System Model

2.1 System architecture

The system architecture is designed to provide a simple and efficient way to detect, analyse, and report water leakage in a smart home environment. It is divided into four main layers that work together to ensure accurate sensing, intelligent processing, reliable communication, and clear user interaction. This layered design makes the system more organized, scalable, and easy to understand.

1. **Sensing Layer**

   The sensing layer is responsible for detecting the presence of water. It uses two analog water leakage sensors placed in the kitchen and living room to measure moisture levels. These sensors continuously monitor the environment and generate analog signals based on how wet the surface is. This layer acts as the first point of contact between the physical environment and the system.

2. **Processing Layer**

   The processing layer uses the ESP32 microcontroller to read and analyze the sensor data. The ESP32 converts the raw sensor values into meaningful information and applies a TinyML-inspired classification algorithm to determine the leakage severity, such as Normal, Detecting, Minor, or Major. This layer provides the intelligence of the system by making decisions based on sensor inputs.

3. **Communication Layer**

   The communication layer enables data transfer between the ESP32 and the cloud platform. Using a WiFi connection, the ESP32 sends processed information to the Blynk cloud servers. This allows real-time data sharing and makes remote monitoring possible from anywhere using an internet connection.

4. **Application Layer**

   The application layer is the user interface of the system. It displays real-time information on a dashboard in the Blynk mobile application, including moisture levels, leakage location, and severity status. It also sends event-based notifications to alert users immediately when a serious water leak is detected.

In conclusion, the four-layer architecture ensures that water leakage is detected quickly and handled intelligently. Each layer plays an important role in transforming raw sensor data into useful information and timely alerts for the user. This structured approach improves system reliability, response time, and overall effectiveness in preventing water damage.

2.2 Interaction Flow

Sensors → ESP32 → TinyML Classifier → Blynk Cloud → User Dashboard

The interaction starts when the water leakage sensors in the living room and kitchen detect moisture. The sensor readings are sent to the ESP32, where the data is analysed. The ESP32 checks how wet the area is, how fast the moisture changes, and how long the surface stays wet. Using a TinyML-inspired method and simple thresholds, the system classifies the condition as Normal, Detecting, Minor, or Major leakage.

After that, the ESP32 sends the results to the Blynk cloud through WiFi. The information is shown on the Blynk dashboard in real time, including the leakage location and its severity. If a Minor or Major leak is detected, the system sends an instant notification to the user's phone so action can be taken quickly to prevent further damage.

3.0 Hardware Design-Model

The hardware architecture of the system is centered around an ESP32 development board, which acts as the main processing and communication unit.

Main Components:

- ESP32 Microcontroller
- Two analog water leakage sensors
- Green LED (Normal condition indicator)
- Red LED (Leakage warning indicator)
- Buzzer (Audible alert)

Pin Configuration:

- Water Sensor A (Living Room): GPIO 35 (ADC)
- Water Sensor B (Kitchen): GPIO 34 (ADC)
- Green LED: GPIO 23
- Red LED: GPIO 22
- Buzzer: GPIO 25

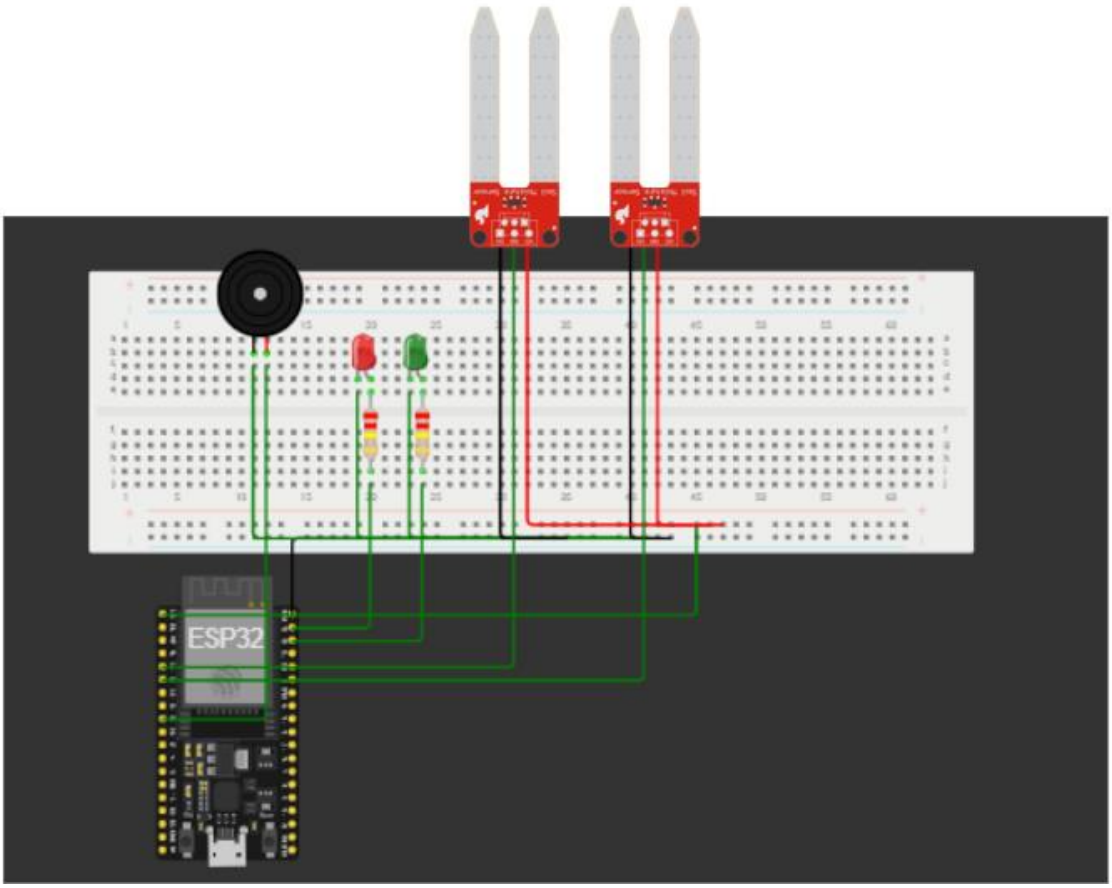


Figure 1. Hardware Design

4.0 Software Design

a) Software Architecture

The software uses a non-blocking design with BlynkTimer so the system can run smoothly without stopping other tasks. The updateSystem() function runs every second to read sensors, classify leakage, send alerts, and update the Blynk cloud. The code is modular, with clear sections for sensing, classification, alerting, and cloud updates, making it efficient and easy to manage.

b) Leak Detection Algorithm

The leak detection algorithm uses threshold checking, time tracking, and change rate analysis. Sensor values below a set threshold indicate moisture, while millis() is used to measure how long the area stays wet. Sudden changes in sensor readings are also detected to identify possible burst leaks, making the system reliable for both slow and fast leakage detection.

c) Flowchart of the water leak program

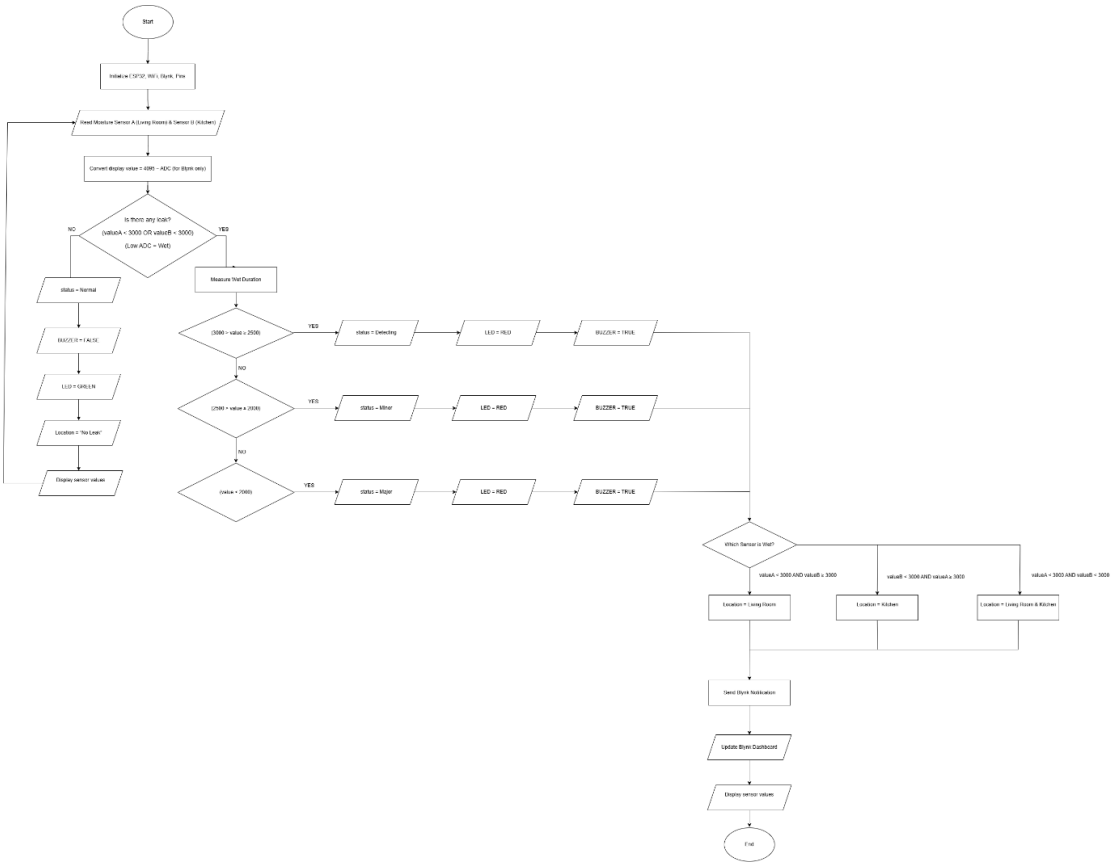


Figure 2. WaterLeakage Flowchart

The system starts by initializing the ESP32, Wi-Fi, Blynk, and all pins, then continuously reads the moisture values from the living room and kitchen sensors. In the code, the ADC range is from 0 to 4095, where 4095 means completely dry and lower values indicate more moisture. The ESP32 checks if any sensor value is below 3000 to decide if a leak exists. If no leak is detected, the status is set to *Normal*, the green LED turns on, the buzzer stays off, and "No Leak" is shown on the dashboard and Serial Monitor.

When a leak is detected, the system classifies it as *Detecting*, *Minor*, or *Major* based on the sensor values. The red LED and buzzer are activated, the location of the leak (living room, kitchen, or both) is identified, and a notification is sent through Blynk. In the Serial Monitor, 4095 still represents dry, but in Blynk the value is shown as 4095 - sensorValue so that higher values mean higher leakage, making the display more user-friendly and easier to understand.

d) TinyML Integration

This project integrates a TinyML-inspired approach by using a lightweight, rule-based classifier instead of a complex machine learning model. This design choice makes the system suitable for the ESP32, which has limited memory and processing power. The classifier simulates TinyML behavior by making intelligent decisions based on simple mathematical rules applied to real-time sensor data, ensuring fast response and low computational cost.

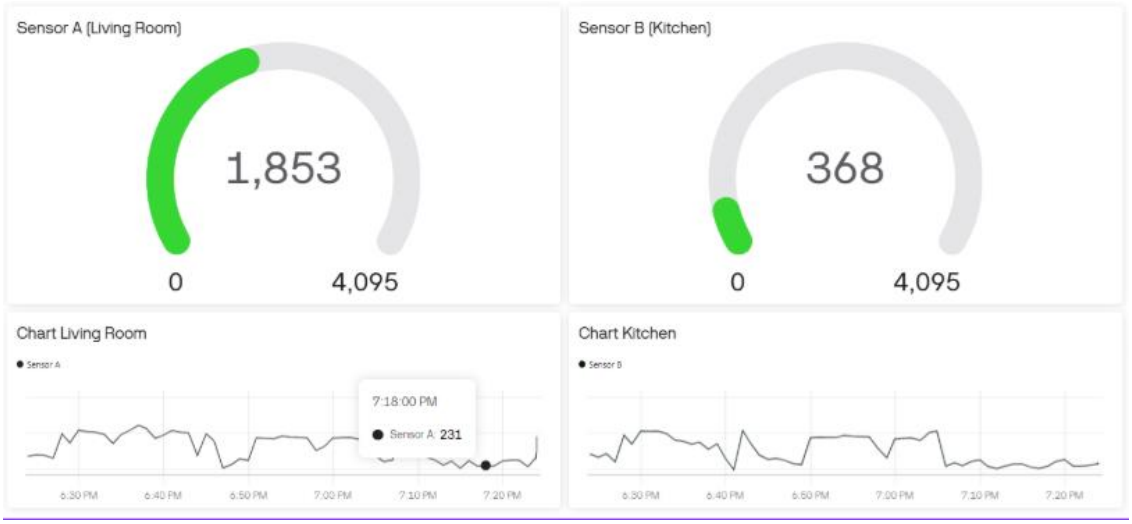
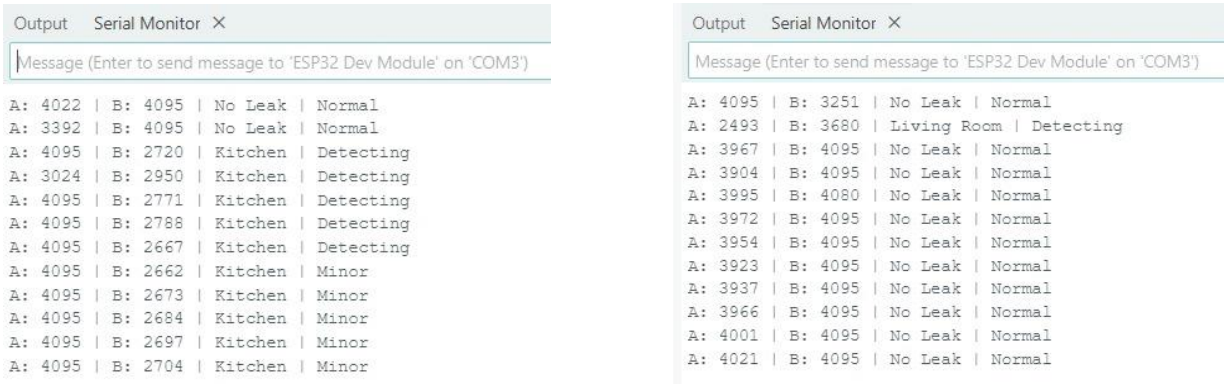


Figure3. Dashboard Design



Figures4.Alerts

TinyML-Based Classification Approach

Instead of deploying a computationally expensive machine learning model, this project implements a TinyML-inspired rule-based classifier optimized for embedded systems.

Input Features:

- Normalized sensor value
- Rate of change of sensor readings
- Duration of wet condition

Output Classes:

- Normal
- Detecting
- Minor Leak
- Major Leak

This approach improves detection accuracy while minimizing false alarms and computational overhead, making it suitable for low-power IoT devices.

Prototype Implementation:

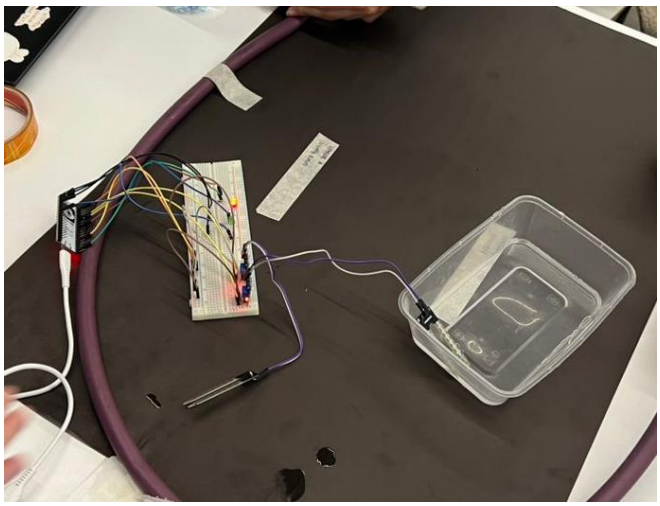


Figure5.Prototype

5.0 Results and Discussion

5.1 System Performance

The system successfully detected water leakage events at different severity levels in real time. The dual-sensor setup allowed independent monitoring of multiple locations, improving coverage and reliability.

5.2 Key Observations

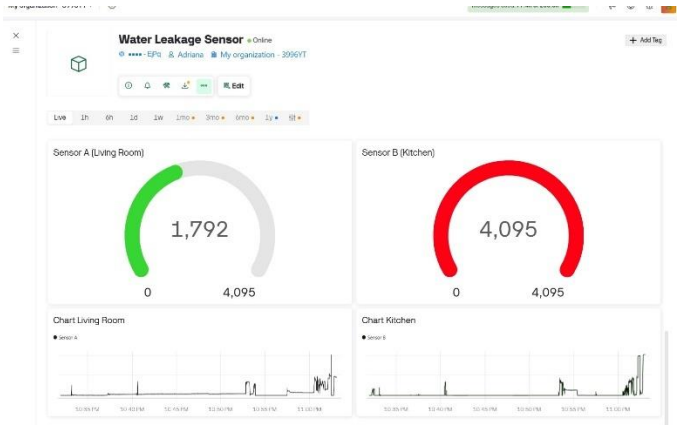


Figure6.Dashboard Design

This picture shows the real-time sensor graph on the Blynk dashboard. When the system is in a normal and dry condition, the graph remains green, indicating no leakage. In the code, this happens when the severity is set to "Normal". When water is detected and the severity changes to *Detecting*, *Minor*, or *Major*, the system considers it an abnormal condition and the graph changes to red. This color change helps users quickly understand that a leak has occurred. Since the value sent to Blynk is 4095 - sensorValue, higher graph values represent more water, making the visualization more intuitive and user-friendly.

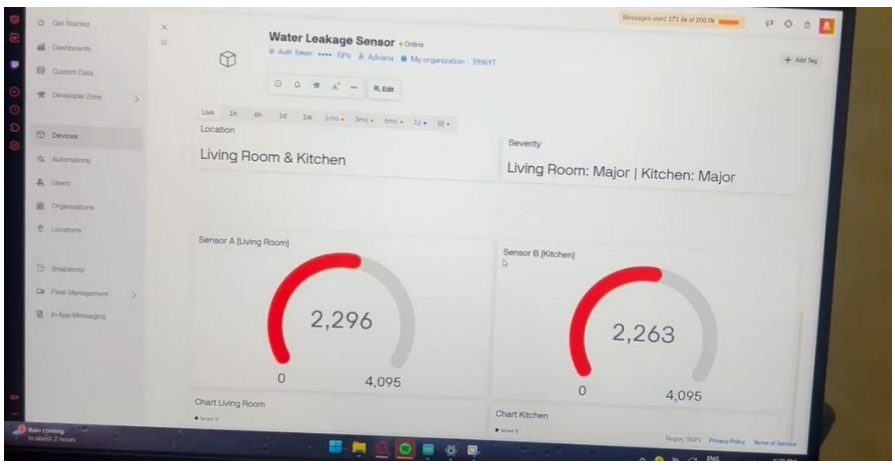


Figure7.Location & Severity Level

The label displayed on the Blynk dashboard clearly shows the leakage location and its severity level in real time. In the code, this is updated using Blynk.virtualWrite(V2, loc) for the location and Blynk.virtualWrite(V3, sev) for the severity. When a leak occurs, the label

automatically changes to show whether the problem is in the Living Room, Kitchen, or both, and whether it is Detecting, Minor, or Major. This allows users to quickly understand where the leakage is happening and how serious it is without checking raw sensor values, making the system simple and user-friendly.

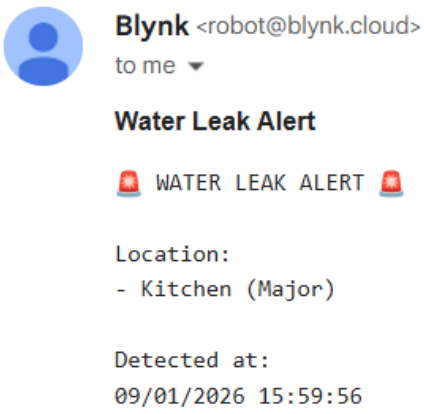


Figure8.Alert Notification

This picture shows the notification sent by Blynk when a Minor or Major leak is detected. In the code, this is done using Blynk.logEvent("water_leak", msg);. The notification is only triggered when the condition is serious and an alert has not already been sent. The message includes the warning symbol, the detected location, the severity level, and the time of detection. This proves that the system can alert the user instantly so that action can be taken immediately.

5.3 Challenges

One of the main challenges is the calibration of sensor thresholds for different environments. Moisture levels can vary depending on room conditions such as humidity, temperature, and floor material. A threshold that works well in one house may not be suitable for another, so careful adjustment is required to avoid false detections or missed leaks.

Another challenge is handling sensor noise and sudden spikes in readings. Water leakage sensors can sometimes produce unstable values due to electrical noise, dust, or momentary splashes of water. These sudden changes can affect accuracy and must be filtered properly to prevent false alarms.

Ensuring stable WiFi connectivity is also critical because the system depends on the internet to send data to the Blynk cloud and deliver notifications. Any network interruption can delay alerts or cause temporary loss of monitoring, which reduces system reliability.

5.4 Future Scope

In the future, the rule-based classifier can be replaced with a trained TinyML model. This would allow the system to learn from real data, improve classification accuracy, and better distinguish between real leaks and temporary moisture conditions.

More sensor nodes can be added to cover additional rooms such as bathrooms, laundry areas, or basements. This would expand system coverage and make it more suitable for larger houses or buildings.

Predictive analytics can be integrated to analyse historical data and identify patterns that may indicate potential leaks before they happen. This would shift the system from reactive detection to proactive prevention.

Finally, the system can be scaled for smart city deployment by integrating it into larger water management networks. With multiple sensor nodes across buildings or neighbourhoods, authorities could monitor water usage, detect pipeline leaks early, and improve urban water sustainability.

Source Code

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL6CB2Yd4ew"
#define BLYNK_TEMPLATE_NAME "Water Leakage Sensor"
#define BLYNK_AUTH_TOKEN "ILyrjzHyqainROcdK0iu5Ngixd2NEjPq"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <time.h>

/* WIFI */
char ssid[] = "AGUStudent";
char pass[] = "Un7a38uN";

/* PINS */
#define SENSOR_A 35   // Living Room
#define SENSOR_B 34   // Kitchen

#define GREEN_LED 23
#define RED_LED   22
#define BUZZER    25

/* THRESHOLDS */
#define MICRO_TH 3000
#define MINOR_TH 2500
#define MAJOR_TH 2000

/* SMART ANALYTICS */
#define MIN_WET_TIME 5000
#define WARNING_TIME 120000
#define BURST_CHANGE 600

BlynkTimer timer;

/* SMART VARIABLES */
unsigned long wetStartA = 0;
unsigned long wetStartB = 0;

bool wasWetA = false;
bool wasWetB = false;

int lastValueA = 4095;
int lastValueB = 4095;

bool alertSent = false;

/* TIME */
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3 * 3600;
const int daylightOffset_sec = 0;
```

```
bool timeReady = false;

/* TINYML CLASSIFIER  */
/*
0 = Normal
1 = Detecting
2 = Minor
3 = Major
*/

int tinyML_predict(float valueNorm, float changeNorm, float timeNorm) {

  if (valueNorm > 0.75 && changeNorm < 0.15)
    return 0;

  if (timeNorm < 0.2)
    return 1;

  if (valueNorm <= 0.75 && valueNorm > 0.55)
    return 2;

  if (valueNorm <= 0.55 || changeNorm > 0.4)
    return 3;

  return 0;
}

String tinyML_toSeverity(int cls) {
  if (cls == 1) return "Detecting";
  if (cls == 2) return "Minor";
  if (cls == 3) return "Major";
  return "Normal";
}

/* GET TIME STRING */
String getDateTime() {
  if (!timeReady) return "Time syncing...";
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) return "Time error";

  char buffer[30];
  strftime(buffer, sizeof(buffer), "%d/%m/%Y %H:%M:%S", &timeinfo);
  return String(buffer);
}

/* MAIN FUNCTION */
void updateSystem() {

  int valueA = analogRead(SENSOR_A);
  int valueB = analogRead(SENSOR_B);
```

```
bool wetA = valueA < MICRO_TH;
bool wetB = valueB < MICRO_TH;

unsigned long now = millis();

if (wetA && !wasWetA) wetStartA = now;
if (wetB && !wasWetB) wetStartB = now;

if (!wetA) wetStartA = 0;
if (!wetB) wetStartB = 0;

unsigned long durA = wetA ? now - wetStartA : 0;
unsigned long durB = wetB ? now - wetStartB : 0;

int changeA = lastValueA - valueA;
int changeB = lastValueB - valueB;

lastValueA = valueA;
lastValueB = valueB;

wasWetA = wetA;
wasWetB = wetB;

/* TinyML INPUT (Sensor A) */
float mlValueA  = valueA / 4095.0f;
float mlChangeA = abs(changeA) / 4095.0f;

float mlTimeA = durA / (float)MIN_WET_TIME;
if (mlTimeA > 1.0f) mlTimeA = 1.0f;

int mlClassA = 0;
if (wetA) {
  mlClassA = tinyML_predict(mlValueA, mlChangeA, mlTimeA);
}

String mlSeverityA = tinyML_toSeverity(mlClassA);

String loc = "No Leak";
String sev = "Normal";
String sevA = "";
String sevB = "";

if (wetA) sevA = mlSeverityA;

if (wetB) {  // Threshold
if (valueB <= MAJOR_TH) {
  sevB = "Major";
}
else if (valueB <= MINOR_TH) {
  sevB = "Minor";
}
```

```
  else {
    sevB = "Detecting";
  }
}


  if (!wetA && !wetB) {
    loc = "No Leak";
    sev = "Normal";
  }
  else if (wetA && wetB) {
    loc = "Living Room & Kitchen";
    sev = "Living Room: " + sevA + " | Kitchen: " + sevB;
  }
  else if (wetA) {
    loc = "Living Room";
    sev = sevA;
  }
  else if (wetB) {
    loc = "Kitchen";
    sev = sevB;
  }

  /* ALERT */
 bool isSerious =
  (sevA == "Minor" || sevA == "Major") ||
  (sevB == "Minor" || sevB == "Major");

if (isSerious && !alertSent && Blynk.connected()) {

  String msg = "🚨 WATER LEAK ALERT 🚨\n\n";

  msg += "Location:\n";
  if (wetA) msg += "- Living Room (" + sevA + ")\n";
  if (wetB) msg += "- Kitchen (" + sevB + ")\n";

  msg += "\nDetected at:\n";
  msg += getDateTime();

  Blynk.logEvent("water_leak", msg);
  alertSent = true;
}

if (!wetA && !wetB) {
  alertSent = false;
}


  /* OUTPUT */
sev.trim();  // removes leading/trailing whitespace
```

```
digitalWrite(GREEN_LED, sev.equals("Normal"));
digitalWrite(RED_LED, !sev.equals("Normal"));
digitalWrite(BUZZER, sev.equals("Normal"));

  /* BLYNK UPDATE */
  if (Blynk.connected()) {
    Blynk.virtualWrite(V0, 4095 - valueA);
    Blynk.virtualWrite(V1, 4095 - valueB);
    Blynk.virtualWrite(V2, loc);
    Blynk.virtualWrite(V3, sev);
  }

  Serial.print("A: "); Serial.print(valueA);
  Serial.print(" | B: "); Serial.print(valueB);
  Serial.print(" | "); Serial.print(loc);
  Serial.print(" | "); Serial.println(sev);
}

/* SETUP */
void setup() {
  Serial.begin(115200);
  analogReadResolution(12);

  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(GREEN_LED, HIGH);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  while (!Blynk.connected()) {
    Blynk.run();
  }

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

struct tm timeinfo;
if (getLocalTime(&timeinfo)) {
  timeReady = true;
} else {
  timeReady = false;
}


  timer.setInterval(1000L, updateSystem);
}

/* LOOP */
void loop() {
  Blynk.run();
```

```
  timer.run();
}
```

Demo

https://drive.google.com/file/d/1Gt_sbEM6mqlQxTEQ0L5lzsnXs_kcbU2/view?usp=drive_link

Live Dashboard Link

https://blynk.cloud/dashboard/965794/templates/edit/2170472/dashboard

WATER LEAKAGE DETECTOR – GROUP 5

AGÜ

AGÜ

PROBLEM STATEMENT

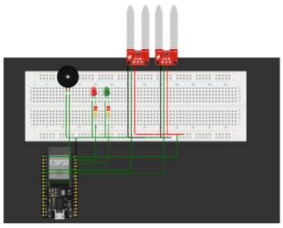
- Undetected leaks lead to water waste, high repair costs, electrical hazards, and structural damage.
- Traditional detection methods rely on manual inspection, which is inefficient and unreliable.
- Therefore, an automated and real-time water leakage detection system is required.

OBJECTIVES

- Detect water leakage at an early stage
- Provide real-time alerts to users by Gmail
- Reduce water waste and property damage
- Offer a low-cost and easy-to-install solution
- Enable integration with smart home and office systems

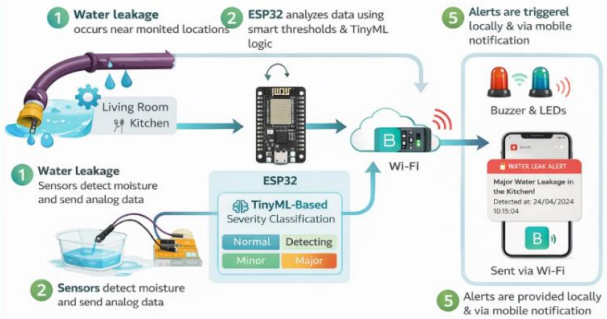
SAVE WATER

HARDWARE-SOFTWARE DESIGN



Software Technologies Used:
Arduino IDE
C++
Blynk IoT platform
Wi-Fi communication

Proposed System



REFERENCES

6 CLEAN WATER AND SANITATION

11 SUSTAINABLE CITIES AND COMMUNITIES

RESULTS

Experimental tests show that the system successfully:

- Triggers audio-visual alerts instantly
- Sends notifications to the user's mobile device
- Detects water leakage in real time

Future Improvements

- Automatic water valve shut-off system
- Multi-sensor support for large areas
- Cloud-based data analysis
- Enhanced decision-making using tinyML

COMP 413 - Internet of Things - Dr. Abdulkadir Köse

AINUR BATRISYIA BINTI MUHAMAD ZAIRUL
NOMAN TANVEER
KÜBRA CERİT
NUR ADRIANA BINTI YUSERIZAL