SmartDrop

A Real Time Irrigation Soil Monitoring Irrigation System

EED308 - Embedded Systems

Group 11:

Suchit Reddi
Akshat Punia
Mukku Sohan
Kolli Sreeram Sai
Akshay Veeragandham
Medikonda B V Sai Praveen Reddy

Under the supervision of:

Dr. Rohit Singh
Dr. Sonal Singhal
Mr. Govinda
Mr. Vakil

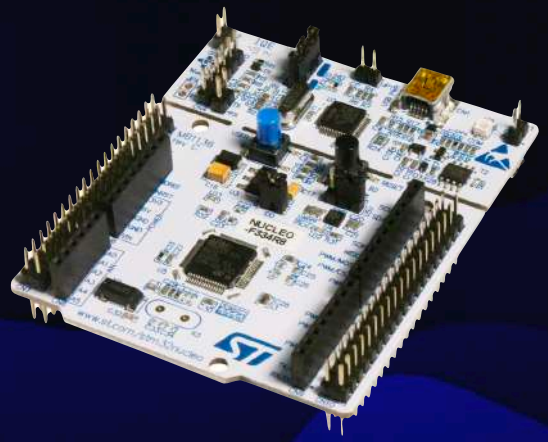
Rain Sensor
SmartDrop

Abstract

The proposed project aims to develop a Real-Time Soil Monitoring Irrigation System using STM32microcontroller and sensors. The system includes a soil moisture sensor that provides moisture data and controls the irrigation system using a DC Servo motor. The system also includes a rain sensor to detect rainfall and adjust the water flow accordingly. The main objective of this project is to design an efficient and cost-effective system that can maintain the optimal moisture level in the soil and save water by controlling the irrigation system based on soil and weather conditions.

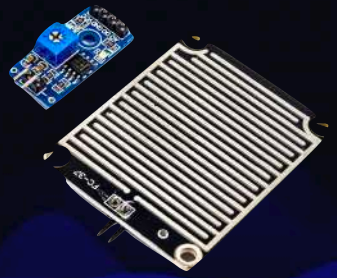
This project involves designing and programming the STM32 microcontroller, interfacing the sensors with the microcontroller, and assembling the hardware components. The proposed system will offer several benefits, including reduced water usage, improved crop yield, and increased efficiency in agricultural practices.

Hardware Overview

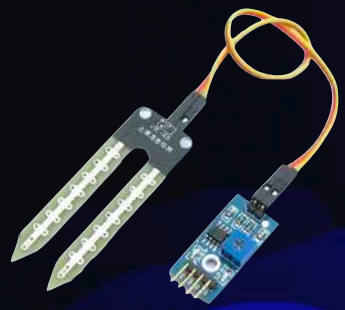


STM32F303RET6 Board

JQC-3FF-S-Z Relay

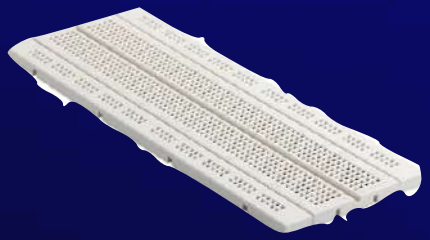


FC-37 Rain Sensor



YL-69 Soil Sensor

Buzzer



Bread Board

DC Motor

Introduction

Irrigation plays a critical role in modern agriculture, allowing farmers to control the water supply to crops and maximize yields. However, traditional irrigation systems that rely on manual control or fixed schedules have several drawbacks. They can lead to overwatering or underwatering, which can harm crops, waste water, and lead to increased costs for farmers.

Additionally, traditional systems cannot adapt to changing weather conditions since they have fixed timings for turn on and off.

Efficient irrigation systems are essential for sustainable agriculture, especially in regions with water scarcity or limited resources. By using technology to optimize water usage and reduce waste, farmers can improve their yields while also reducing costs and conserving resources.

Objectives

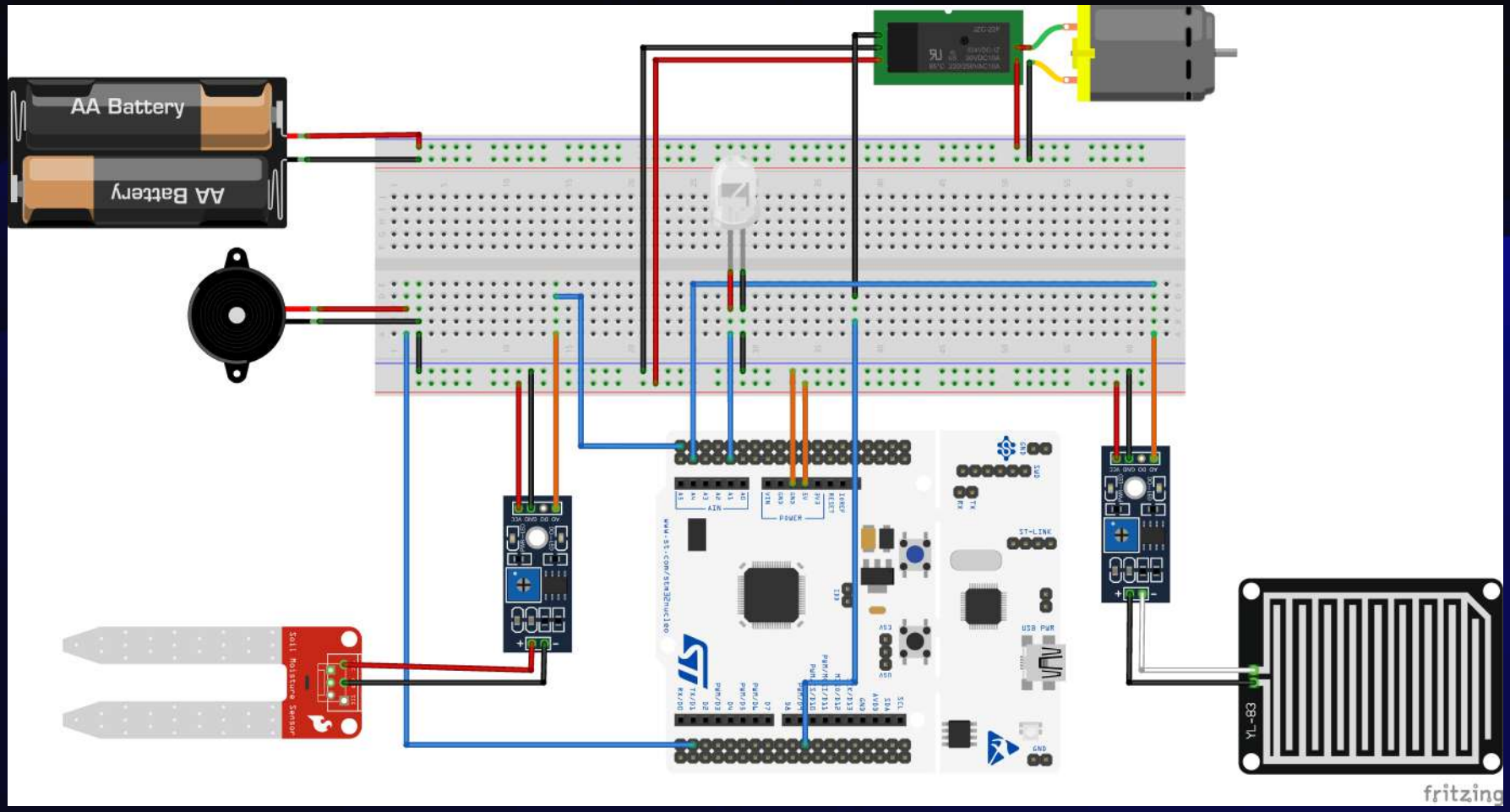
The main objective of this project is to design an efficient and cost-effective irrigation system that can maintain optimal soil moisture levels for crops at all times irrespective of weather conditions.

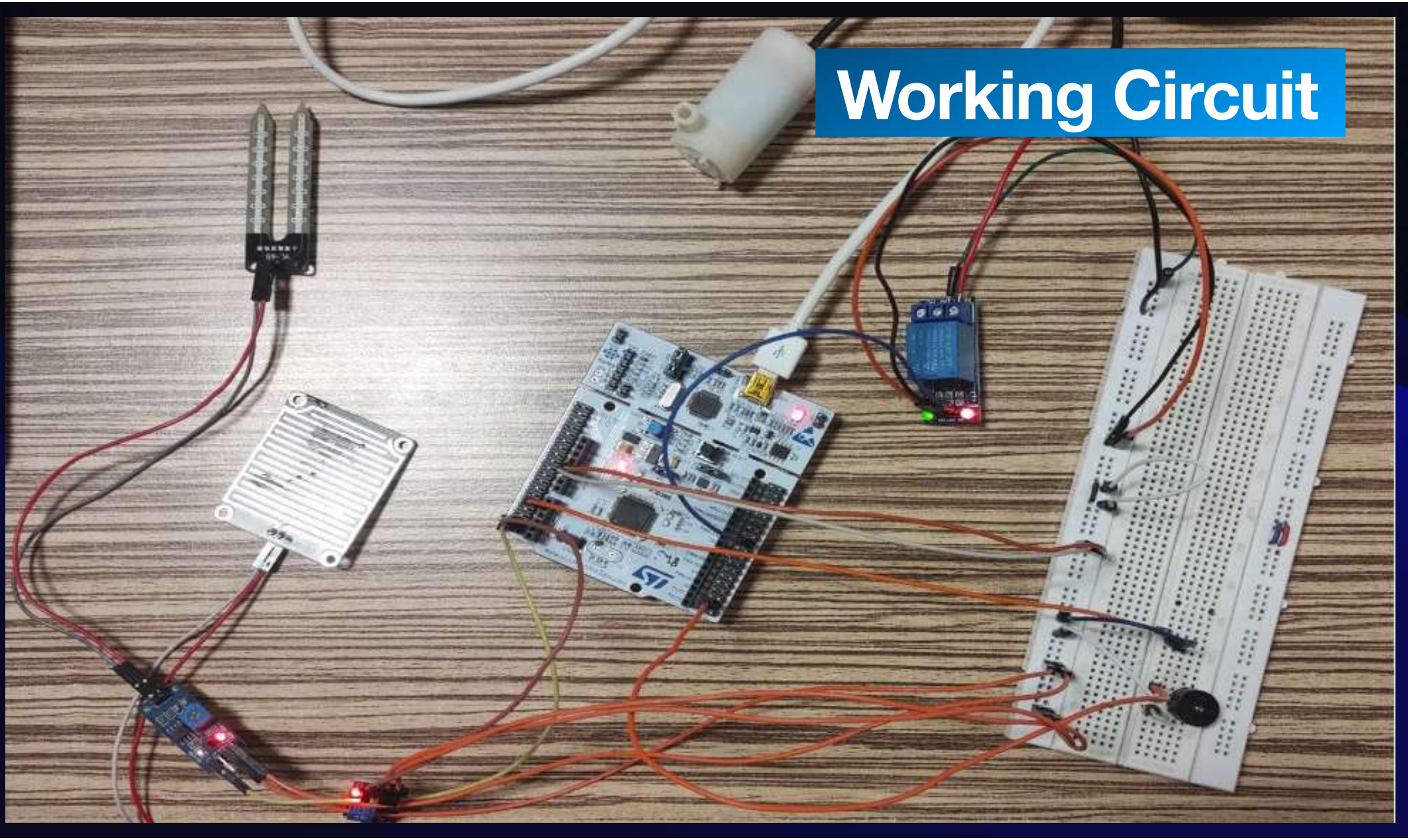
Our system is able to adapt to changing weather conditions and adjusts watering schedules based on soil moisture levels and other environmental factors.

In addition that, we aim to reduce the water usage by using technology to precisely control the amount of water applied to crops. This will not only help to conserve water resources but also reduce costs for farmers.

Other objectives of the project include improving crop yields and quality, reducing labor requirements for irrigation since we have automated the entire system in such a way that is easy to use and maintain for farmers.

Schematic




Working Circuit

Program Code and Logic

This program is written in C language for STM 32 microcontroller. It reads the analog input signals from two sensors (rain & soil moisture sensors) , converts them to a percentage, and uses the values to control a motor. We used the STM32F103RE microcontroller and it is written using STM32CubeIDE framework.

We start by initialising the peripherals and starting the ADC conversion. STM32 reads the values from the two sensors and the signal is converted to a 12 bit digital signal. Since the value is of 12 bits, we divide by 4095 to normalise the value and multiply by 100 to obtain a percentage value. It then uses these moisture percentages and calculates the motor control value which in turn turns on and off the servo motor.

If the soil moisture percentage is below 14% or the rain percentage is below 40%, the motor is turned off. Otherwise, the motor is turned on. (The limiting percentage values 14% and 40 are derived in our testing and are purely experimental).

Program Code and Logic

The program runs in an infinite loop and waits for 1 second before repeating the process. The delay is achieved using the HAL_Delay() function, which is a waiting function that essentially delays the program for a specified time.

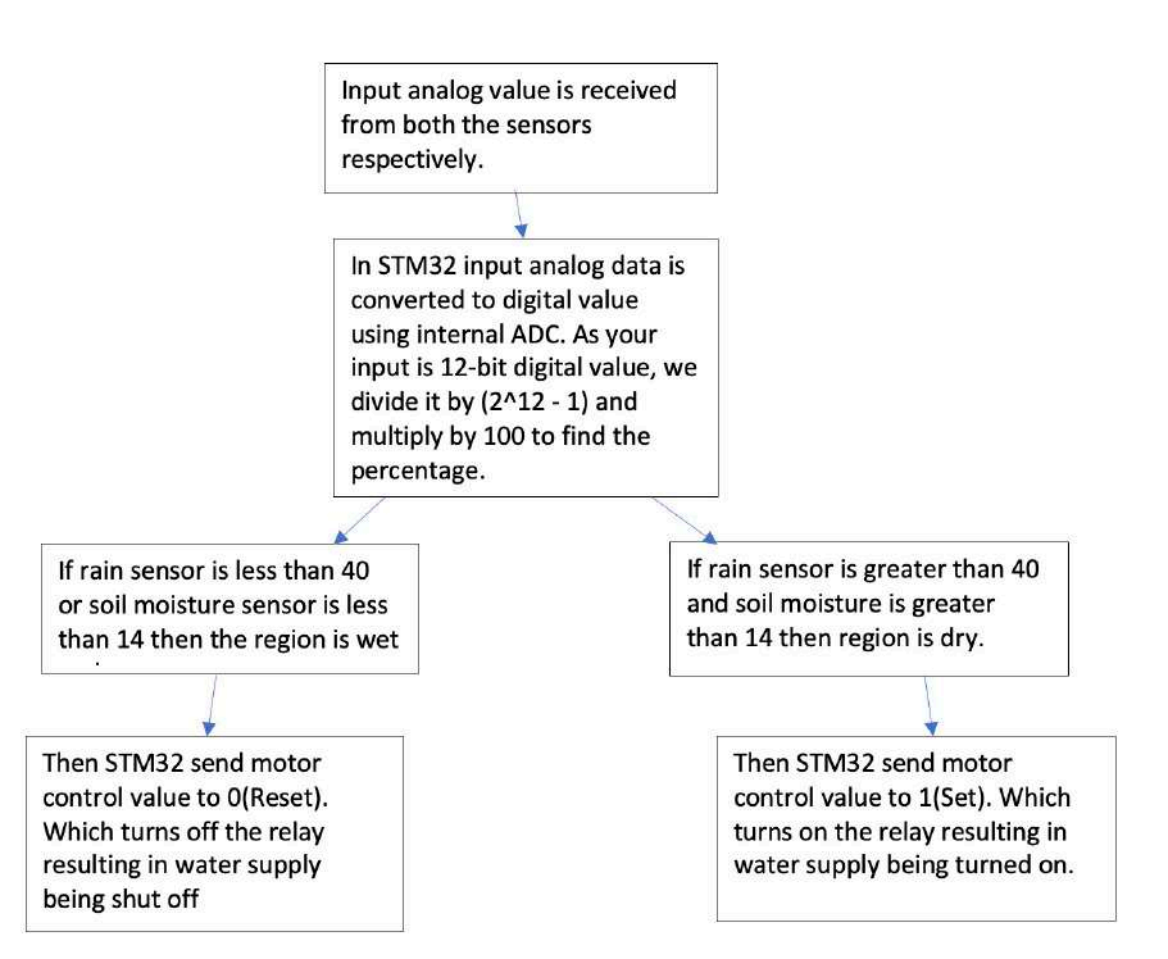
The program uses two ADCs, hadc1 and hadc2, to read the analog signals from the sensors. The GPIO pins PB6 and PA1 are used to control the motor & buzzer respectively.

The motor is controlled via a relay. This relay works on the principle of induction. The control pin on the relay receives a high or low signal which turns the motor on & off respectively.

The HAL_GPIO_WritePin() function is used to set the pins to high or low.

Overall, we are reading analog signals and use them to control a motor (for certain specified conditions) using a microcontroller.

Flow of code


Input analog value is received from both the sensors respectively.
In STM32 input analog data is converted to digital value using internal ADC. As your input is 12-bit digital value, we divide it by (2^12 - 1) and multiply by 100 to find the percentage.
If rain sensor is less than 40 or soil moisture sensor is less than 14 then the region is wet
If rain sensor is greater than 40 and soil moisture is greater than 14 then region is dry.
Then STM32 send motor control value to 0(Reset). Which turns off the relay resulting in water supply being shut off
Then STM32 send motor control value to 1(Set). Which turns on the relay resulting in water supply being turned on.

Code Snippets and IOC

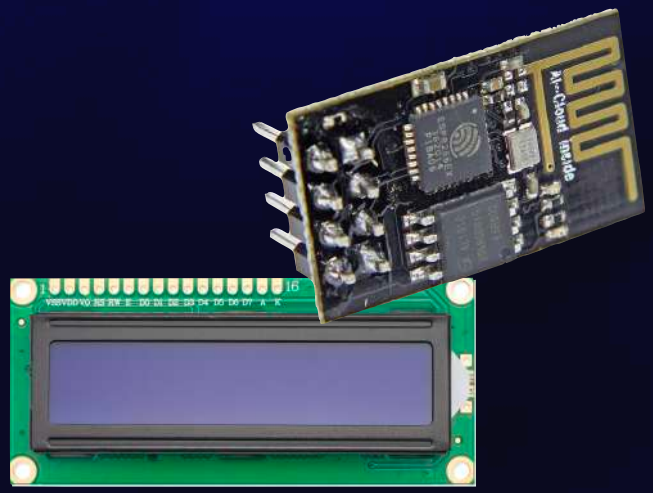
```
// Start ADC conversion
    HAL_ADC_Start(&hadc1);                    // Poll for conversion completion
    HAL_ADC_PollForConversion(&hadc1, 10); // Read the ADC value
    soil_moisture_adc_value = HAL_ADC_GetValue(&hadc1); // Convert the ADC value to a percentage
    soil_moisture_percentage = (float)soil_moisture_adc_value / 4095.0 * 100.0;

    HAL_ADC_Start(&hadc2);                     // Poll for conversion completion
    HAL_ADC_PollForConversion(&hadc2, 10);  // Read the ADC value
    rain_adc_value = HAL_ADC_GetValue(&hadc2);  // Convert the ADC value to a percentage
    rain_percentage = (float)rain_adc_value / 4095.0 * 100.0;

// Final motor control value
    if( soil_moisture_percentage < 14 || rain_percentage < 40 )
    {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);


    }
    else{
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
```

Working Demo

Future Iterations and fallbacks

One of the ways we see this improving in the future is through the use of stepper motors. Right now the setup doesn't offer the flexibility to change the amount of water being released. Since this required constantly turning off/ on the motor we did not implement this. But by using a stepper motor we can make it a full fledged drip irrigation system greatly increasing the efficiency.

Wi-Fi Module



LCD Display

The whole system will be practically very effective when we introduce a Wi-Fi module which relays the relevant info into a web server and also allow the user to turn the motor on/off as well as receive relevant field data.

One fallback we had was that we intended to include the LCD Module but due to issues it couldn't be integrated with the system in time. Although we learnt a lot, we can replace the LCD Display use case by utilizing the web server and the Wi-FI Module.


THANK
YOU