**Smart Plant Pot Logbook - ky97**

**Objective:** Design and create a smart plant pot using Firebeetle esp32 to monitor a plant's health and automate the watering

**WEEK 1 (30/10/2023): Researching and Planning**

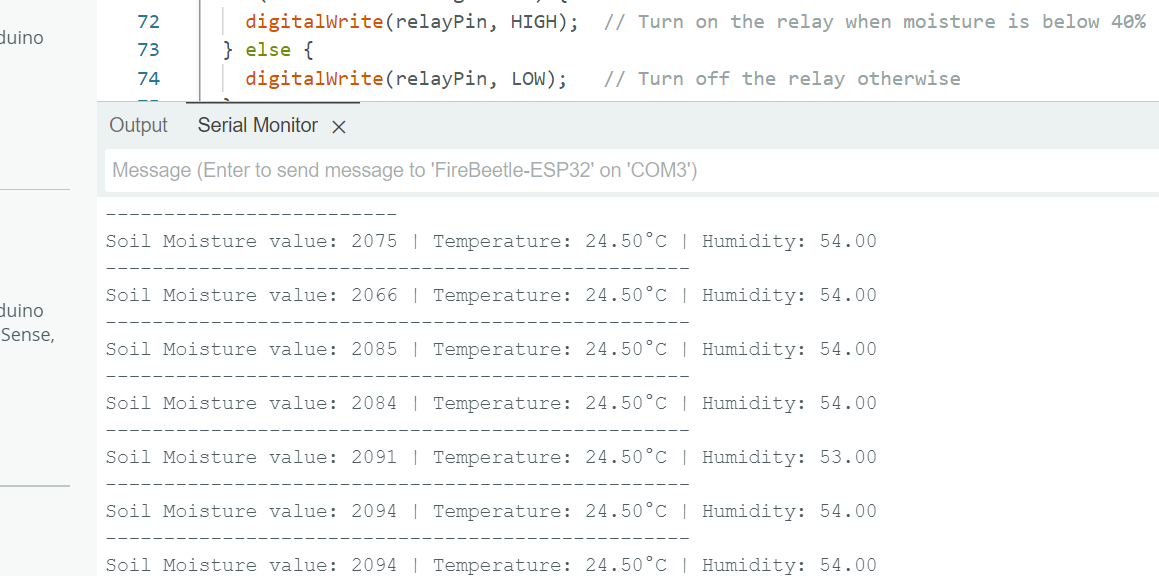
* Researched existing smart plant pots and similar projects for inspiration.
* Researched power-efficient communication methods to ensure minimal energy consumption during wireless data transmission.
* Set weekly goals to achieve with the project.
* Selected Firebeetle ESP32 as the main microcontroller for its capabilities, built-in Wi-Fi support, and compatibility with my laptop.
* Outlined the components needed: soil moisture sensor, temperature and humidity sensor, water pump, and LCD display.
* Regular planned testing after every implemented feature.

**WEEK 2 (06/11/2023): Gathering Materials**

* Created a list of required materials and components on a Word document.
* Ordered a Firebeetle ESP32 development board from HACKSPACE
* Ordered the sensors from Amazon. Soil moisture sensor and DHT11 sensor.
* Ordered other components. Water pump, LCD Display, relay module and breadboard
* Ordered external power supplies
* Checked existing supplies for additional materials (pots, tubing, etc.).

**WEEK 3 (13/11/2023): Setting Up Development Environment, Documentation and Reflection**

* Set up Arduino IDE with Firebeetle ESP32 board support.
* Installed necessary libraries. Sensor libraries, Blynk app, LCD Display and Wifi connectivity.
* Tested basic code snippets for each sensor to ensure compatibility.
* Documented the final circuit diagram, code, and assembly instructions.
* Reflect on the challenges faced, solutions found, and lessons learned throughout the project.
* Prepared for potential future improvements and enhancements.



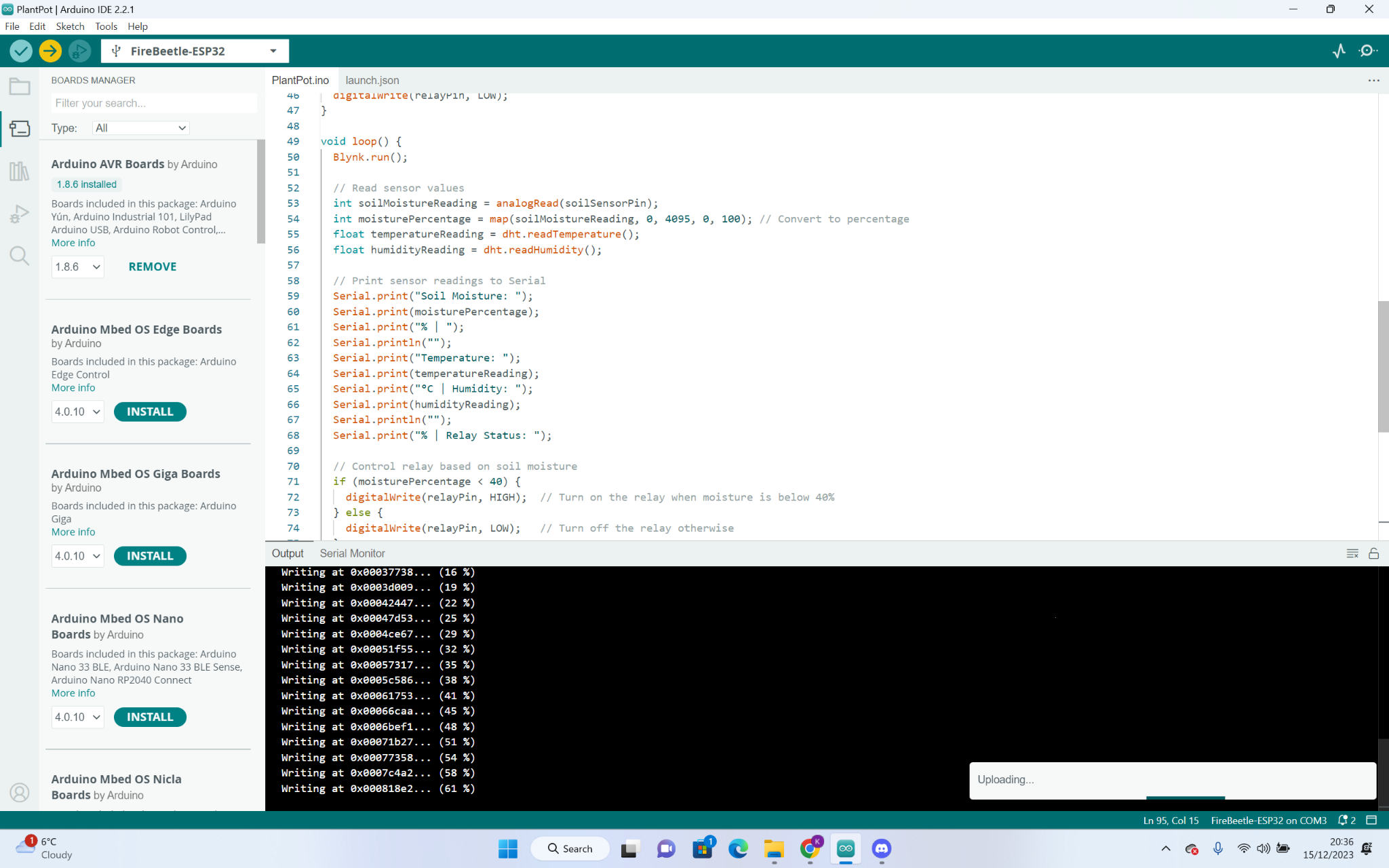
**WEEK 4 (20/11/2023): Assembling Hardware**

* Received and inspected components.
* Made sure components were working with the ESP 32 device.
* Assembled the circuit on a breadboard, connecting sensors and actuators to the Firebeetle ESP32.
* Checked jumper wires and made necessary adjustments.
* Made sure there were no faulty pins on the Esp 32 device.

**WEEK 5 (27/11/2023): Component Calibration and Integration**

* Connected soil moisture and DHT11 sensor to GND, VCC and appropriate signal pins
* Connected LCD Display to GND, VCC, SDA and SCL pins
* Adjusted sensor calibration values in the code.
* Connected relay module to NO (Normally open) and COM (Common)
* Integrated all components into the plant pot.
* Secured the components to ensure durability and stability.
* Conducted final tests to ensure all systems were functioning correctly.

**WEEK 6 (04/12/2023): Developing Software, Testing, Debugging and Data Logging**

* Imported necessary libraries for components and connections.
* Added debugging and testing code throughout.
* Added Wi-Fi connectivity for remote monitoring and control through the Blynk app and serial monitor
* Initialised the components in the setup function of Arduino
* Displayed values and readings in the loop function of Arduino
* Identified and fixed bugs in the code.
* Fine-tuned control thresholds for watering.
* Implemented data logging to record sensor readings over time.
* Set up the Blynk app to show users when the soil moisture level is low or other conditions are detected.
* 

**– How I approached this piece of work: ....**

Initially, I had to figure out what device I needed for my laptop and checked reviews online. Then, I had to set up my Arduino development environment and made sure that the device was connecting successfully. Once the prerequisites were out of the way, I first connected the soil moisture sensor and ensured it gave correct readings when the soil was dry and wet. I used the serial monitor for a lot of debugging. Connecting the other sensors became much easier to do when I completed the soil sensor correctly. Once I started to add the other components, e.g. temperature humidity sensor, relay module LCD display, etc, I realised I needed a breadboard to make the common connections with the GND and VCC with the esp device and other components since there was not enough pins on the ESP 32 device.

**What I found fairly easy was ....**

What I found fairly easy was using the serial monitor on the Arduino development environment to debug a lot of my work. This helped me pinpoint exactly what part of the code was causing issues. For example, I could see if the relay module and sensors are correctly working by displaying the status and values on the serial monitor.

**– What I found most difficult was ....**

What I found most difficult was using the breadboard. Firstly, figuring out how to connect these devices took up much of my time. Then, I had to figure out the different features of the breadboard (e.g. GND pin for the negative rail) so I could create multiple circuits for multiple sensors/components.

**– If I were to do the work again, I would do the following differently: ....**

If I were to start over, I would set up all the circuits correctly with the breadboard and make sure the jumper wires are organised neatly as well so any sudden movements of the devices don’t displace the wires.

**– What I learned is …**

When moving on to the automated watering part of the project, I realised that connecting the water pump and relay module to the breadboard was causing signal issues with the other sensors. For example, the soil sensors were showing higher values than usual. After research, the relay module and water pump needed an external power supply and circuit.

**– I would like specific feedback on …**

Specific feedback I would love is my connections with the jumper wires and project components. I would love to know how to make the product as simple and efficient as possible. This is because if this became an actual product, the design of the plant pot would contain the wires inside, so having the least number of connections would be beneficial.

**– I felt this way about the work ....**

While learning Arduino and the “Internet of Things” course while doing this project, looking back, I realised how much more I could have done. Since a lot of time was spent researching and understanding the course, this meant there was less time implementing features and coming up with designs.

**CODE**

// Wi-Fi library for connecting to a wireless network

#include <WiFi.h>

#include <WiFiClient.h>

// Library for controlling I2C-based LCD displays

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Library for HD44780-compatible LCDs

#include <hd44780.h>

#include <hd44780ioClass/hd44780\_I2Cexp.h>

// Library for interfacing with DHT sensors

#include <DHT.h>

// Blynk app template details

#define BLYNK\_TEMPLATE\_ID "TMPL5Scu-z8wi"

#define BLYNK\_TEMPLATE\_NAME "Quickstart Template"

#define BLYNK\_AUTH\_TOKEN "eiIL-ZpBVYsZZdvM44AGCsxhBcTToHB8"

#include <BlynkSimpleEsp32.h>

// Wi-Fi Login

char ssid[] = "KentTest213";

char pass[] = "12345666";

// LCD config

hd44780\_I2Cexp lcd;

// Pin definitions

const int soilSensorPin = A0; // Analog pin connected to SIG pin of the soil sensor

const int dhtPin = 14; // Replace with the GPIO pin your DHT sensor is connected to

const int relayPin = 18; // Replace with the GPIO pin your relay is connected to

// DHT sensor setup

DHT dht(dhtPin, DHT11);

void setup() {

// Initialization serial communication

Serial.begin(115200);

lcd.begin(16, 2);

dht.begin();

// Connect to Wi-Fi and Blynk

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

// Set relay pin as output and turn it off initially

pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, LOW);

}

void loop() {

Blynk.run();

// Read sensor values

int soilMoistureReading = analogRead(soilSensorPin);

int moisturePercentage = map(soilMoistureReading, 0, 4095, 0, 100); // Convert to percentage

float temperatureReading = dht.readTemperature();

float humidityReading = dht.readHumidity();

// Print sensor readings to Serial

Serial.print("Soil Moisture: ");

Serial.print(moisturePercentage);

Serial.print("% | ");

Serial.println("");

Serial.print("Temperature: ");

Serial.print(temperatureReading);

Serial.print("°C | Humidity: ");

Serial.print(humidityReading);

Serial.println("");

Serial.print("% | Relay Status: ");

// Control relay based on soil moisture

if (moisturePercentage < 40) {

digitalWrite(relayPin, HIGH); // Turn on the relay when moisture is below 40%

} else {

digitalWrite(relayPin, LOW); // Turn off the relay otherwise

}

// Combine sensor values for Blynk visualization

String combinedSensorValues = "Moisture: " + String(moisturePercentage) + "%\nTemp: " + String(temperatureReading) + "°C\nHum: " + String(humidityReading) + "%";

Blynk.virtualWrite(V3, combinedSensorValues);

// Update LCD displays

lcd.print("Moisture Level: ");

lcd.print(moisturePercentage);

lcd.print("%");

delay(2000);

lcd.clear();

lcd.print("Humidity: ");

lcd.print(humidityReading);

delay(2000);

lcd.clear();

lcd.print("Temperature: ");

lcd.setCursor(0, 1);

lcd.print(temperatureReading);

lcd.print("C ");

delay(2000);

lcd.clear();

}