اونيۏرسيتي مليسيا ڤهڠ السلطان عبدالله

UNIVERSITI MALAYSIA PAHANG

¡AL-SULTAN ABDULLAH

FAKULTI TEKNOLOGI KEJURUTERAAN ELEKTRIK DAN ELEKTRONIK

PROJECT: SMART PLANT MONITORING SYSTEM USING TXT FUNCTION

BVI1142 TECHNOLOGY SYSTEM PROGRAMMING I

BIL	NO ID	NAME
1	VC23054	AHMAD SYAHIDUL AMIN BIN MOHAMAD AZNAL
2	VC23053	MUHAMMAD AMIRUL ADLY BIN MOHAMAD RASHIDI
3	VC23047	SYAMSUL ARIFIN BIN AWI BOWO
4	VC23052	MOHAMAD AQMAR HARITH BIN KAMAROLZAMAN

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROJECT BACKGROUND

The project aims on collect and analysis growth and health parameters data of plants like temperature, humidity, volumetric water and light intensity. By using a text file function on c programming, the smart plant monitoring records all real-life time data like I mentions the parameter earlier. This system ensures the accuracy and timely recording data allowing the user to do a quick intervention like adjusting watering or light conditions. User can access and review all the information by the data collected in a text file. Automated alerts bring a significant role on notify users of critical changes while data analysis provide and improving strategies in the future. As these conditions meet, the system should be a better plant health and productivity monitoring system.

1.2 PROBLEM STATEMENT

In these past few years, plant care has become one of the most challenging things due to time constraints and limited knowledge about the optimal environmental conditions for plant growth. This has been worst when traditional methods of monitoring plants require significant attention and are often led to errors, such as overwatering or underexposure to light.

Moreover, most of the existing monitoring systems nowadays lack real-time data logging and visualization. The decision about plant health is crucial eventually when making timely and informed decisions. These systems often fail to integrate multiple environmental factors such as temperature, humidity, soil moisture, and light intensity. This project aims to develop a real-time data monitoring system which is Smart Plant Monitoring System that captures data like temperature, soil moisture, and light intensity with various type of sensors while the data being recorded and stored in a text file for analysis.

1.3 OBJECTIVE

- To design a monitoring environmental system that are cost-effective for plant growth and health.
- To develop an automatic monitoring system to saving more time and more organized.
- To provide real time data logging of light intensity, temperature, humidity, and soil moisture levels of plant to enable user in decision making for plant growth.

1.4 SCOPE & LIMITATION PROJECT

1.4.1 DATA MONITORING

The project captures the data environmental using:

- Temperature using DHT 11 sensor
- Soil moisture level using the soil moisture sensor
- Lighting intensity using the LDR sensor

1.4.2 COLLECTING DATA AND STREAMING

- F function to analyze and review the recorded data in txt file.
- Basic analysis of trends in environmental factors can be recorded by reviewing the stored text file data (.txt).
- For collecting and processing data, we use an Arduino Mega which is then saved into the text file for easy access and further analysis.

1.4.3 SENSOR ACCURACY

- Limited precision on accuracy because of specification and cheap price also be the potential sensors like DHT 11, soil moisture, and LDR that has limited precision.
- Extreme temperature of humidity can be the biggest effect on sensor performance.

1.4.4 RANGE AND SCALABILITY

- The system's monitoring range is limited by the physical range of the sensors and the Arduino setup.
- Limitations when handling large volume of data or complex data can trigger the text file system storage.

CHAPTER 2

LITERATURE

2.1 CONVENTIONAL METHODS ON MANUAL OBSERVATIONS

Conventional methods have already proven that limited knowledge and constrains of times has become a challenge on keeping the plant care. This is because it requires a significant attention. This often leads in errors such as overwatering and insufficient light exposure. Studies has proven that this traditional system lacking on providing a real-time data integration and visualization. This often leading to delays and ineffective decisions regarding plant health.

Moreover, environmental factors like temperature, humidity, soil moisture and light intensity has become one of the most critical items in monitoring plant care health. This project addresses these gaps on develop Smart Plant Monitoring System that captures real-time data through sensors for temperature, soil moisture and light intensity. The system records data on text file for efficient logging and analysis data without needing a third-party application only by using your laptop or computer.

2.2 COMPONENTS

This is a short description of component used in this project such as Arduino Mega 2560, Soil moisture, DHT11 and LDR sensors.

2.2.1 ARDUINO MEGA 2560

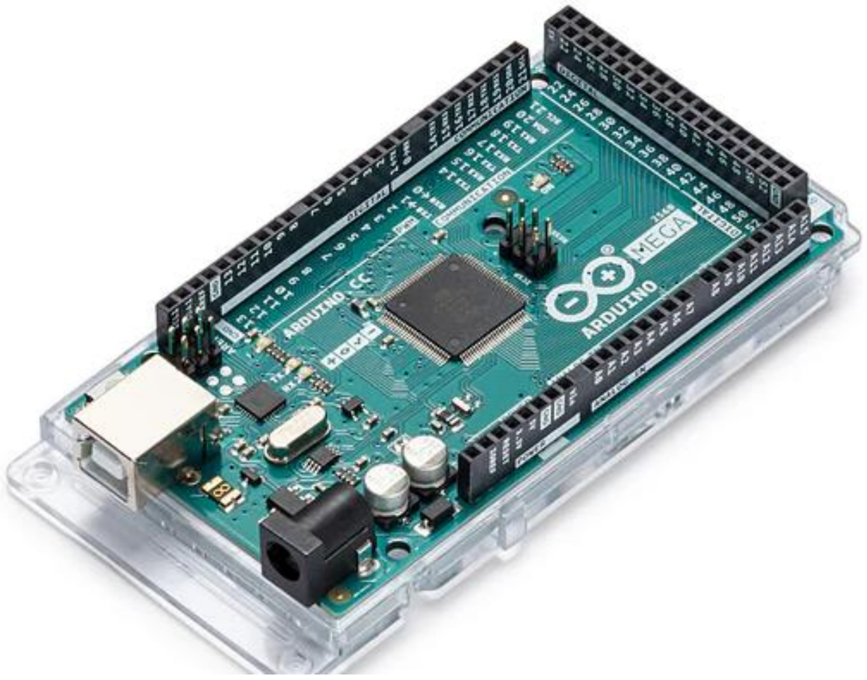


Figure 2.1 Arduino Mega 2560

The Arduino Mega is a microcontroller board based on. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila. The Arduino Mega is used to interfaces with various sensors like (DHT11, LDR, soil moisture sensor, etc.) to collect real-time data on environmental factors like temperature, humidity, light intensity, and soil moisture.

Specifications

Table 2.1 Specifications of Arduino Mega 2560

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Pin I/O Diagram

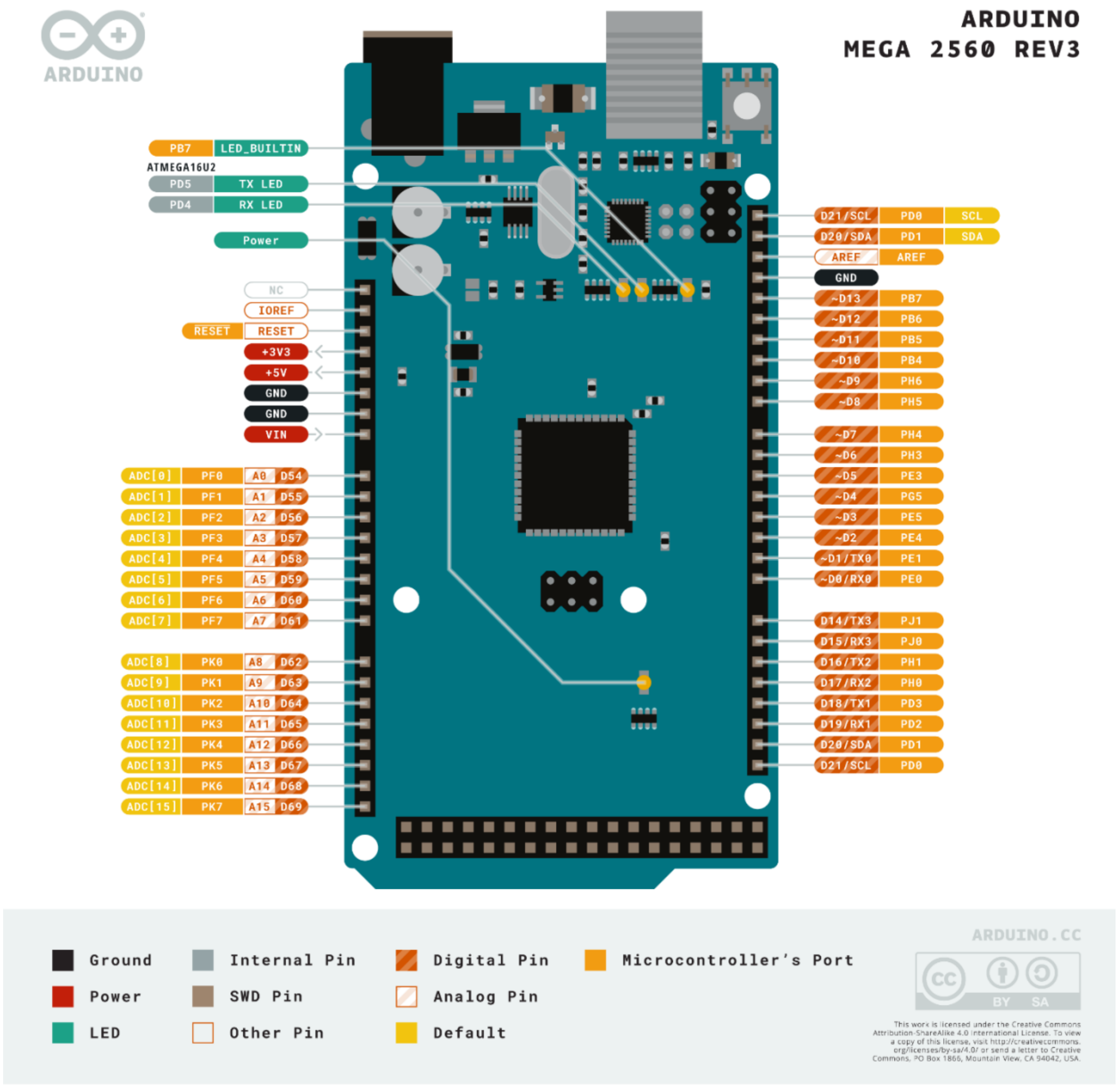


Figure 2.2 Arduino Mega Pin I/O Diagram

2.2.2 SOIL MOISTURE SENSOR

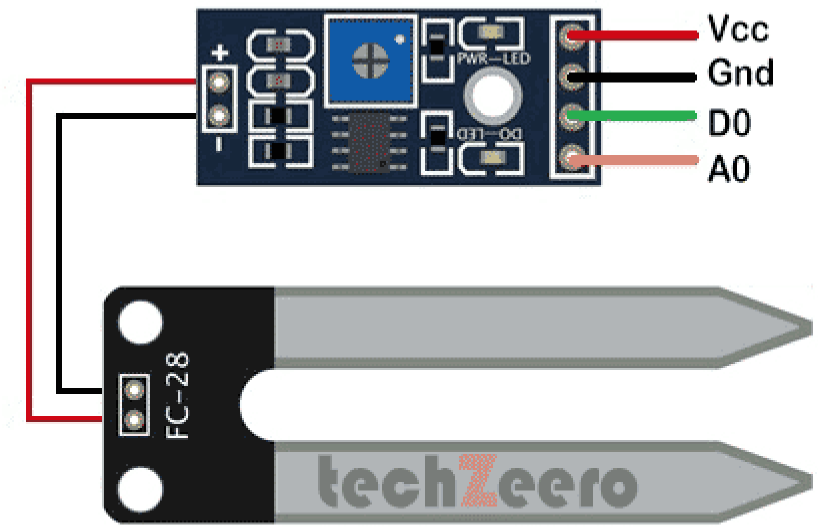


Figure 2.3 Soil Moisture Sensor

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting. These sensors measure the volumetric water content not directly with the help of some other rules of soil like dielectric constant, electrical resistance, otherwise interaction with neutrons, and replacement of the moisture content.

The relation among the calculated property as well as moisture of soil should be adjusted & may change based on ecological factors like temperature, type of soil, otherwise electric conductivity. The microwave emission which is reflected can be influenced by the moisture of soil as well as mainly used in agriculture and remote sensing within hydrology.

Specifications

Table 2.2 Specifications of Soil Moisture Sensor

Input Voltage	5V
Input Current	<20mA
Type of interface	Analog
Board Size	3.2 cm x 1.4 cm
Working Temperature	10°C~30°C

Soil moisture to Arduino Mega Connection

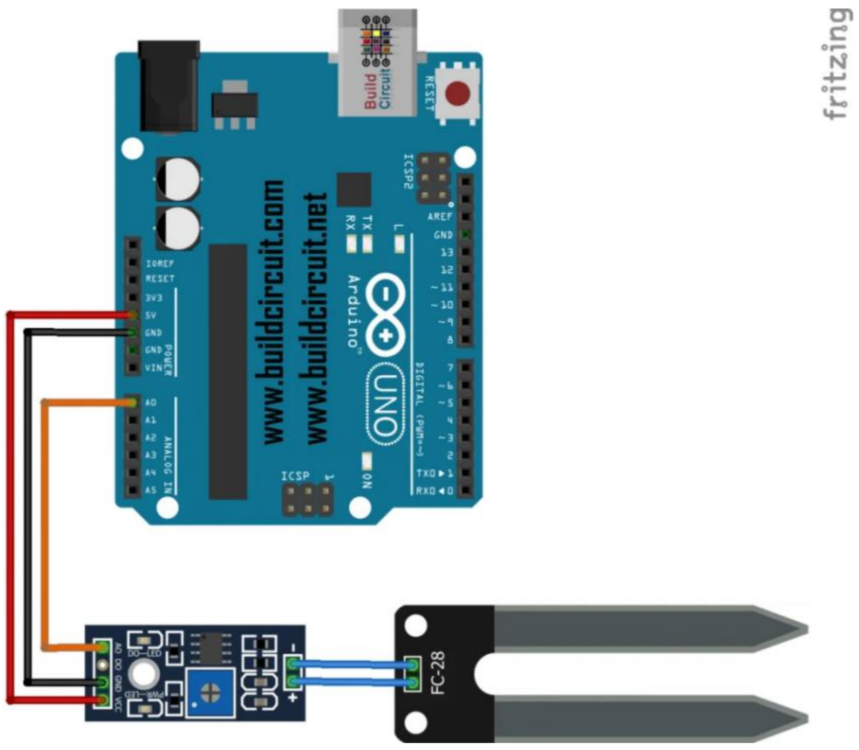


Figure 2.4 Example of connection between Soil Moisture to Arduino Mega

Table 2.3 Soil Moisture connection to Arduino Mega

Soil Moisture	Arduino Mega
Vcc	5V
Gnd	Gnd
DO	-
AO	A0

2.2.3 SENSOR DHT 11

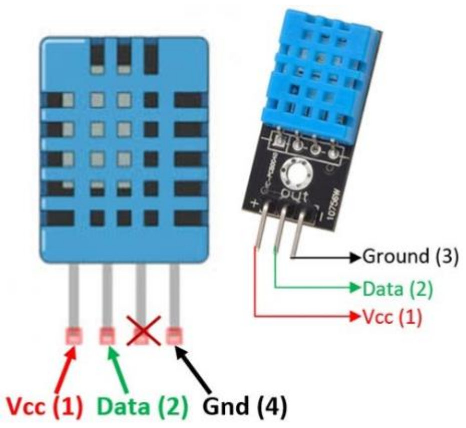


Figure 2.5 Example of DHT 11 and pin DHT11

DHT11 is the most common sensor that is usually used by students and makers out there. The main function of this sensor is to measure the temperature and humidity of the surroundings. The sensor is also factory calibrated and hence easy to interface with any microcontrollers like Arduino. The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. The DHT11 sensor can either be purchased as a sensor or as a module. Either way, the performance of the sensor is the same. The sensor will come as a 4-pin package out of which only three pins will be used whereas the module will come with three pins as shown below. The only difference between the sensor and module is that the module will have a filtering capacitor and pull-up resistor inbuilt.

DHT 11 Adopt special digital module acquisition technology and temperature and humidity sensing technology to ensure high reliability and excellent long-term stability. The sensor is composed of a resistance liquid contact element and an NTC temperature measuring element and is connected to a high-performance 8-bit microcontroller. Therefore, the product has the advantages of high quality, ultra-fast response, strong anti-interference ability, and high-cost performance. The single-wire serial interface makes system integration simple and fast. Ultra-small size, low power consumption, and a signal transmission distance of more than 20 meters make it the best choice for any application, even the most demanding ones. The product is easy to connect and can be plugged directly into the sensor expansion board.

Specification

Table 2.4 Specification of DHT11

Operating Voltage	3.0V to 5.5V
Operating Current	0.3mA (measuring)
Output	Serial data
Temperature Range	0°C to 50°C
Humidity Range	20% to 90%
Resolution	Temperature and Humidity both are 16-bit
Accuracy	±1°C and ±1%

DHT11 to Arduino Mega Connection

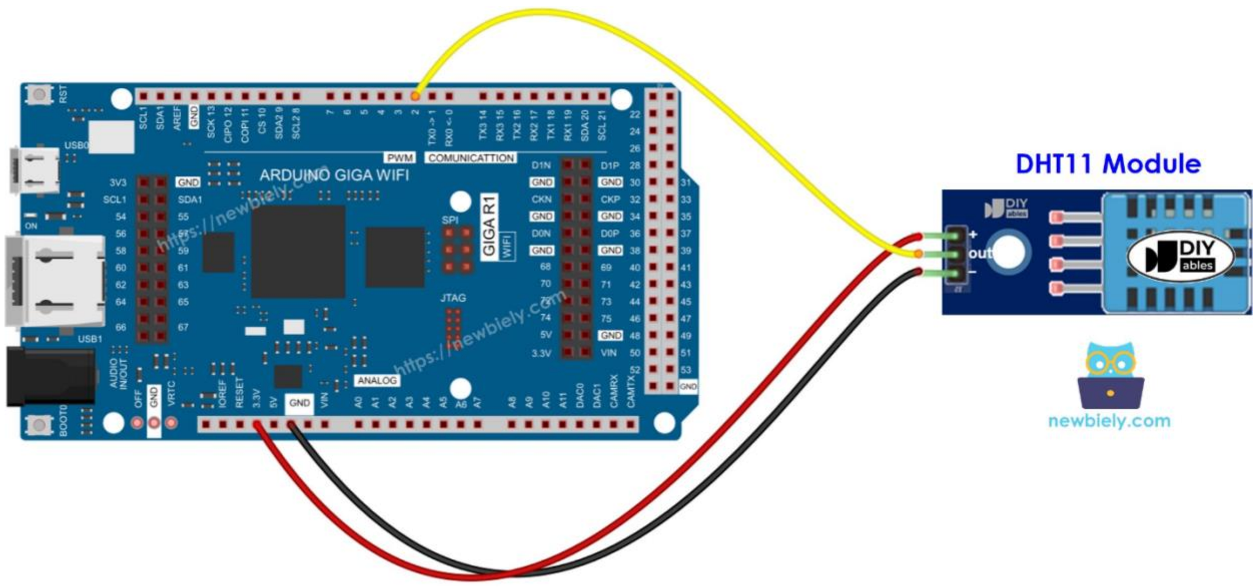


Figure 2.6 Example connection between Arduino Mega and DHT11

Table 2.5 DHT11 Module connection to Arduino Mega

DHT 11	Arduino Mega
VCC	5V
GND	GND
OUT	D2

2.2.4 LDR SENSOR

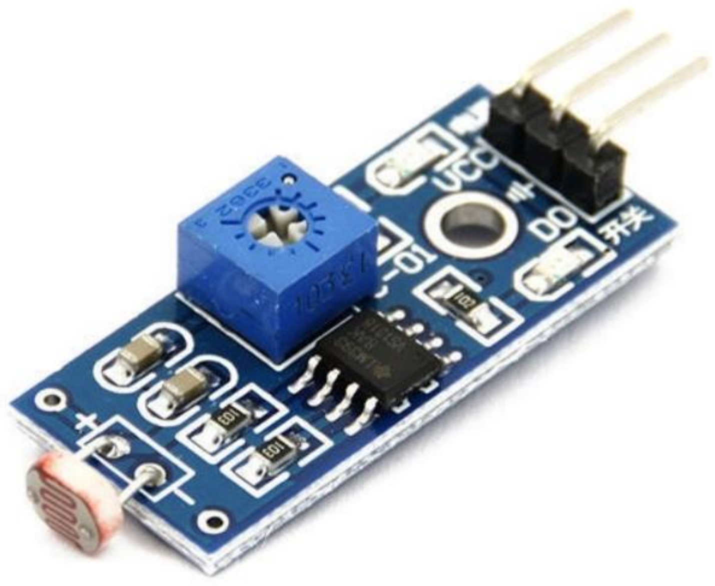


Figure 2.7 LDR Sensor

LDR full form Light Dependent Resistor. LDR is a simple device that can be used to detect light level and respond to light. When there's more light, its resistance goes down, and when there's less light, its resistance goes up. LDR light sensor is also known as a photoresistor, photocell, or photoconductor. In the dark, it has a very high resistance, sometimes up to 1M ohm, but when the LDR sensor is exposed to light, the resistance drops drastically, to as low as a few ohms, depending on the light intensity. The sensitivity of LDR varies with the wavelength of the light applied and are nonlinear device.

Light Dependents Resistor or LDR is a transducer that senses light intensity and converts it into electric current. It has two electrodes, one of which acts as the cathodes while the other acts anode. The resistance of LDR decreases when exposed to a certain level or amount of radiant energy such as sunlight and artificial light. This property makes them suitable for use in applications like solar panel controller, security system, and variable lighting controls by changing their value according to variations in ambient illumination conditions.

Specification

Table 2.6 Specification LDR sensor

Operating Voltage	3.3V to 5.5V
Operating Current	15mA
Output Digital	0V to 5V, adjustable trigger level from preset
Output Analog	0V to 5V based on light falling on the LDR
Size	33mm x 15mm x 2mm

LDR to Arduino Mega Connection

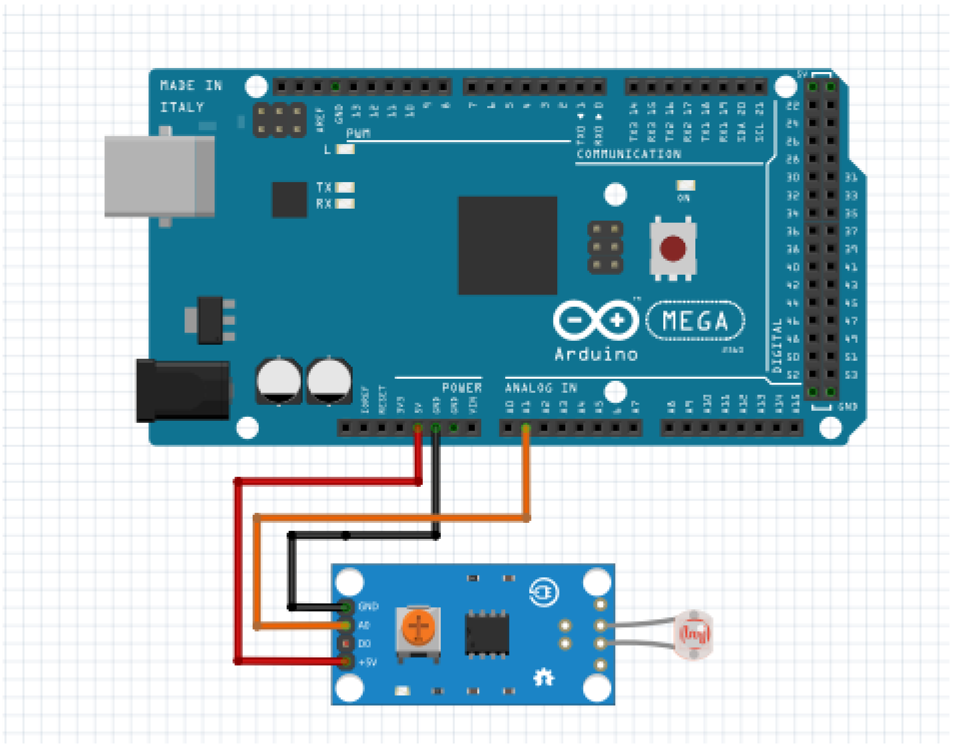


Figure 2.8 Example connection between Arduino Mega and LDR

Table 2.7 LDR sensor Module connection to Arduino Mega

LDR Module	**Arduino Mega**
VCC	**5V**
GND	**GND**
A0	**A1**
D0	**-**

2.3 PREVIOUS STUDIES

2.3.1 ARDUINO-BASED SOIL MOISTURE TESTING SYSTEM

Sounder. Jeevan, N Deepa (2023) ‘Arduino-Based Soil Moisture Testing System’. Designed a system that automatically senses the soil moisture conditions and nourishes them without any need for human involvement. It will automatically monitor the soil and reduce water waste and increase plant growth. Arduino and Soil moisture sensor are used in this project to solve these soil moisture problems, and the major benefits is that the system’s operation can be altered according to the condition of crops, soil, weather situation, and all kind of soil moisture problem.

2.3.2 SMART PLANT MONITORING SYSTEM USING IoT

Sreeram Sadasivam, Viswanath Vadhri, Supradha Ramesh (2015). People in this technological era want to connect with all things in their life, for example the household plants in the urban areas which get neglected in their busy schedules. The Internet of Things is the platform of machines and digital devices which help to transmit data without any human involvement. Also, with the rapid demand for applications of IoT in various fields, this project is done by using IoT. It has the design and implementation of a Smart Plant Monitoring System using a soil moisture sensor, temperature and humidity sensor, relay module, Wi-Fi module, Blynk app, etc. This project detects changes in the moisture, temperature and light conditions in and around the plant, and performs a machine-based curation on the plant by providing necessary irrigation and illumination for the plant.

2.4 PROJECT THEORY

2.4.1 Data Logging System (Text File-Based)

C programming theory involves studying the rules and guidelines that govern the C programming language. As a general-purpose programming language, C is widely used to create operating systems, compilers, and other software. Theoretical concepts in C include variables, which store data such as integers, floating-point numbers, or characters, and data types, which define the kind of data a variable can hold, such as integers, floats, characters, arrays, and structures. Control structures, like loops and if-else statements, manage the program's flow, while functions allow specific tasks to be modularized and reused within a program. Return statements indicate the value a function provides back to the program.

Additionally, modifiers in C influence the memory allocation of variables, optimizing their size and behavior.

This project involves using sensors such as DHT11 for temperature and humidity, soil moisture sensors, and LDR sensors to capture environmental data. The system stores the data in a text file, allowing for efficient and straightforward data logging and analysis. It records the data in a simple text format, making it accessible for analysis without the need for specialized software. Users can review the logged data to identify trends and make data-driven decisions to optimize plant care.

The system offers an accessible, low-cost approach for monitoring plant health, with data easily stored and reviewed in a text file, which can be opened and analyzed on any basic text editor or spreadsheet software.

CHAPTER 3

METHODOLOGY

3.1 PROJECT DESIGN

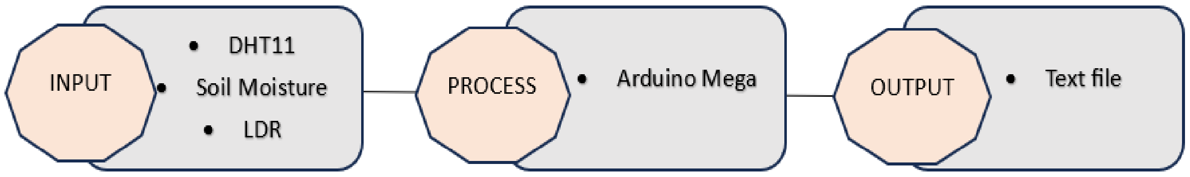


Figure 3.1 Project Design

In the input block, as you can see there are three sensors are utilized likes DHT11, Soil Moisture sensor, and LDR. Each sensor has their own roles in monitoring environmental conditions. It is to maintain the health and growth of plants. The DHT11 sensor measures temperature and humidity making it ideal for identifying optimal planting locations for temperature or humidity especially when you're planting sensitive plant. It enables farmers to monitor environmental conditions and ensure proper care for plants.

The Soil Moisture sensor use to tracks the volumetric water in soil's moisture level to prevent it from becoming too dry or too waterlogged. This help users determine their ideal time for watering. This feature is especially beneficial for individuals with limited experience in plant care and can also assist farmers more effectively.

The LDR (Light Dependent Resistor) monitors light intensity. It's to ensure that plants, particularly those grown indoors, receive adequate lighting for healthy growth.

The process block features the Arduino Mega, which processes data collected from the sensors and transmits it via serial communication. While ensuring its optimal conditions for plant growth, the microcontroller is used to analyses the data that also can be a trigger actions to address environmental irregularities.

The output block utilizes a text file as the medium storage to log the data. Data from the sensors are recorded and stored in the text file named Project_VC23047_VC23052_VC23053_VC23054.txt, allowing users to analyse changes in environmental readings over time. The data will be recorded automatically since the sensor will read the environmental data. This system emphasizes on how important to growth a health care to maintain a sustainability and development goal on green care. Users also can review the recorded data lively to identify periods of insufficient watering or lighting exposure to adjust their care routines times.

3.2 PROJECT DEVELOPMENT & IMPLEMENTATION

The implementation and development of this project involving few of several critical aspects such as software development, hardware development and data logging and analysis implementation. Each of these components contribute on the successful data collection, monitoring and management of plant growth environments.

3.2.1 SOFTWARE DEVELOPMENT & IMPLEMENTATION

The software development for this project that we are using which is C programming. Arduino IDE used for serial communication between sensor and text file on computer. This to ensure an establish communication between the Arduino Mega and c programming to log sensor data continuously before plot it in text file.

i) Serial Communication Setup

The software initializes communication using the F Create File function. The COM port is used in connecting the code to Arduino Mega. Serial port parameters like

The software initializes serial communication using the F Create File function, which opens the COM port connected to the Arduino Mega. To ensure a proper communication, make sure use the right serial port parameters.

- Syntax to include library DHT11

```
#include <DHT.h>
```

Figure 3.2 Code for library DHT 11

- Syntax to define pin all sensor

```
#define DHTPIN 2
#define DHTTYPE DHT11
#define SOIL_MOISTURE_PIN A0
#define LDR_PIN A1
```

Figure 3.3 Code for Pin all sensor

- Syntax to Initialize the DHT11

```
DHT dht(DHTPIN, DHTTYPE);
```

Figure 3.4 Code for initialize DHT11

- Syntax to transfer data between the Arduino and Code Block

```
Serial.begin(9600);
```

Figure 3.5 Code for transfers data to Code Block

- Syntax to initialize the Sensor as input

```
pinMode(SOIL_MOISTURE_PIN, INPUT);
pinMode(LDR_PIN, INPUT);
```

Figure 3.6 Code for initialize sensor

- Syntax to Read DHT11 data

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
```

Figure 3.7 Code for read DHT11

- Syntax to Read Soil Moisture data

```
int soilMoisture = analogRead(SOIL_MOISTURE_PIN);
```

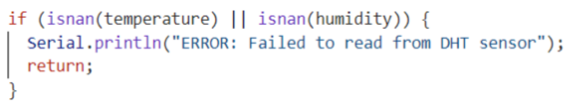
Figure 3.8 Code for read soil moisture

- Syntax to Read Light Intensity

```
int lightIntensity = analogRead(LDR_PIN);
```

Figure 3.9 Code for read light intensity

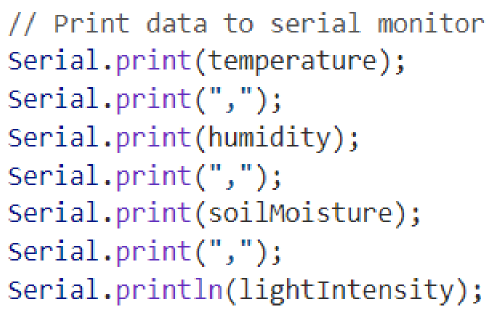
- Syntax to heck for errors in DHT11 readings



```
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("ERROR: Failed to read from DHT sensor");
  return;
}
```

Figure 3.10 Code for check errors in DHT11 readings

- Syntax to print data to serial in serial monitor



```
// Print data to serial monitor
Serial.print(temperature);
Serial.print(",");
Serial.print(humidity);
Serial.print(",");
Serial.print(soilMoisture);
Serial.print(",");
Serial.println(lightIntensity);
```

Figure 3.11 Code for print data to serial monitor

ii) Data logging

- A text file named Project_VC23047_VC23052_VC23053_VC23054.txt is created or opened for writing data.
- The file begins with a header: "Temperature Humidity Soil Moisture Light Intensity".
- Sensor data received from the Arduino via the serial port is continuously read, formatted, and logged to the file.

iii) Real-Time Console Feedback

- Data received through the serial port is printed to the console for real-time monitoring.
- The **printf** function allows users to view incoming data directly.

iv) Data handling

- The data buffer ensures that incoming data is null-terminated for proper formatting.

v) Error handling

- The program handles errors related to opening the serial port, configuring communication settings, and file access. Proper messages are displayed to notify users of any issues.

3.2.3 DATA LOGGING USING SERIAL TO CODE BLOCK

Data logging plays a crucial role in the success of this plant monitoring project, enabling continuous data collection and analysis of the plant environment. This process minimizes human involvement by providing real-time monitoring for key parameters, including temperature, humidity, soil moisture, and light intensity. The logged data facilitates trend analysis and decision-making to optimize plant health and growth.

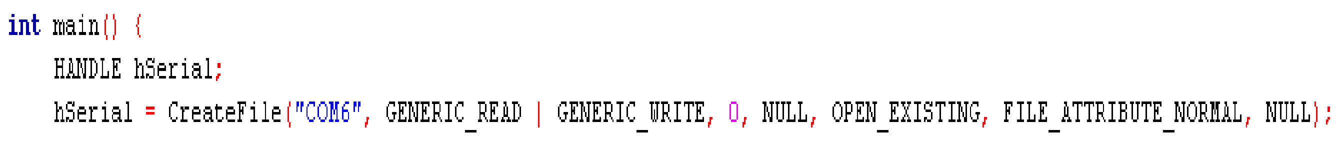
In this implementation, sensor data is captured at a predefined time interval of two seconds, ensuring consistent and reliable updates. The Arduino Mega processes the data and sends it via serial communication to the logging system. The C program accept the data from Serial Communication and save to it in the notepad

The received data is written into a text file named Project_VC23047_VC23052_VC23053_VC23054.txt. This file is structured with columns for each parameter—temperature, humidity, soil moisture, and light intensity—ensuring the data is organized and easy to analyse.

The program continuously appends new data to the file in real-time, ensuring no data is lost during logging. By utilizing a text file for data storage, the system provides a lightweight and accessible solution. The text file can be easily imported into analysis tools like Excel for visualization and trend analysis, enabling users to make informed decisions about the plant environment.

i) Open the Serial Port:

- The serial port is opened using the CreateFile function.
- This function attempts to open the COM port (COM6 in this case) for both reading and writing. If successful, it returns a handle (hSerial) to the port. If it fails, INVALID_HANDLE_VALUE is returned, and an error message is displayed.

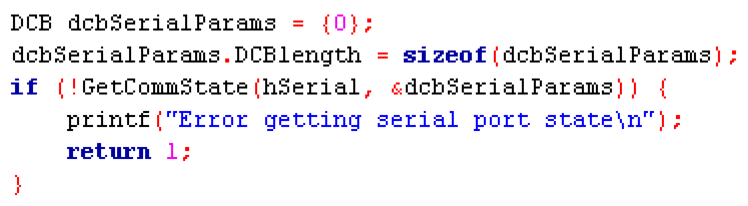


```
int main() {
    HANDLE hSerial;
    hSerial = CreateFile("COM6", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

Figure 3.12 Code for open to serial monitor

ii) Initializing the DCB Structure:

- The DCB (Device Control Block) structure is initialized to configure the serial port's communication parameters.
- The DCBlength field is set to the size of the DCB structure. This structure holds the configuration for serial communication, such as baud rate, byte size, stop bits, and parity.

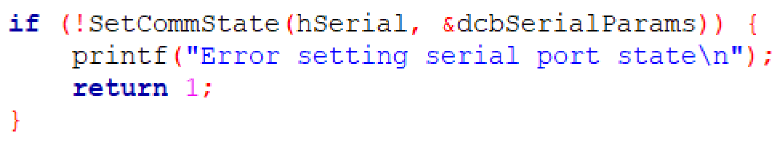


```
DCB dcbSerialParams = {0};
dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
if (!GetCommState(hSerial, &dcbSerialParams)) {
    printf("Error getting serial port state\n");
    return 1;
}
```

Figure 3.13 Code for initialize DCB structure

iii) Getting the Current Serial Port Settings:

- The GetCommState function is used to retrieve the current settings of the serial port.

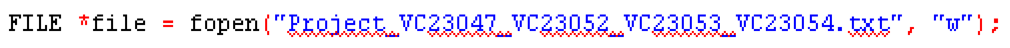


```
if (!SetCommState(hSerial, &dcbSerialParams)) {
    printf("Error setting serial port state\n");
    return 1;
}
```

Figure 3.14 Code for getting the current serial port setting

iv) Open the File for Writing

- The program attempts to open a file named Project_VC23047_VC23052_VC23053_VC23054.txt in write mode ("w"), using the fopen function.

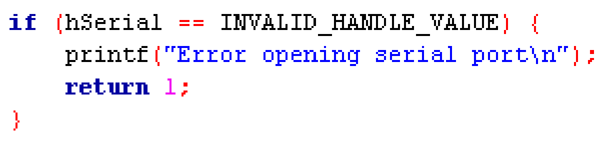


```
FILE *file = fopen("Project_VC23047_VC23052_VC23053_VC23054.txt", "w");
```

Figure 3.15 Code for writjng the file

v) Error Checking for File Opening:

- The program checks if the file was successfully opened:

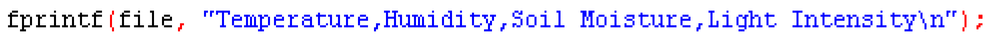


```
if (hSerial == INVALID_HANDLE_VALUE) {
    printf("Error opening serial port\n");
    return 1;
}
```

Figure 3.16 Code for check error while opening file

vi) Writing the Header to the File:

- The program writes the header row to the file, specifying the names of the data columns (Temperature, Humidity, Soil Moisture, and Light Intensity). This is done using the fprintf function:

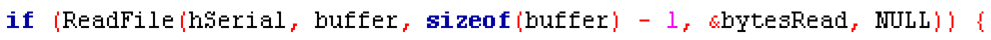


```
fprintf(file, "Temperature,Humidity,Soil Moisture,Light Intensity\n");
```

Figure 3.17 Code for writing header to the file

vii) Infinite loop for Continuous Data Reading

- ReadFile function is used to read data from the serial port (represented by the handle hSerial) into a buffer. The function reads up to 255 bytes of data (reserving 1 byte for the null terminator to mark the end of the string). The number of bytes successfully read is stored in the bytesRead variable, which helps keep track of how much data was retrieved from the port.

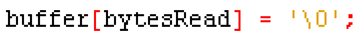


```
if (ReadFile(hSerial, buffer, sizeof(buffer) - 1, &bytesRead, NULL)) {
```

Figure 3.18 Code for continuous reading data raw

viii) Null-Termination of the String:

- The data read into the buffer is null-terminated to make it a valid string.

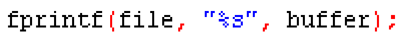


```
buffer[bytesRead] = '\0';
```

Figure 3.19 Code for read data into the buffer

ix) Writing Data to the File:

- The fprintf function writes the data from the buffer to the file in CSV format.

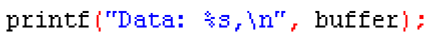


```
fprintf(file, "%s", buffer);
```

Figure 3.20 Code for write the file in string

x) Printing Data to the Console

- The printf function prints the data to the console, providing real-time feedback of the sensor values.

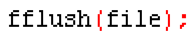


```
printf("Data: %s,\n", buffer);
```

Figure 3.21 Code for print data to the console

xi) **Flushing the File Buffer:**

- The fflush function is used to immediately write the data to the file, ensuring no data is lost or delayed.



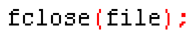
```
fflush(file);
```

Figure 3.22 Code for ensure data save to the file

xii) **Close file:**

Closing the file with `fclose()` ensures that:

- Any buffered data (in case it wasn't written yet) is flushed to the file.
- System resources that were allocated to access the file (like file handles) are released.
- The file pointer is no longer valid after this, and you should not use it for further file operations.

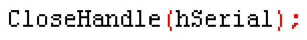


```
fclose(file);
```

Figure 3.23 Code for close the file

xiii) **Close a handle Serial Port**

- Serial Port: When you open a serial port with CreateFile(), you get a handle to that port. After you're done using it for communication (in this case, reading data from it), you close the handle with CloseHandle(hSerial) to release the port and its resources.



```
CloseHandle(hSerial);
```

Figure 3.24 Code for Close the Serial port

3.2.4 DATA ANALYSIS

Data analysis in plant care involves interpreting collected data to identify trends and patterns that support informed decision-making. The primary focus is on monitoring key parameters such as temperature, humidity, soil moisture, and light intensity to understand their fluctuations and detect potential anomalies. These anomalies, such as insufficient watering, overwatering, or inadequate lighting, can adversely affect plant health and growth.

The logged data is stored in a text file for easy access. By reviewing the data, users can observe trends over time, spot any abnormal readings, and address issues like low soil moisture, high temperatures, or inadequate lighting. Identifying such anomalies helps ensure timely interventions, improving plant care and optimizing the growing environment.

- Simplified Data Storage: Text files are easy to generate, access, and store, requiring minimal technical setup.
- Easy Portability: Text files can be easily shared, transferred, and opened across different systems, making them a versatile solution.
- Lightweight: Text files are lightweight and do not consume significant system resources, making them ideal for basic data logging tasks.
- Clear and Readable Format: Data in text files is stored in a plain, human-readable format, making it easier to manually review and analyse without the need for specialized software.

3.3 PROJECT USABILITY FLOW CHART

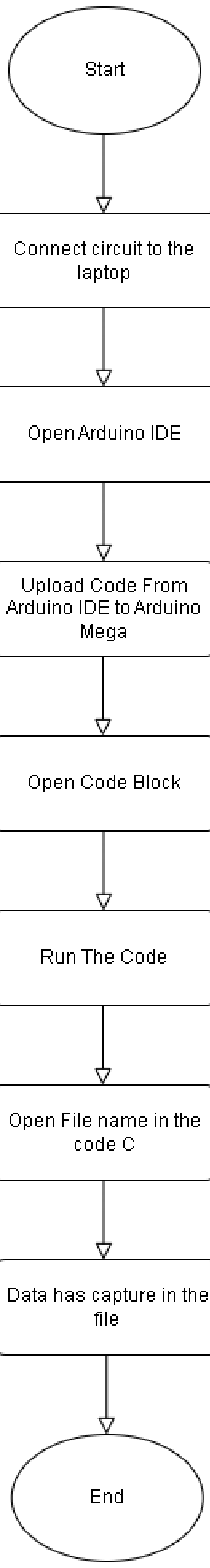

Start
Connect circuit to the laptop
Open Arduino IDE
Upload Code From Arduino IDE to Arduino Mega
Open Code Block
Run The Code
Open File name in the code C
Data has capture in the file
End

Figure 3.25 Project Flow Chart Usability

CHAPTER 4

FINDING AND ANALYSIS

4.1 RESULT & ANALYSIS

For this chapter represent the finding we get obtained from the implementation and testing environmental monitoring system. The system was created to collect data from three sensor which is DHT11 for temperature and humidity, soil moisture sensor, and the LDR sensor for light intensity. By using Arduino Mega, data was collected into a text file using real-time data. The data collected to show the integration between sensor and c programming.

Data are collected into text files show the environmental patterns on humidity, temperature, volumetric water and light intensity using our Smart Monitoring Plant system. The comparisons with expected outcomes are also provided to assess the system reliability and practicality. Challenges that we face during data collection and system testing are discussed along with their potential impact for the result.

This chapter aims to we provide a comprehensive understanding of how the system perform under the real-world conditions and to interpret the significance of the collected data in relation to the project objectives

4.1.1 VOLTAGE READING

This voltage test is done to see the voltage received by each component or sensor used so that the reading received is stable when the project is underway. The test data was collected as shown in the table below.

Table 4.1 Voltage Reading of Sensors

No	Testing	Voltage Reading			Number of Test	Status
		Pin	Normal	Measure		
1	Soil Moisture	VIN	5V	4.5V	1	Succeeded
		GND	0V	0V	2	Succeeded
					3	Succeeded
		Pin	Normal	Measure		
2	LDR Module	VIN	5V	4.3V	1	Succeeded
		GND	0V	0V	2	Succeeded
					3	Succeeded
		Pin	Normal	Measure		
3	DHT11	VIN	5V	4.8V	1	Succeeded
		GND	0V	0V	2	Succeeded
					3	Succeeded

4.2 SENSITIVITY SETTINGS

In this section, a sensitivity test is performed to identify the sensitivity of the sensor in order to launch the monitoring process.

4.2.1 LDR SENSOR SENSITIVITY

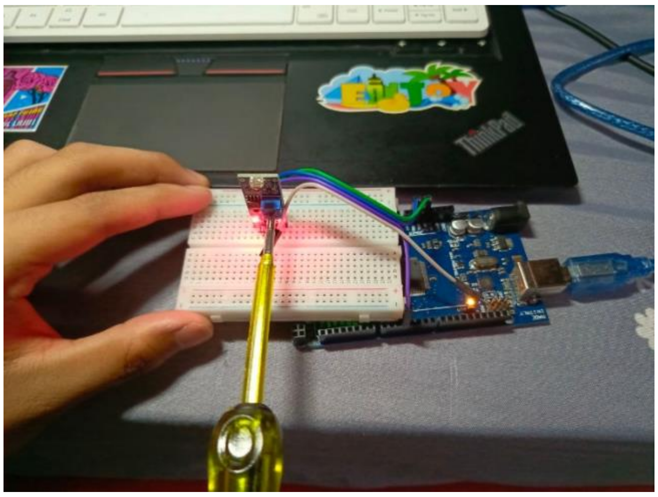


Figure 4.1 LDR Sensor Sensitivity Test

This sensitivity test is done to see the differentiate range on light intensity detected by LDR sensor. By adjust this sensitivity settings, user can measure and know the exact range of light level. Higher lux value indicates that the LDR sensor is less exposed to the light while low lux value indicates that the LDR sensor is exposed to light. These adjustments ensure the voltage receive by sensor remains stable. The measured data is presented in the table below.

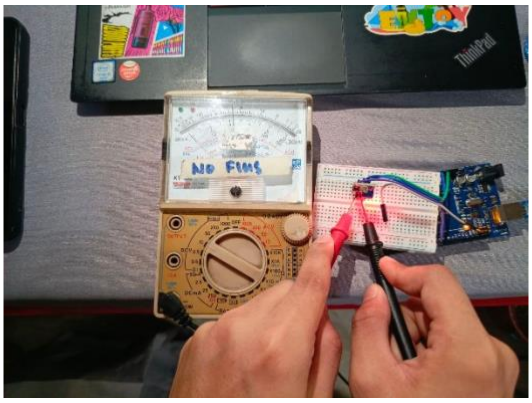


Figure 4.2 LDR Sensor Voltage Measuring Value

Table 4.2 Measured Lux Value from LDR Sensor

Lux Value	Conditions
0	Dark
4.26	Light

4.2.2 SOIL MOISTURE SENSITIVITY

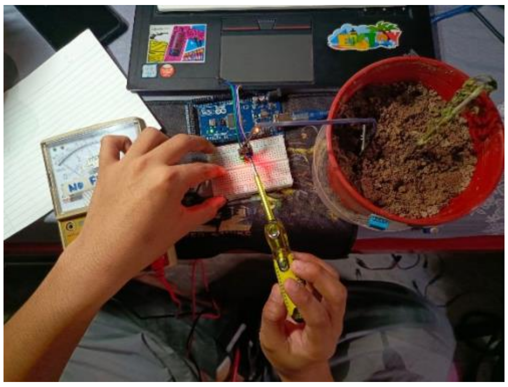


Figure 4.3 Soil Moisture Sensor Sensitivity Test

A sensitivity test was made on the soil moisture sensor before project monitoring was carried out. This is because it is to identify the sensitivity of the sensor to soil moisture. This sensitivity test is made by identifying the raw value for soil moisture by knowing the estimated raw value for dry soil, moist soil and wet soil. This is done so that the monitoring process can be done efficiently and the data obtained is appropriate to the condition of the land. In addition, it is to set the sensitivity of the sensor to the detection of soil moisture whether sensitive or less sensitive, adjusted by using an adjustable resistor that is included in the sensor module.

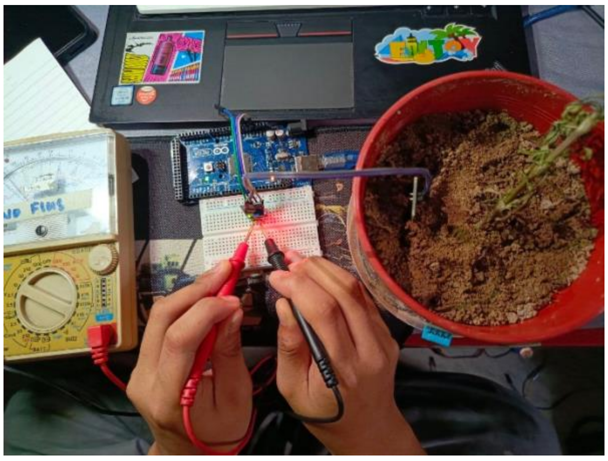


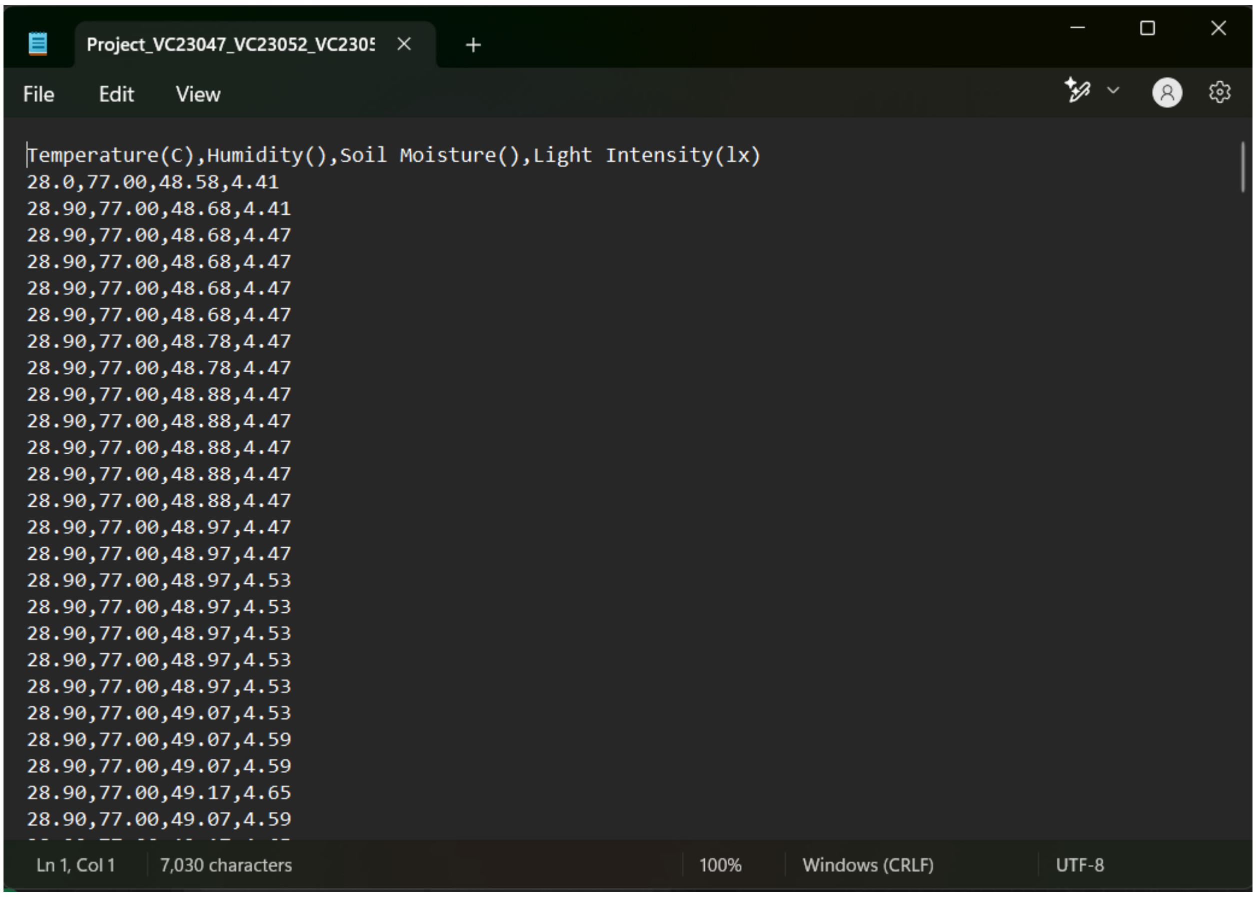
Figure 4.4 Soil Moisture Sensor Voltage Measuring

Table 4.3 Measured Soil Percentage from Soil Moisture Sensor

Range Value (%)	Conditions
0 - 30	Dry
31 - 80	Moist
81 - 100	Wet

A measured test was made on the soil moisture sensor to know the range value in percentage when the conditions is dry, moist and wet. This is to identify the value of volumetric water through soil moisture sensor. When the range value is around 0 to 30%. It's shown that the conditions of soil are dry. When the range value is around 31 to 80%, the conditions is moist. For range value 81 to 100% show that the condition of soil is wet. It will help users especially farmers to monitor out their plant ensuring the plants is in the good health care.

4.3 TEXT FILE



```
Temperature(C),Humidity(),Soil Moisture(),Light Intensity(lx)
28.0,77.00,48.58,4.41
28.90,77.00,48.68,4.41
28.90,77.00,48.68,4.47
28.90,77.00,48.68,4.47
28.90,77.00,48.68,4.47
28.90,77.00,48.68,4.47
28.90,77.00,48.78,4.47
28.90,77.00,48.78,4.47
28.90,77.00,48.88,4.47
28.90,77.00,48.88,4.47
28.90,77.00,48.88,4.47
28.90,77.00,48.88,4.47
28.90,77.00,48.88,4.47
28.90,77.00,48.97,4.47
28.90,77.00,48.97,4.47
28.90,77.00,48.97,4.53
28.90,77.00,48.97,4.53
28.90,77.00,48.97,4.53
28.90,77.00,48.97,4.53
28.90,77.00,48.97,4.53
28.90,77.00,49.07,4.53
28.90,77.00,49.07,4.59
28.90,77.00,49.07,4.59
28.90,77.00,49.17,4.65
28.90,77.00,49.07,4.59
```

Figure 4.5 Soil Moisture Sensor Record Data

This data provides an overview of environmental conditions like temperature, humidity, soil moisture and light intensity. Its shows that the temperature remain stable fluctuating between 28-29°C. Same goes to humidity, light intensity and water volumetric which is for humidity is consistently play around 77%. For the soil moisture varies between 46-50% indicating the volume of water are just enough as not dry and not really wet. Light intensity fluctuates between 3 and 5 *lx.* Reflecting the variations when exposed to the light and when not. This shows how the data are collected into the text file and store it automatically using real time.

CHAPTER 5

RECOMMENDATION AND SOLUTION

5.1 INTRODUCTION

The Smart Plant Monitoring System, developed as part of this project, uses DHT11, LDR, and soil moisture sensors to monitor key environmental factors like temperature, humidity, light intensity, and soil moisture. These sensors help in capturing real-time data, ensuring that plant care can be optimized within the project's designated range. However, the current system is limited to monitoring within the immediate area. After discussing potential future improvements, we have outlined recommendations to enhance the system's functionality and broaden its capabilities.

5.2 RECOMMENDATION

5.2.1 UPGRADE SENSOR

- **Incorporate a pH Sensor**: To enhance the accuracy of soil health monitoring, it's recommended to include a specific **pH sensor**. This would allow for better monitoring of soil acidity or alkalinity, providing a more comprehensive view of the plant's growing environment.

- **Use Multiple Soil Moisture Sensors**: To achieve more precise soil hydration data across different areas, consider implementing **multiple types of soil moisture sensors**. This could help in evaluating how various sections of the soil or plant beds are hydrated, ensuring that watering strategies are optimized for different plant types.

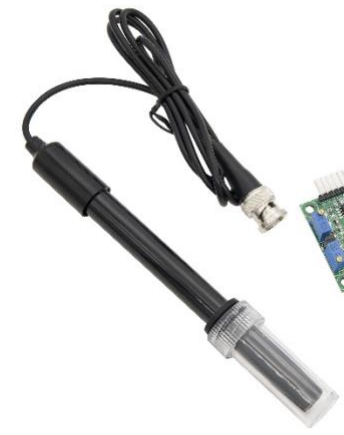


Figure 5.1 Image of pH sensor

5.2.2 EXPAND MONITORING RANGE

- **Wireless Connectivity**: Implementing **wireless communication technologies** (such as Wi-Fi, Bluetooth, or Zigbee) would allow the system to send data remotely, thus enabling **real-time monitoring from anywhere**, even outside the project's physical radius. This could be achieved by integrating a wireless module such as ESP8266 or using IoT platforms for cloud-based data storage and analysis.

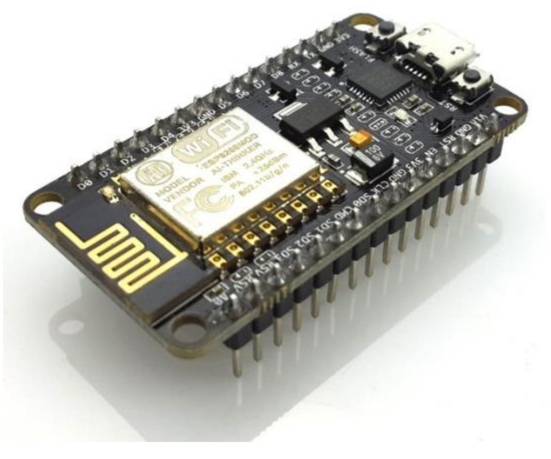


Figure 5.2 Image of Arduino ESP8266 WiFI

5.2.3 ENHANCE DATA ANALYSIS AND VISUALIZATION

- For better decision-making, consider incorporating advanced data visualization tools (like **charts, graphs, and trend lines**) that could be easily accessed via mobile devices or computers.
- These tools would allow users to identify patterns, outliers, or anomalies in the data more efficiently, improving overall plant care.

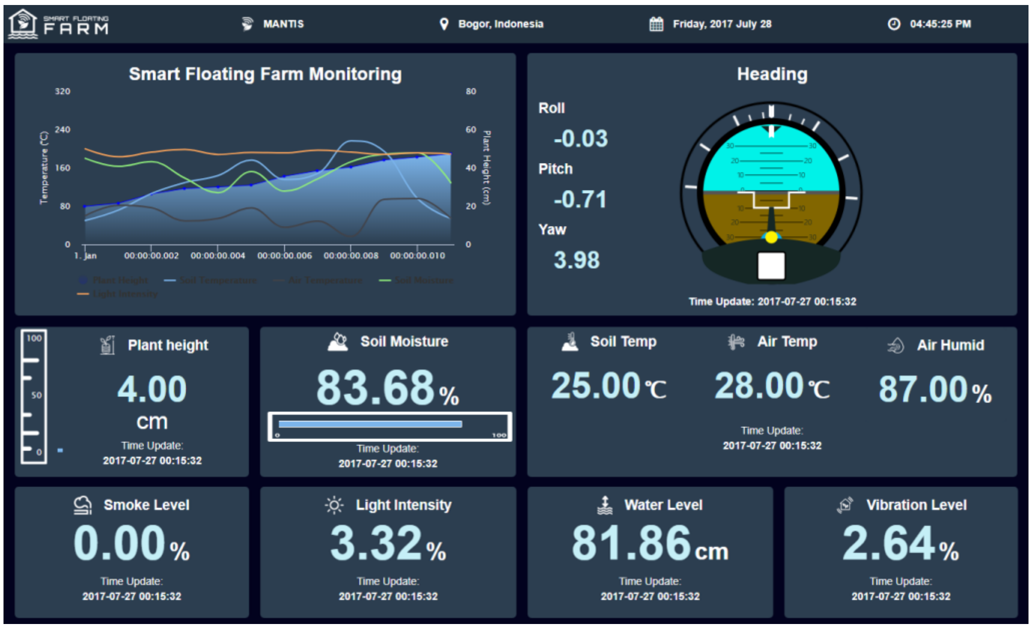


Figure 5.3 Example of using dashboard app

5.3 SOLUTION

The current Smart Plant Monitoring System provides a solid foundation for monitoring key environmental factors that influence plant health. While the system has proven effective within its limitations, the recommended upgrades and future improvements will enhance its performance, expand its functionality, and provide more precise and comprehensive data. By incorporating these changes, the system will better support plant care and help users make more informed decisions for optimized plant health.

5.4 CONCLUSION

To conclude this, this smart plant monitoring system has its potential to be improved more efficiency to collect the data. By using our implement idea which is using new dashboard and sensor will make the data that are collected more details. It's also having some potential on market demanding itself. The objective by making this project has already been accomplished as we can see the result at above. The objective is like below:

- To design a cost-effective system for monitoring environmental conditions essential for plant growth and health
- To develop an automatic monitoring system to saving time and more organized
- To provide a real time data logging light intensity, temperature, humidity and soil moisture level of plant to enable data-driven decision making

Table 5.1 Proof that this project is more worthwhile.

Feature	Other Projects	Your Project	Evidence of Difference
Cost-Effectiveness	Uses more expensive components like xiomi smart plan 2 have price and RM 86.90	Uses affordable components like DHT11, LDR, and soil moisture sensors and price below RM30.	Component list and pricing show lower costs for your project without compromising performance.
Automation	Automation is limited to basic monitoring As an example, only one item can be monitored at a time.	Fully automated with monitoring and control of systems like irrigation or lighting.	Flow diagrams showing how full automation works in your system.
Real-Time Monitoring	Data is not recorded or accessible in real-time.	Provides real-time data logging for light intensity, temperature, humidity, and soil moisture.	Demo or screenshots showing real-time data accessible via apps or dashboards.

REFFERENCES

Cytron Technologies Sdn Bhd. (2025). *DHT11 Temperature and Humidity Sensor Module Breakout*. Cytron Technologies Malaysia. https://my.cytron.io/p-dht11-sensor-module-breakout?srsltid=AfmBOoqvXM-1ZlDkj38bwXOS1eVSpO-P4B-Nqhja0QFPsXnerth7T4cS&r=1

Last Minute Engineers. (2019, November 29). *How Soil Moisture Sensor Works and Interface it with Arduino*. Last Minute Engineers; Last Minute Engineers. https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/

Sounder. Jeevan, & N Deepa. (2023). Arduino-Based Soil Moisture Testing System. *ResearchGate*, *8*(2), 1–6. https://www.researchgate.net/publication/372966475_Arduino-Based_Soil_Moisture_Testing_System

Sreeram Sadasivam, Viswanath Vadhri, & Ramesh, S. (2015). Smart Plant Monitoring System. *ResearchGate*. https://doi.org/10.13140/RG.2.1.2915.6563/1

techZeero. (2018, September 10). *LDR Sensor (Light Dependent Resistor) - Pinout, Connections*. TechZeero. https://techzeero.com/sensors-modules/ldr-sensor/

APPENDIX

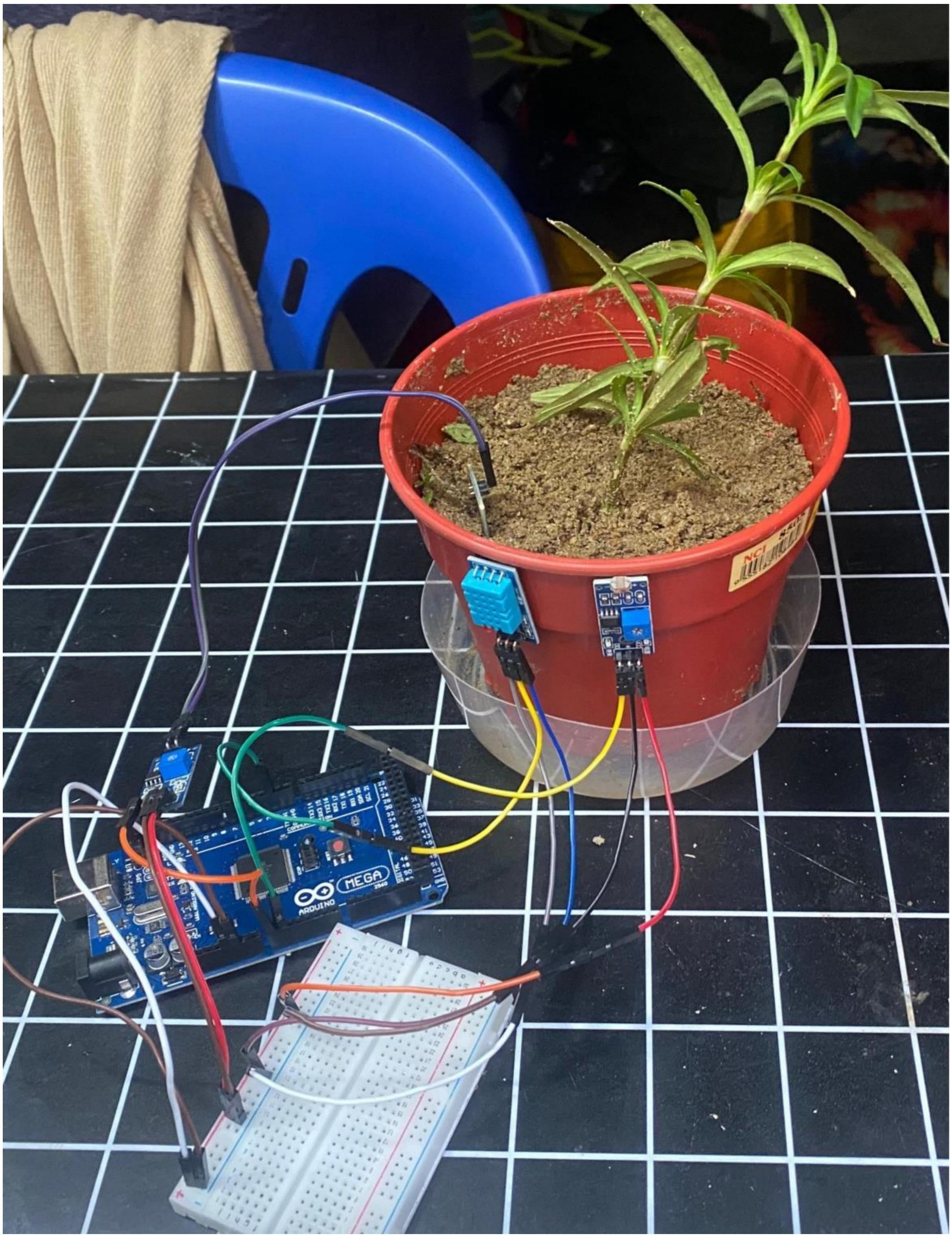
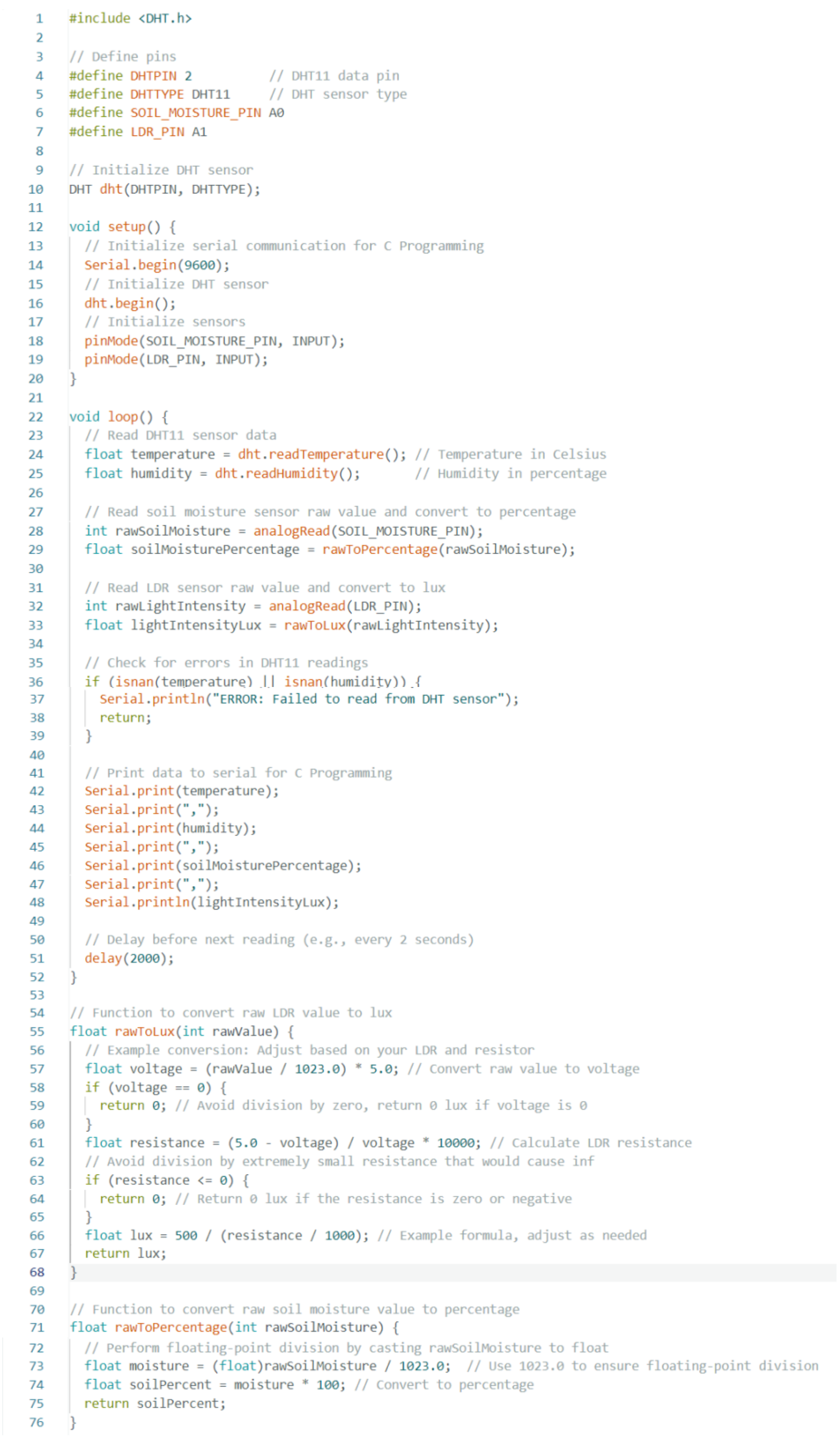
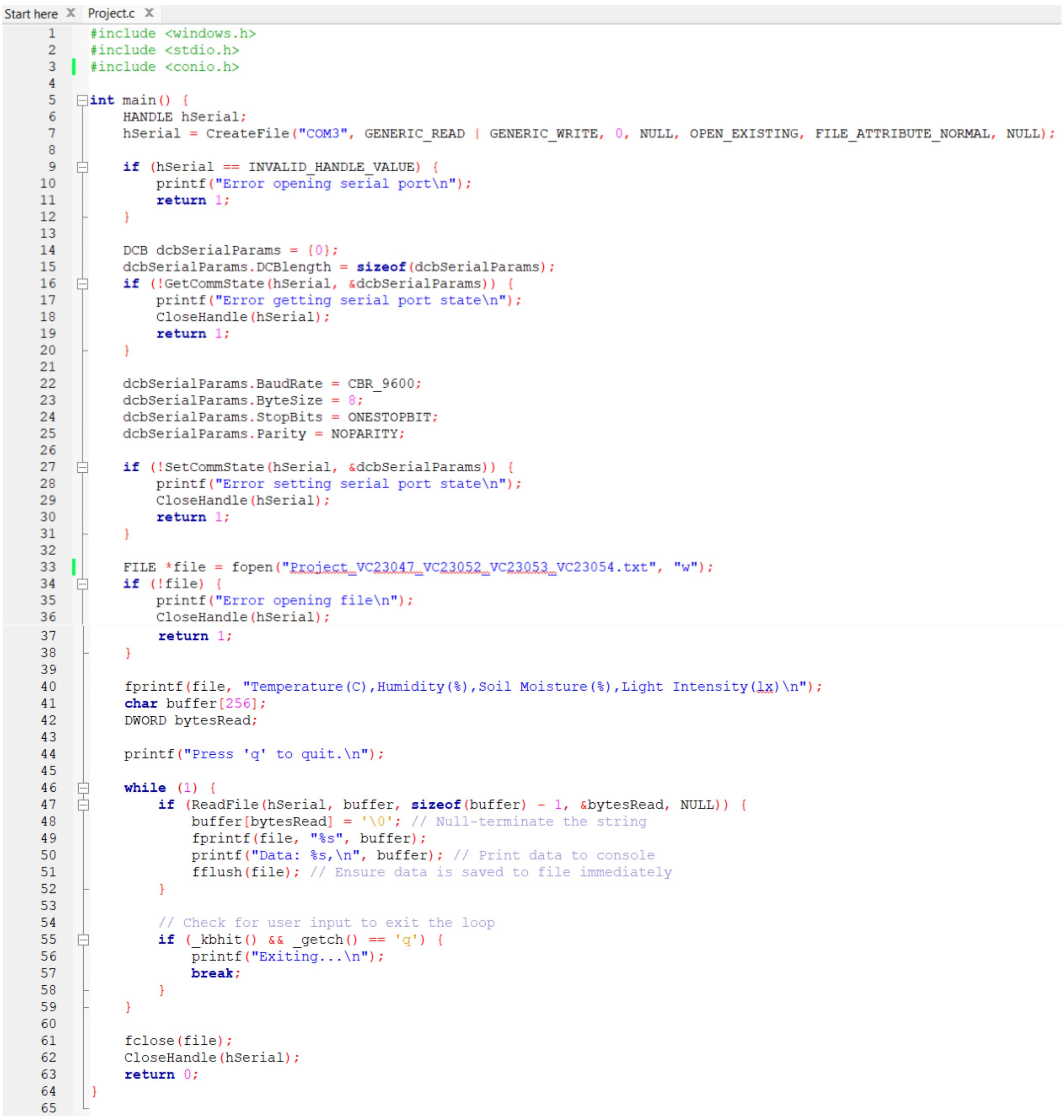


Figure 6.1 Actual Project Smart Plant Monitoring System



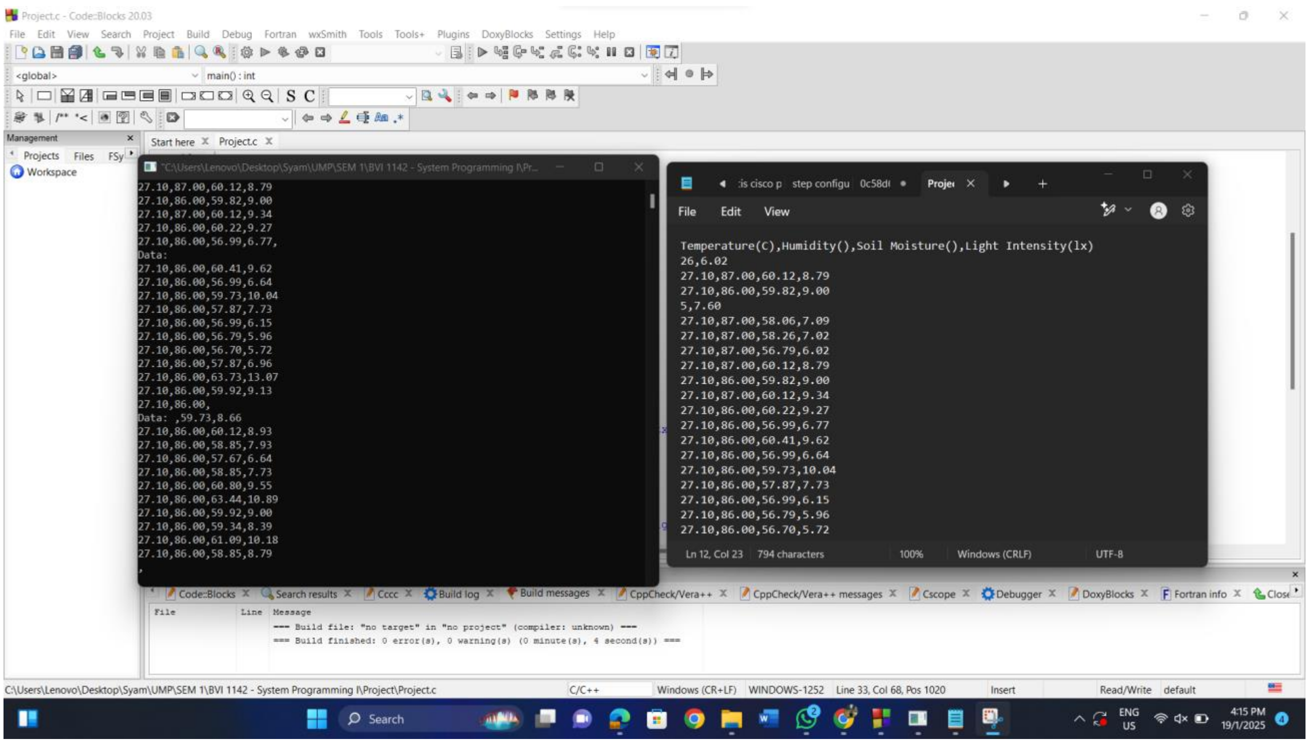
```
#include <DHT.h>

// Define pins
#define DHTPIN 2          // DHT11 data pin
#define DHTTYPE DHT11     // DHT sensor type
#define SOIL_MOISTURE_PIN A0
#define LDR_PIN A1

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // Initialize serial communication for C Programming
  Serial.begin(9600);
  // Initialize DHT sensor
  dht.begin();
  // Initialize sensors
  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
}

void loop() {
  // Read DHT11 sensor data
  float temperature = dht.readTemperature(); // Temperature in Celsius
  float humidity = dht.readHumidity();       // Humidity in percentage

  // Read soil moisture sensor raw value and convert to percentage
  int rawSoilMoisture = analogRead(SOIL_MOISTURE_PIN);
  float soilMoisturePercentage = rawToPercentage(rawSoilMoisture);

  // Read LDR sensor raw value and convert to lux
  int rawLightIntensity = analogRead(LDR_PIN);
  float lightIntensityLux = rawToLux(rawLightIntensity);

  // Check for errors in DHT11 readings
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("ERROR: Failed to read from DHT sensor");
    return;
  }

  // Print data to serial for C Programming
  Serial.print(temperature);
  Serial.print(",");
  Serial.print(humidity);
  Serial.print(",");
  Serial.print(soilMoisturePercentage);
  Serial.print(",");
  Serial.println(lightIntensityLux);

  // Delay before next reading (e.g., every 2 seconds)
  delay(2000);
}

// Function to convert raw LDR value to lux
float rawToLux(int rawValue) {
  // Example conversion: Adjust based on your LDR and resistor
  float voltage = (rawValue / 1023.0) * 5.0; // Convert raw value to voltage
  if (voltage == 0) {
    return 0; // Avoid division by zero, return 0 lux if voltage is 0
  }
  float resistance = (5.0 - voltage) / voltage * 10000; // Calculate LDR resistance
  // Avoid division by extremely small resistance that would cause inf
  if (resistance <= 0) {
    return 0; // Return 0 lux if the resistance is zero or negative
  }
  float lux = 500 / (resistance / 1000); // Example formula, adjust as needed
  return lux;
}

// Function to convert raw soil moisture value to percentage
float rawToPercentage(int rawSoilMoisture) {
  // Perform floating-point division by casting rawSoilMoisture to float
  float moisture = (float)rawSoilMoisture / 1023.0;  // Use 1023.0 to ensure floating-point division
  float soilPercent = moisture * 100; // Convert to percentage
  return soilPercent;
}
```

Figure 6.2 Full Arduino Code for Capture Data from Sensor



```c
#include <windows.h>
#include <stdio.h>
#include <conio.h>

int main() {
    HANDLE hSerial;
    hSerial = CreateFile("COM3", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

    if (hSerial == INVALID_HANDLE_VALUE) {
        printf("Error opening serial port\n");
        return 1;
    }

    DCB dcbSerialParams = {0};
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (!GetCommState(hSerial, &dcbSerialParams)) {
        printf("Error getting serial port state\n");
        CloseHandle(hSerial);
        return 1;
    }

    dcbSerialParams.BaudRate = CBR_9600;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;

    if (!SetCommState(hSerial, &dcbSerialParams)) {
        printf("Error setting serial port state\n");
        CloseHandle(hSerial);
        return 1;
    }

    FILE *file = fopen("Project_VC23047_VC23052_VC23053_VC23054.txt", "w");
    if (!file) {
        printf("Error opening file\n");
        CloseHandle(hSerial);
        return 1;
    }

    fprintf(file, "Temperature(C),Humidity(%),Soil Moisture(%),Light Intensity(lx)\n");
    char buffer[256];
    DWORD bytesRead;

    printf("Press 'q' to quit.\n");

    while (1) {
        if (ReadFile(hSerial, buffer, sizeof(buffer) - 1, &bytesRead, NULL)) {
            buffer[bytesRead] = '\0'; // Null-terminate the string
            fprintf(file, "%s", buffer);
            printf("Data: %s,\n", buffer); // Print data to console
            fflush(file); // Ensure data is saved to file immediately
        }

        // Check for user input to exit the loop
        if (_kbhit() && _getch() == 'q') {
            printf("Exiting...\n");
            break;
        }
    }

    fclose(file);
    CloseHandle(hSerial);
    return 0;
}
```

Figure 6.3 Full Code of File Operation on C to Capture data from Serial



```
"C:\Users\Lenovo\Desktop\Syam\UMP\SEM 1\BVI 1142 - System Programming I\Pr...
27.10,87.00,60.12,8.79
27.10,86.00,59.82,9.00
27.10,87.00,60.12,9.34
27.10,86.00,60.22,9.27
27.10,86.00,56.99,6.77,
Data:
27.10,86.00,60.41,9.62
27.10,86.00,56.99,6.64
27.10,86.00,59.73,10.04
27.10,86.00,57.87,7.73
27.10,86.00,56.99,6.15
27.10,86.00,56.79,5.96
27.10,86.00,56.70,5.72
27.10,86.00,57.87,6.96
27.10,86.00,63.73,13.07
27.10,86.00,59.92,9.13
27.10,86.00,
Data: ,59.73,8.66
27.10,86.00,60.12,8.93
27.10,86.00,58.85,7.93
27.10,86.00,57.67,6.64
27.10,86.00,58.85,7.73
27.10,86.00,60.80,9.55
27.10,86.00,63.44,10.89
27.10,86.00,59.92,9.00
27.10,86.00,59.34,8.39
27.10,86.00,61.09,10.18
27.10,86.00,58.85,8.79
,
```

```
Temperature(C),Humidity(),Soil Moisture(),Light Intensity(lx)
26,6.02
27.10,87.00,60.12,8.79
27.10,86.00,59.82,9.00
5,7.60
27.10,87.00,58.06,7.09
27.10,87.00,58.26,7.02
27.10,87.00,56.79,6.02
27.10,87.00,60.12,8.79
27.10,86.00,59.82,9.00
27.10,87.00,60.12,9.34
27.10,86.00,60.22,9.27
27.10,86.00,56.99,6.77
27.10,86.00,60.41,9.62
27.10,86.00,56.99,6.64
27.10,86.00,59.73,10.04
27.10,86.00,57.87,7.73
27.10,86.00,56.99,6.15
27.10,86.00,56.79,5.96
27.10,86.00,56.70,5.72
```

```
File          Line  Message
                    --- Build file: "no target" in "no project" (compiler: unknown) ---
                    === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 4 second(s)) ===
```

Project.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
Management
Projects Files FSy
Workspace
Start here
Project.c
File Edit View
Ln 12, Col 23 794 characters 100% Windows (CRLF) UTF-8
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info
C:\Users\Lenovo\Desktop\Syam\UMP\SEM 1\BVI 1142 - System Programming I\Project\Project.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 33, Col 68, Pos 1020 Insert Read/Write default
4:15 PM 19/1/2025

Figure 6.4 Data Logging Interfaces

Link video presentation:

https://youtu.be/iE6EkViPATc?si=4TU8rHGeaf4pEuOV