Department of Information Technology

Bharati Vidyapeeth College of Engineering, Navi Mumbai

Academic Year 2023-2024

Sensor Lab Project Report On

"Plant Language Translator"

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

By

Yash Bodhe	08
Arpita Chavan	10
Pushkraj Dhamale	12
Rohit Gupta	19

Under the Guidance of

Prof. V. E. Pawar.

Bharati Vidyapeeth College of Engineering, Navi Mumbai

Department of Information Technology

CERTIFICATE

This is to certify that,

Yash Bodhe	08
Arpita Chavan	10
Pushkraj Dhamale	12
Rohit Gupta	19

have completed the Sensor Lab Project ***"Plant Language Translator"*** *Satisfactorily in the Department of Information Technology as prescribed by the Mumbai University in the academic year 2023-2024.*

Prof. V. E. Pawar		Prof. S. N. Mahtre
Subject In charge	External Examiner	H.O.D

ACKNOWLEDGEMENT

I would like to express my sincere gratitude towards Prof. V. E. Pawar for the help, guidance and encouragement he provided during the Project work. This work would have not been possible without his valuable time, patience and motivation. I thank her for making my stint thoroughly pleasant and enriching. I take the privilege to express my sincere thanks to Prof S. N. Mhatre H.O.D and Dr Sandhya Jadhav, our Principal for providing encouragement and much support throughout my work.

Yash Bodhe	08
Arpita Chavan	10
Pushkraj Dhamale	12
Rohit Gupta	19

Content

Abstract

The "Plant Language Translator" project is a unique fusion of Arduino Uno, a moisture sensor, and the DHT11 sensor, enabling effective communication between humans and plants. This innovative system harnesses the capabilities of Arduino-based microcontrollers and environmental sensors to monitor and interpret the specific needs of indoor and outdoor plants. The DHT11 sensor measures ambient temperature and humidity, providing crucial insights into the plant's surroundings, while a moisture sensor assesses soil moisture levels to ensure optimal hydration.

The core functionality of this system lies in its ability to "translate" these environmental parameters into actionable information for plant care. Through a user-friendly interface or display, real-time feedback on the plant's well-being is presented, including data on temperature, humidity, and soil moisture. The microcontroller processes sensor data and, based on predefined thresholds, generates alerts or recommendations for plant maintenance. For example, if the soil moisture falls below a certain level, the system can prompt the user to water the plant.

In essence, the "Plant Language Translator" project embodies the harmonious blend of technology and nature, showcasing how Arduino Uno, moisture sensors, and the DHT11 sensor can enhance plant care by providing essential environmental insights and translating them into practical care instructions for plant enthusiasts and gardeners. This system introduces an innovative approach to plant nurturing, highlighting the practical applications of Arduino-based solutions in the fields of agriculture and environmental monitoring.

Chapter 1

Introduction

In a world where technology continually shapes our lives, the profound connection between humanity and the natural world becomes increasingly significant. The "Plant Language Translator" project is an ambitious undertaking that seeks to merge the realms of science, technology, and nature by harnessing the power of Arduino Uno microcontrollers, environmental sensors, and a Python-based graphical user interface. At its core, this project endeavors to create a novel and intuitive communication channel with plants, driven by the understanding of their vital requirements.

Plants, though silent and seemingly passive, communicate through subtle environmental cues. They respond to changes in temperature, humidity, and soil moisture, sending signals that reflect their well-being. The "Plant Language Translator" project takes on the challenge of deciphering these signals and translating them into human-readable messages. By employing a DHT11 sensor to monitor temperature and humidity levels and a soil moisture sensor to assess the soil's hydration, the system compiles a comprehensive dataset.

The gathered data is then meticulously organized into JSON format, which serves as the bridge for transmitting vital information to a Python-based graphical user interface. This interface, thoughtfully designed with the tkinter library, not only offers a visually appealing display but also provides a user-friendly platform for observing and interpreting the plant's well-being in real-time. This amalgamation of technology and nature facilitates more informed care for our green companions, transcending traditional care practices.

Chapter 2

Literature Survey

1) "Worldwide Auto-Mobi: Arduino IoT Home Automation System for IR Devices" by Ayad Ghany Ismaeel and Mohammed Qasim Kamal is an IEEE paper that presents the design and implementation of an Arduino-based IoT home automation system that can control IR devices worldwide. The system uses an Arduino board with an Ethernet shield to connect to the internet. The IR receiver is used to receive signals from the remote, and the IR LED is used to send signals to the appliances. The system is designed to allow users to control their home appliances such as TVs, air conditioners, and DVD players using their smartphones from anywhere in the world.

2) "DIY Smart Home IR Blaster with ESP8266" by Anurag Chugh. In This Paper Author explains how to use an ESP8266 module and an IR receiver module to receive and send IR signals wirelessly using MQTT. The advantage of this project is that it allows for the control of IR devices from a remote location using a smartphone or a computer, making it a convenient and accessible solution for smart homes. However, the disadvantage is that it requires some technical knowledge to set up, and the code may need to be modified for specific IR devices.

3) "ESP8266 Wi-Fi Remote Control for Air Conditioner" by Pedro Henrique is a project that uses an ESP8266 module, an IR receiver module, and an IR LED to create a device that can control an air conditioner remotely using WiFi. The advantage of this project is that it provides an easy and affordable solution for controlling air conditioning units from a remote location, without the need for complex and expensive smart home systems. However, the limitation is that it is specific to air conditioners and may not work with other IR devices.

4) "Arduino IR Remote Control" by Dejan Nedelkovski is a project that uses an Arduino Uno board, an IR receiver module, and an IR LED to create a device that can receive and send IR signals. The advantage of this project is that it provides a simple

and affordable solution for controlling IR devices using an Arduino board. However, the limitation is that it may not work with all IR devices and may require some modifications to the code.

5) "DIY IR Remote Control for Home Appliances with ESP8266" by Hari Nair This project uses an ESP8266 module, an IR receiver module, and an IR LED to create a device that can control home appliances remotely using WiFi. The device can be controlled using a web interface, and the project also includes instructions for creating a mobile app to control the device. The advantage of this project is that it provides a simple and cost-effective solution for remotely controlling home appliances, without the need for complex wiring or additional hardware. However, the project does require some programming skills and may not be suitable for beginners.

6) "ESP8266 Wi-Fi Controlled IR Remote" by Pranav Pai Vernekar This project uses an ESP8266 module, an IR receiver module, and an IR LED to create a device that can receive and send IR signals wirelessly using a web interface. The project includes instructions for building the circuit, programming the ESP8266, and creating the web interface. The advantage of this project is that it provides a simple and cost-effective solution for remotely controlling devices using WiFi. However, the project requires some basic knowledge of electronics and programming, and may not be suitable for beginners. Additionally, the web interface may not be as user-friendly as other remote control solutions.

7) "Design and Implementation of an Infrared Communication System using Arduino" by Aditi Chaurasia and Jatin Agrawal. In this Paper authors presented the design and implementation of an IR communication system using Arduino. The system is designed to transmit data between two Arduinos using IR communication. The IR LED is used as a transmitter, and the IR photodiode is used as a receiver. The system is easy to implement and low-cost, but it has a limited range and is susceptible to interference from other light sources. It is Low-cost and low-power method of wireless communication, suitable for short-range communication, and easy to implement with

Arduino. But it has Limited range and susceptible to interference from other light sources

8) "IR remote control for home appliances using Arduino " by Muhammad Fawad Khan, et al. In this paper the author presents the design and implementation of an IR remote control system for home appliances using Arduino. The system allows users to control home appliances such as TVs, DVD players, and air conditioners using a remote control. The IR receiver is used to receive signals from the remote, and the IR LED is used to send signals to the home appliances. The system is easy to implement and customizable, but it has a limited range and is susceptible to interference from other light sources.

9) "Arduino based IR remote controlled robot" by Akash Jaiswal and Vivek Kumar Singh. In this paper authors present the design and implementation of an IR remote controlled robot using Arduino. The robot is designed to be controlled using an IR remote control and can move in any direction. The IR receiver is used to receive signals from the remote, and the IR LED is used to send signals to the motors. The system is easy to implement and customizable, but it has a limited range and is susceptible to interference from other light sources.

10) "IR Gesture Recognition using Arduino" by Nikhil Borkar and R. K. Agrawal. In this paper authors present the design and implementation of an IR gesture recognition system using Arduino. The system is designed to recognize hand gestures using IR sensors and can be used to control various devices such as TVs and computers. The IR receiver is used to receive signals from the IR sensors, and the IR LED is used to send signals to the devices. The system is easy to implement and customizable, but it has a limited range and is susceptible to interference from other light sources.

Chapter 3

Problem Definition

The problem at hand is the inability to effectively interpret and respond to the needs of plants in a potted environment. Plants communicate their well-being through factors like soil moisture, light levels, and environmental conditions, but humans lack a comprehensive method to understand this communication. This gap hinders optimal plant care and our ability to address their requirements accurately. To tackle this issue, we aim to create a Plant Language Translator using an Arduino Uno and a suite of sensors. This system will collect real-time data on soil moisture, light exposure, temperature, and humidity, translating it into human-readable information. By doing so, we empower individuals to make informed decisions, enhancing plant health and fostering a deeper connection between humans and nature.

Chapter 4
Objectives of Work

The objectives of the project are:

- Map Different reading from sensors with valid messages.
- Interfacing Arduino with python.

Chapter 5

Models / Block Diagram

Circuit Diagram

Components used in the circuit are Breadboard, Arduino Uno R3, DHT11, Moisture Sensor, Jumper Wires, USB Type B Cable. This circuit takes 5V DC Supply from the USB Connector. DHT11 has three pins' Data (Out), Ground (GND), VCC. The Data pin is connected to Pin A1; Ground Pin is connected to GND and VCC pin is connected to VCC on the Arduino board. The Moisture Sensor has 4 pins GND, VCC, DO (Digital Out) and AO (Analog Out) and AO is connected to pin A0 on Arduino Board.

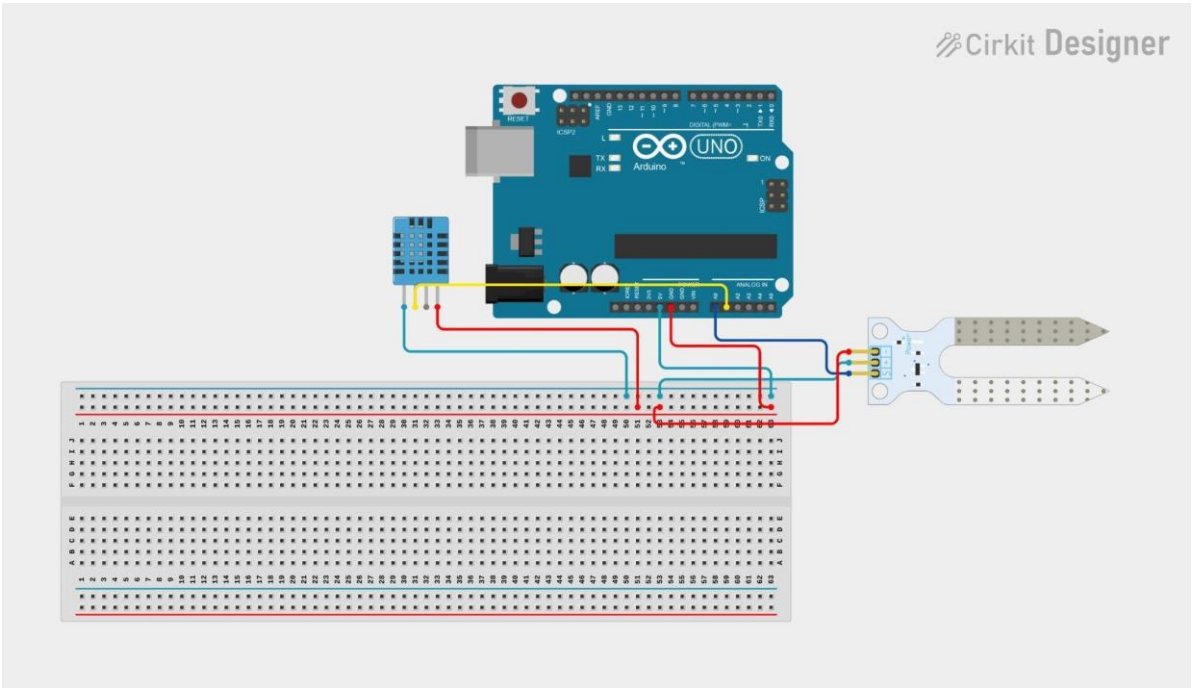


Figure: Circuit Diagram.

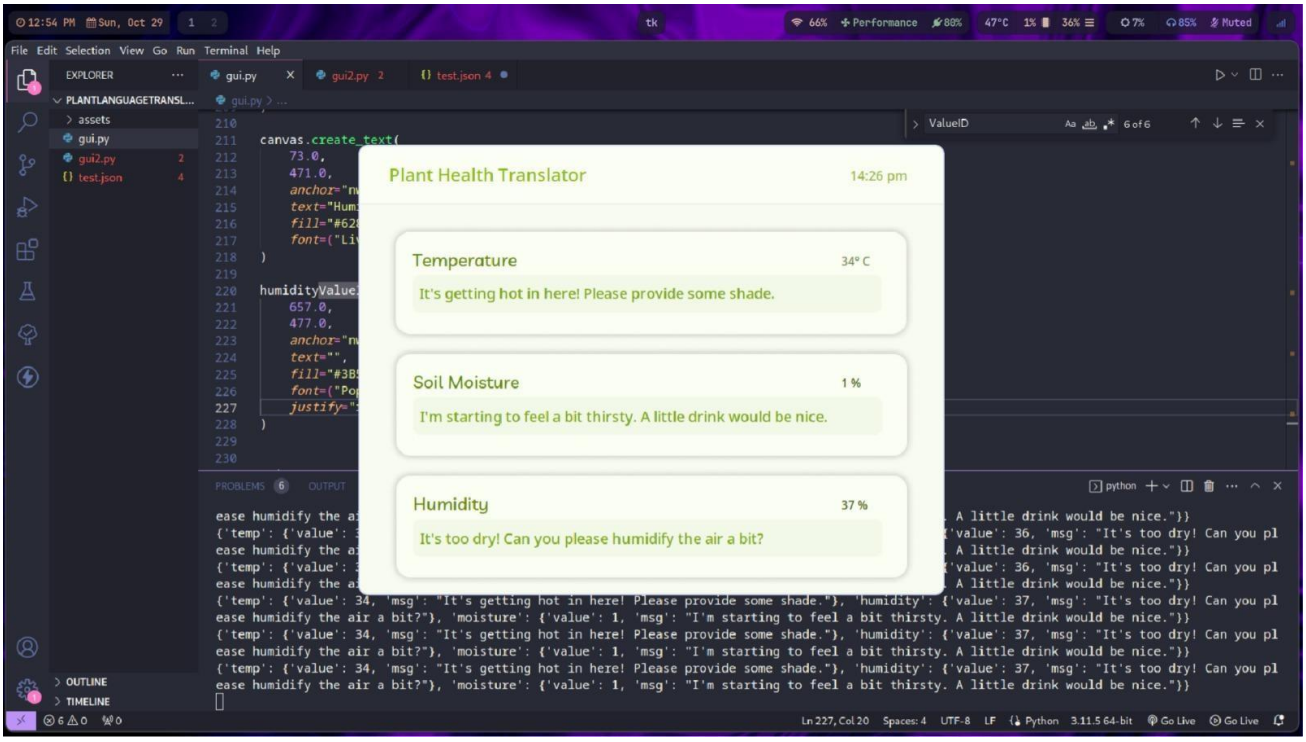

Plant Health Translator
14:26 pm
Temperature
34° C
It's getting hot in here! Please provide some shade.
Soil Moisture
1 %
I'm starting to feel a bit thirsty. A little drink would be nice.
Humidity
37 %
It's too dry! Can you please humidify the air a bit?

Figure: GUI

Chapter 6

System Requirements

Hardware Requirements

SR. No	Components	Quantity
1	Arduino Uno	1
2	DHT11	1
3	Soil moisture sensor	1
4	USB Type B Cable	1
5	Jumper Wires	9
6	Desktop / Laptop	1
7	Breadboard	1

Software Requirements

SR. No	Name
1	**Operating System:** Ubuntu / Arco Linux/ Other Linux Distributions
2	Python
3	**Python Libraries**: PySerial, pysonDB, Streamlit
4	Visual Studio Code, Arduino IDE
5	**Arduino Libraries:** IR Remote, Arduino Json

Chapter 7

Implementation

To implement our project, we followed these steps:

Designed a circuit diagram: We started by designing a circuit diagram for our project, which included a Components used in the circuit are Breadboard, Arduino Uno R3, DHT11, Moisture Sensor Jumper Wires, USB Type B Cable. This circuit takes 5V DC Supply from the USB Connector. DHT11 has three pins' Data (Out), Ground (GND), VCC. The Data pin is connected to Pin A1; Ground Pin is connected to GND and VCC pin is connected to VCC on the Arduino board. The Moisture Sensor has 4 pins GND, VCC, DO (Digital Out) and AO (Analog Out) and AO is connected to pin A0 on Arduino Board.

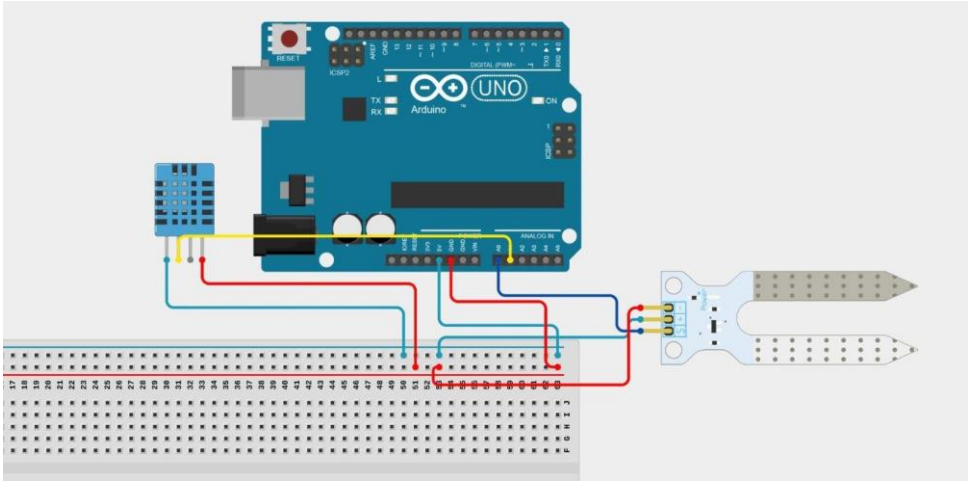


Figure: Circuit Diagram.

Programmed the Arduino: We connected the Arduino to a computer and used the Arduino IDE for programming. We integrated two libraries: ArduinoJson for handling JSON data and DHT11 for interfacing with the sensor to obtain temperature and humidity values.

Uploaded the program to Arduino: After finishing the program, we uploaded it to the Arduino using the Arduino IDE.

Created a GUI using Python Tkinter: We also created a GUI for our project using Tkinter, which is a Python library for creating Desktop applications. The GUI interacts with the Arduino using the serial port and PySerial library

Tested and refined the project: We tested the project to ensure that it worked as expected and made refinements as necessary.

Arduino Code:

```
include    "dht.h"  include <ArduinoJson.h>

 define dht_apin A1 // Analog Pin sensor is connected to  define sensor_pin A0

dht DHT;

void setup() { Serial.begin(6600); pinMode(sensor_pin, INPUT);

delay(500); // Delay to let the system boot Serial.println("Plant Sensor\n");

delay(1000); // Wait before accessing the sensors

}

int mapSoilMoisture(int sensorValue) {

// Map the sensor value from the 0-1023 range to the 0-100 range int percentage = map(sensorValue, 1023, 340, 0, 100);

percentage = constrain(percentage, 0, 100);

return percentage;

}

void loop() {

// Read soil moisture

int sensorValue = analogRead(sensor_pin);

int moisturePercentage = mapSoilMoisture(sensorValue);

// Read temperature and humidity DHT.read11(dht_apin);

// Create a JSON object to store the data

DynamicJsonDocument jsonDoc(200); // Adjust the buffer size as needed

// Store temperature data in JSON
```

```
JsonObject tempData = jsonDoc.createNestedObject("temp"); tempData["value"] = DHT.temperature;

if (DHT.temperature >= 30) {

tempData["msg"] = "It's getting hot in here! Please provide some shade.";

} else if (DHT.temperature >= 20) {

tempData["msg"] = "Ah, the temperature is just right. Thanks for keeping me comfortable!";

}

// Store humidity data in JSON

JsonObject humidityData = jsonDoc.createNestedObject("humidity"); humidityData["value"] = DHT.humidity;

if (DHT.humidity >= 70) {

humidityData["msg"] = "It's so humid! Please, a bit less moisture in the air.";

} else if (DHT.humidity >= 40) {

humidityData["msg"] = "Perfect humidity level. Thanks for keeping me cozy!";

} else {

humidityData["msg"] = "It's too dry! Can you please humidify the air a bit?";

}

// Store moisture data in JSON

JsonObject moistureData = jsonDoc.createNestedObject("moisture"); moistureData["value"] = moisturePercentage;

if (moisturePercentage >= 80) {

moistureData["msg"] = "It's so wet! No more water, please!";

} else if (moisturePercentage >= 60) {

moistureData["msg"] = "Ah, perfect moisture levels. Thanks for taking care of me!";

} else {

moistureData["msg"] = "I'm starting to feel a bit thirsty. A little drink would be nice.";

}

// Serialize the JSON to a string String jsonString; serializeJson(jsonDoc, jsonString);

// Print the JSON to the Serial port Serial.println(jsonString);

delay(1000); // Wait 5 seconds before the next reading

}
```

Tkinter GUI Code:

```
from pathlib import Path import serial

from tkinter import Tk, Canvas, Entry, Text, Button, PhotoImage import json

OUTPUT_PATH = Path(  file  ).parent

ASSETS_PATH = OUTPUT_PATH / Path(r"./assets/frame0") window = Tk()

window.geometry("766x610") window.configure(bg = " FFFFFF")

ser = None MSG_FONT_SIZE = 20

def relative_to_assets(path: str) -> Path: return ASSETS_PATH / Path(path)

def read_data(): try:

data = ser.readline().decode('utf-8').strip() window.after(1000, read_data)   Update every
1 second jsonData=json.loads(data)

print(jsonData) update_gui(jsonData)

except Exception as e:

print(f"Error reading data: {e}")

def update_gui(data): canvas.itemconfig(humidityMsgID,text=data["humidity"]["msg"])

canvas.itemconfig(soilMoistMsgID,text=data["moisture"]["msg"])
canvas.itemconfig(tempMsgID,text=data["temp"]["msg"])

canvas.itemconfig(humidityValueID,text=str(data["humidity"]["value"])      +      "      %")
canvas.itemconfig(soilMoistValueID,text=str(data["moisture"]["value"])     +      "      %")
canvas.itemconfig(tempValueID,text=str(data["temp"]["value"]) + "° C")

def main(): try:

global ser

ser = serial.Serial('/dev/ttyACM0', 6600) read_data()

window.resizable(False, False) window.mainloop()

except Exception as e:

print(f"Error opening serial port: {e}")

canvas = Canvas( window,

bg = " FFFFFF",

height = 610,

width = 766,

bd = 0,
```

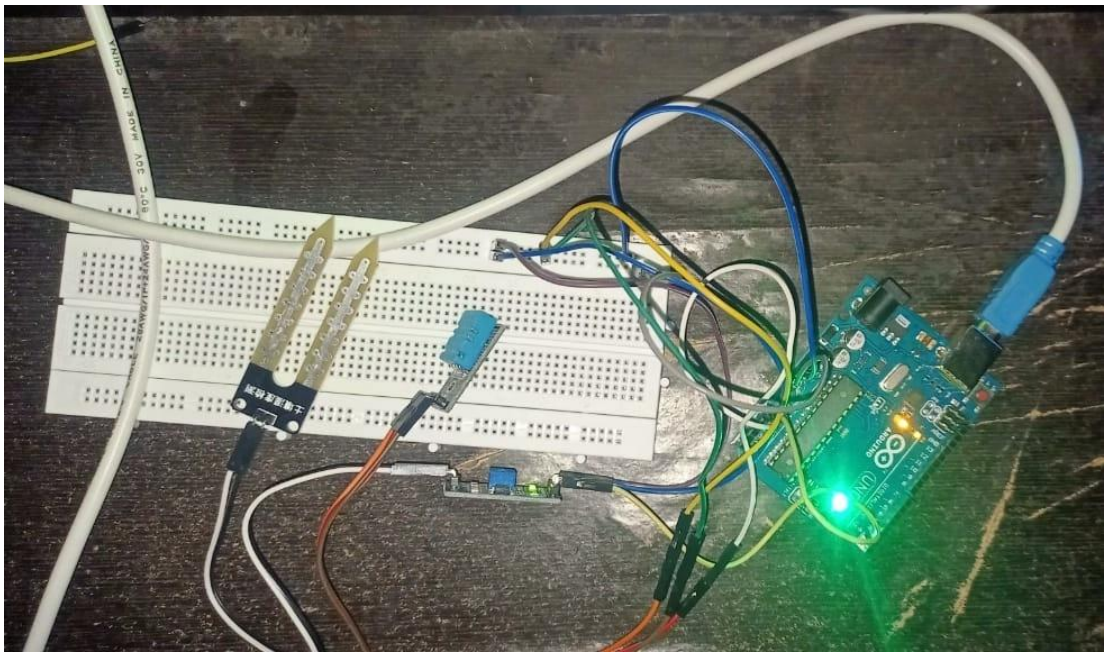
```
highlightthickness = 0, relief = "ridge"

)

canvas.place(x = 0, y = 0) canvas.create_rectangle(

1.0,

80.0,

767.0,

610.0,  fill=" F8FBF0",

outline="")

image_image_1 = PhotoImage( file=relative_to_assets("image_1.png"))

image_1 = canvas.create_image( 368.0,

40.0,

image=image_image_1

)

canvas.create_text( 40.0,

22.0,

anchor="nw",

text="Plant Language Translator", fill=" 8ABB18",

font=("MochiyPopOne Regular", 24 * -1)

)

timeID = canvas.create_text( 666.0,

27.0,

anchor="nw",

text="14:26 pm", fill=" 68B747",

font=("MochiyPopOne Regular", 18 * -1)

)

image_image_2 = PhotoImage( file=relative_to_assets("image_2.png"))

image_2 = canvas.create_image( 367.0,

186.0,

image=image_image_2

)
```

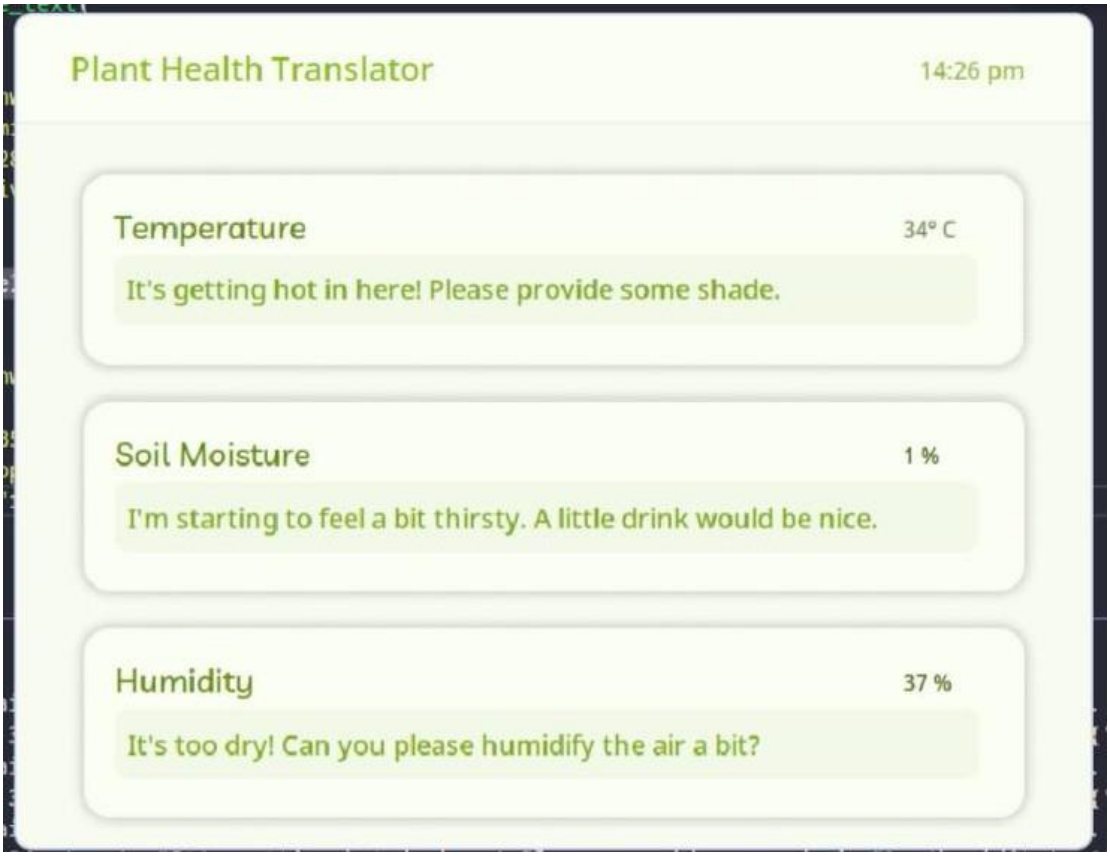
```
image_image_3 = PhotoImage( file=relative_to_assets("image_3.png"))
image_3 = canvas.create_image( 362.0,
201.0,
image=image_image_3
)
tempMsgID = canvas.create_text( 81.0,
186.0,
anchor="nw", text="", fill=" 7BA618",
font=("Livvic Regular", MSG_FONT_SIZE * -1)
)
canvas.create_text( 72.0,
136.0,
anchor="nw", text="Temperature", fill=" 628610",
font=("Livvic Medium", 24 * -1)
)
tempValueID = canvas.create_text( 657.0,
145.0,
anchor="nw", text="", fill=" 71824A",
font=("Poppins Regular", 16 * -1), justify="right")
image_image_4 = PhotoImage( file=relative_to_assets("image_4.png"))
image_4 = canvas.create_image( 368.0,
352.0,
image=image_image_4)
image_image_5 = PhotoImage( file=relative_to_assets("image_5.png"))
image_5 = canvas.create_image( 363.0,
367.0,
image=image_image_5)
soilMoistMsgID = canvas.create_text( 82.0,
354.0,
anchor="nw", text="", fill=" 7BA618",
```

```
font=("Livvic Regular", MSG_FONT_SIZE * -1))
canvas.create_text( 73.0,
305.0,
anchor="nw", text="Soil Moisture", fill=" 628610",
font=("Livvic Medium", 24 * -1) )
soilMoistValueID = canvas.create_text( 657.0,
311.0,
anchor="nw", text="", fill=" 3B5107",
font=("Poppins Regular", 16 * -1), justify="right")
image_image_6 = PhotoImage( file=relative_to_assets("image_6.png"))
image_6 = canvas.create_image( 368.0,
518.0,
image=image_image_6)
image_image_7 = PhotoImage( file=relative_to_assets("image_7.png"))
image_7 = canvas.create_image( 363.0,
533.0,
image=image_image_7)
humidityMsgID = canvas.create_text( 82.0,
520.0,
anchor="nw", text="", fill=" 7BA618",
font=("Livvic Regular", MSG_FONT_SIZE * -1))
canvas.create_text( 73.0,
471.0,
anchor="nw", text="Humidity", fill=" 628610",
font=("Livvic Medium", 24 * -1))
humidityValueID = canvas.create_text( 657.0,
477.0,
anchor="nw", text="", fill=" 3B5107",
font=("Poppins Regular", 16 * -1), justify="right")
main()
```

Chapter 8

Results and Analysis





Chapter 9

Conclusion

As we delve deeper into the intricacies of plant communication, there is great potential for harnessing this knowledge to improve agricultural practices, enhance ecosystem management, and promote environmental sustainability. By deciphering the language of plants, we can gain valuable insights into their needs, stressors, and interactions with the environment, enabling us to respond more effectively to challenges such as climate change, biodiversity loss, and food security.