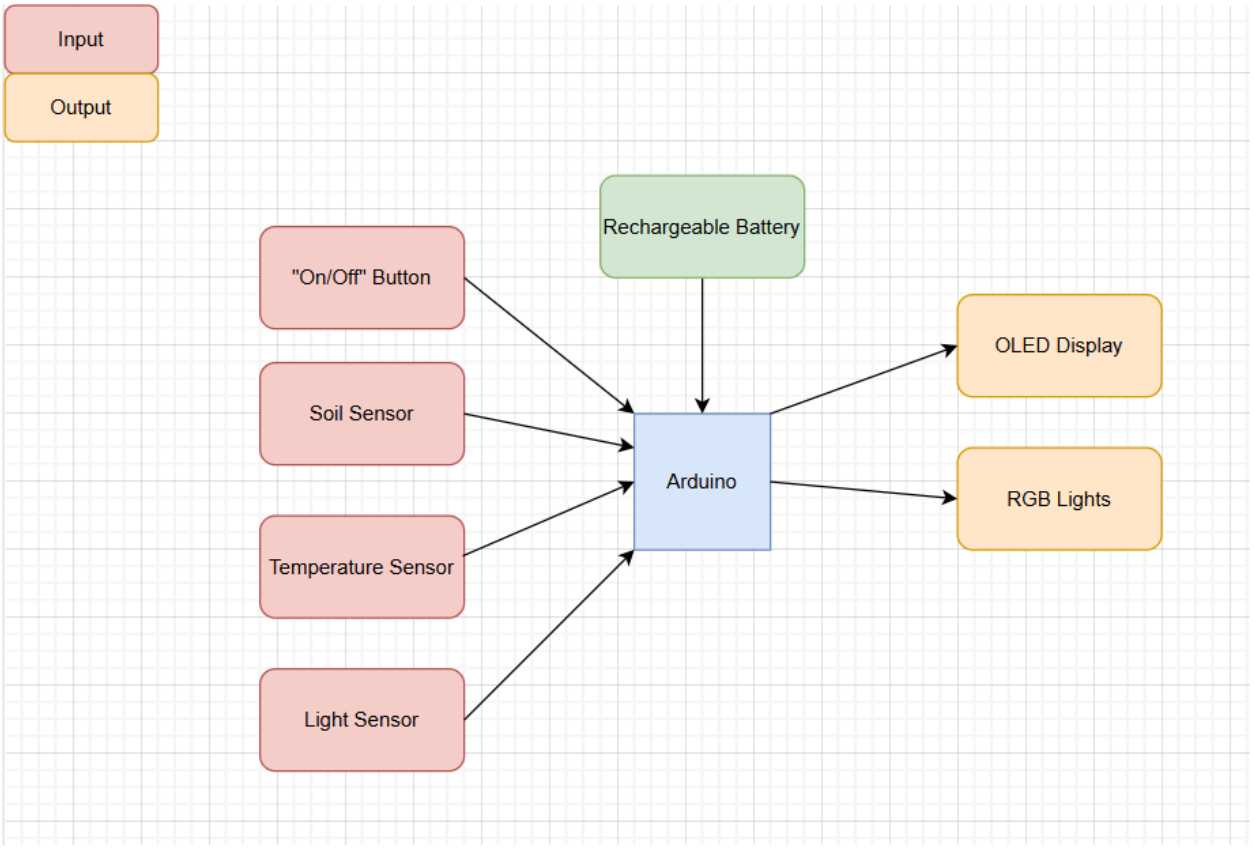
ECET 430 Project Logbook
Project: Smart Plant Care
Group Members: Natwarin Padtha, Jonathan Jou
Github repository link: https://github.com/jonjonjou/Smart-Plant-Care

Major moving parts:

- Sensors: Soil moisture, light intensity, temperature, and humidity sensors continuously collect data.
- Battery & Charging System: This keeps the device powered on for extensive usage.
- Wireless Communication Module: This allows for Bluetooth/Wi-Fi connectivity.

Sections:
- Block Diagram



Priority items:

- Accuracy: It ensures reliable plant monitoring
- Battery Life & Power Management: Keeps the device running for extended periods
- Electrical Hazard: Avoid any safety risks
- Water Resistance: Prevent the environmental exposure

Dependant tasks:
What will require additional research? What needs to be completed before something else can be started? What feels unclear or not obvious?

- The appropriate soil moisture, humidity, and light intensity ranges for different types of plants
- At what moisture level should the device trigger a watering alert?
- How do environmental conditions affect sensor accuracy?
- The best microprocessor that balances power efficiency, processing capability, and connectivity
- How will the design prevent electrical hazards, such as short circuits from moisture exposure?

Architecture:
Options to implement the project:
I chose a Li-ion 14500 battery or a Lithium Polymer (LiPo) 3.7V battery because they are both 3.7V and can output 5V with a voltage regulator, which is what an Arduino, sensor, and display need. Secondly, they are small enough to fit into the smart plant device.

List major components:

- Microcontroller: Arduino UNO
- Sensors: DHT11 Temperature and Humidity Module, Digital LDR Photosensitive Light Sensor Module,LM393 3.3V-5V Soil Moisture Detect Sensor Soil Moisture
- Actuators:None, the device is stationary.
- User interaction:
    - User places the device in a plant pot.
    - The sensors collect environmental data.
    - The output alerts the user about the plant's needs, e.g., "Water needed" or "Too much light."
- Interfaces: OLED screen and Mobile Application
- Anything external: None at the moment

- Specific to this design: None
- Hard to get or expensive items: The most expensive item for this project is the Arduino.
- Others components: None

Write goals for three possible prototypes

- Temperature sensor
- Soil sensor
- Light sensor

Arduino code

```
#include <dht.h>
#include <LiquidCrystal.h>

// Sensor and LED Pins
#define outPin 8       // DHT11 sensor
#define DO_PIN 13      // Light sensor (digital)
int sensorPin = A0;    // Moisture sensor (analog)

// RGB LED Pins
const int redPin = 6;
const int greenPin = 9;
const int bluePin = 10;

// LCD Pins: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Temperature thresholds (in Fahrenheit)
const float minTempThresholdF = 70.0;
const float maxTempThresholdF = 80.0;

dht DHT;  // DHT11 sensor object

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("System Starting...");
```

```
  delay(2000);
  lcd.clear();

  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(DO_PIN, INPUT);
  pinMode(sensorPin, INPUT);
}

void loop() {
  int readData = DHT.read11(outPin);
  float temperatureF = (DHT.temperature * 9.0) / 5.0 + 32.0;
  float humidity = DHT.humidity;
  bool lightGood = checkLightSensor();
  int moistureStatus = checkMoistureSensor();

  bool temperatureGood = (temperatureF >= minTempThresholdF &&
temperatureF <= maxTempThresholdF);
  bool moistureGood = (moistureStatus == 0);
  bool allGood = temperatureGood && lightGood && moistureGood;

  // Display sensor values
  Serial.println("============================");
  Serial.print("Temp = ");
  Serial.print(DHT.temperature);
  Serial.print("C | ");
  Serial.print(temperatureF);
  Serial.println("F");
  Serial.print("Humidity = ");
  Serial.print(humidity);
  Serial.println("%");
  Serial.println(lightGood ? "Light Present" : "No Light Detected");
  Serial.println(moistureStatus == -1 ? "Soil is Dry" : (moistureStatus
== 1 ? "Soil is Too Wet" : "Soil Moisture is Good"));
  Serial.println("============================");

  // LCD first row = Temp & Humidity
  lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperatureF, 1);
  lcd.print("F H:");
  lcd.print(humidity, 0);
  lcd.print("%");

  if (allGood) {
    setGreen();
    lcd.setCursor(0, 1);
    lcd.print("All Good");
  } else {
    if (!temperatureGood) {
      flashRed();
    }
    if (!lightGood) {
      flashBlue();
    }
    if (!moistureGood && moistureStatus == -1) {
      flashPurple();
    }
  }

  delay(2000);
}

bool checkLightSensor() {
  int lightState = digitalRead(DO_PIN);
  return (lightState == LOW);  // LOW = light detected
}

int checkMoistureSensor() {
  int sensorValue = analogRead(sensorPin);
  sensorValue = map(sensorValue, 0, 1024, 255, 0);
  if (sensorValue < 150) return -1;  // Dry
  else if (sensorValue > 200) return 1;  // Too wet
  else return 0;  // Good
}
```

```
void flashRed() {
  lcd.setCursor(0, 1);
  lcd.print("Temp Low");
  for (int i = 0; i < 3; i++) {
    setRGB(HIGH, LOW, LOW);  // Red
    delay(300);
    setRGB(LOW, LOW, LOW);
    delay(300);
  }
}

void flashBlue() {
  lcd.setCursor(0, 1);
  lcd.print("No Light       ");  // Pad with spaces to clear previous
msg
  for (int i = 0; i < 3; i++) {
    setRGB(LOW, LOW, HIGH);  // Blue
    delay(300);
    setRGB(LOW, LOW, LOW);
    delay(300);
  }
}


void flashPurple() {
  lcd.setCursor(0, 1);
  lcd.print("Soil Dry");
  for (int i = 0; i < 3; i++) {
    setRGB(HIGH, LOW, HIGH);  // Purple
    delay(300);
    setRGB(LOW, LOW, LOW);
    delay(300);
  }
}

void setGreen() {
  setRGB(LOW, HIGH, LOW);  // Green
  Serial.println("Everything is good! Green Light ON.");
}
```

```
void setRGB(int r, int g, int b) {
  digitalWrite(redPin, r);
  digitalWrite(greenPin, g);
  digitalWrite(bluePin, b);
}
```

Tasks Completed	Results / Notes
Brainstormed project ideas, selected "Smart Plant Care" as concept	Finalized project scope and key features (sensors, display, alerts)
Built a basic breadboard circuit with Arduino Uno and sensors	All sensors responded to input; OLED not yet integrated
Wrote the first version of Arduino code: read sensors and print to Serial Monitor	The output showed consistent soil moisture and temperature readings
Added RGB LED functionality and logic for color-coded feedback	RGB LED worked correctly; green showed optimal conditions, red for dry soil
Integrated OLED display and attempted I2C communication	Encountered inconsistent display issues; I2C address and wiring were double-checked
Started KiCad schematic and component footprint placement	Replaced Arduino Uno symbol with ATmega328P and supporting components
Finished KiCad PCB design and passed	Gerber files generated; the board was ready

ERC/DRC checks	for fabrication
3D modeling and printing of a vase enclosure	Fit confirmed; adjustments made to sensor slot positions
Performed individual testing of sensors in varied conditions	Moisture, temperature, and light detection passed test cases
Attempted buzzer integration and full prototype assembly	Could not test buzzer due to lack of available pins
Final code tuning, data logging via Serial Monitor, created repository	Sensor thresholds fine-tuned; GitHub repository updated
Presentation preparation, documentation draft, final testing	Final build working, but OLED display remained unreliable in some tests

Things That Did Not Work
- The OLED display had intermittent failures, possibly due to I2C address conflicts or unstable wiring.
- The buzzer could not be tested in the final prototype due to insufficient digital I/O pins.
- The display readability and update timing needed improvement for a better user experience.

Lessons Learned
- Start prototyping earlier to allow more time for debugging hardware interactions (e.g., OLED).
- Pin limitations on Arduino Uno can restrict functionality; future builds should consider microcontrollers with more GPIO pins or use I/O expanders.
- Proper sensor placement and wiring are crucial to ensure accurate readings and system stability.
- KiCad's rule checks are extremely helpful in preventing costly PCB errors and ensuring manufacturability.
- Power supply planning early in the design phase simplifies later integration and improves reliability.