UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

Design of a real time monitoring system for electrophysiological plant status

Mecanismo de Titulación: Tesis en torno a una hipótesis o problema de investigación y su contrastación

Marco Antonio Chávez Peña

Luis Miguel Procel, PhD
César Zambrano, PhD
Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito
para la obtención del título de Máster en Nanoelectrónica

Quito, 7 de enero del 2025

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

Design of a real time monitoring system for electrophysiological plant status

Marco Antonio Chávez Peña

Nombre del Director del Programa:	Luis Miguel Prócel
Título académico:	Doctor of Philosophy
Director del programa de:	Maestría en Nanoelectrónica
Nombre del Decano del colegio Académico:	Eduardo Alba
Título académico:	Doctor of Philosophy
Decano del Colegio:	Colegio de Ciencia e Ingenierías

Nombre del Decano del Colegio de Posgrados:	Dario Niebieskikwiat
Título académico:	Doctor of Physics

Quito, Enero 2025

© DERECHOS DE AUTOR

Nombre del estudiante:	Marco Antonio Chávez Peña
Código de estudiante:	00332932
C.I.:	1713311502
Lugar y fecha:	Quito, 07 de Enero de 2025.

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

A mi Dios y la Virgen María por esta oportunidad, a mis padres Jorge y Ana a los que amo mucho. A mis hermanas Paola y Cristina y mi sobrino Matías, a los cuales espero servir de ejemplo. Dios me los bendiga, por todo su apoyo y fe en mí.

AGRADECIMIENTOS

A la USFQ y al Instituto Politécnico de Toulouse por sus valiosas enseñanzas que espero poner en práctica para el beneficio de la sociedad y el medio ambiente. A mis directores de proyecto Luis Miguel Procel y César Zambrano por su apoyo en este proyecto.

RESUMEN

Este proyecto muestra un innovador sistema de monitoreo en tiempo real diseñado para evaluar la salud y las condiciones ambientales de las plantas. Aprovechando la potencia del Arduino MKR WiFi 1010 y el Arduino MKR Environmental Shield, el sistema captura parámetros ambientales críticos como la temperatura, la presión atmosférica, la humedad y la intensidad de la luz.

Además, utilizando un sensor de suelo conectado mediante protocolos RS485 y RS232, el sistema proporciona datos completos del suelo, incluidos el nivel de pH, temperatura, humedad, conductividad, nitrógeno, fósforo y potasio.

La perfecta integración del sistema con Google Sheets a través de Apps Script permite el registro de datos en tiempo real en intervalos de seis segundos. Esta conectividad permite a los usuarios interactuar dinámicamente con los datos. Al hacer clic en botones intuitivos, los usuarios pueden generar gráficos personalizados de cualquier parámetro en función del tiempo, lo que ofrece información valiosa sobre las plantas.

Esta solución combina tecnología IoT de vanguardia con usabilidad práctica, lo que la hace ideal para aplicaciones en agricultura, investigación y monitoreo ambiental. El proyecto demuestra un enfoque escalable para comprender y optimizar las condiciones de crecimiento de las plantas de una manera fácil y altamente eficiente.

Palabras clave: Arduino MKR 1010 Wifi, Arduino ENV Shield, RS485, RS232, Apps Script, IoT.

ABSTRACT

This project shows an innovative real-time monitoring system designed to assess the health and environmental conditions of plants. Taking advantage of the power of the Arduino MKR WiFi 1010 and the Arduino MKR Environmental Shield, the system captures critical environmental parameters such as temperature, atmospheric pressure, humidity, and light intensity.

Additionally, using a soil sensor connected via RS485 and RS232 protocols, the system provides comprehensive soil data, including pH level, temperature, humidity, conductivity, nitrogen, phosphorus and potassium.

The system's seamless integration with Google Sheets through Apps Script enables real-time data logging at six-second intervals. This connectivity allows users to interact dynamically with the data. By clicking intuitive buttons, users can generate customized graphs of any parameter against time, offering valuable insights into plant health trends.

This solution merges cutting-edge IoT technology with practical usability, making it ideal for applications in agriculture, research, and environmental monitoring. The project demonstrates a scalable approach to understanding and optimizing plant growth conditions in a user-friendly and highly efficient manner.

Keywords: Arduino MKR 1010 Wifi, Arduino ENV Shield, RS485, RS232, Apps Script, IoT.

TABLE OF CONTENT

TABLE INDEX

FIGURES INDEX

INTRODUCTION

State of the Art

Modern IoT-based monitoring systems have revolutionized precision agriculture by enabling the collection, transmission, and visualization of critical environmental and soil parameters. Devices like the Arduino MKR WiFi 1010, combined with communication protocols such as RS485 and RS232, ensure seamless integration of multi-sensor platforms. These technologies provide reliable and scalable solutions for real-time data acquisition and wireless transmission.

Recent advancements in sensor technology have made it possible to measure environmental factors like temperature, humidity, pressure, and light intensity alongside essential soil metrics, including pH, moisture, conductivity, and nutrient concentrations (NPK). These compact, energy-efficient sensors are crucial for understanding plant growth conditions and ensuring optimal health.

Cloud integration and data visualization have become central to modern agricultural systems. Connecting data to platforms like Google Sheets via Apps Script allows real-time access and analysis, empowering users with intuitive graphical representations and actionable insights. Such systems enable quick identification and resolution of plant stressors, optimizing decision-making and resource management.

Precision agriculture now relies heavily on multi-parameter sensing systems for targeted interventions, such as irrigation and fertilization. By providing near-continuous data collection, the prototype of this project ensures timely responses to environmental or soil changes, ultimately improving productivity and sustainability. Furthermore, the modularity and

scalability of this prototype allow for customization based on specific crops or environmental conditions, making it adaptable to diverse agricultural scenarios.

Open-source platforms like Arduino democratize access to sophisticated monitoring tools by offering cost-effective alternatives to proprietary systems. This accessibility empowers researchers and small-scale farmers to adopt advanced solutions without incurring prohibitive costs.

The prototype of this project aligns with the forefront of agricultural innovation by combining IoT technology, advanced sensors, and cloud-based analytics. Its ability to collect data every six seconds and dynamically visualize it in real-time places it among the most effective tools for optimizing agricultural productivity and monitoring plant health. This integration of cutting-edge technology positions this prototype as a vital contribution to sustainable and efficient farming practices.

REVIEW OF LITERATURE: ELECTROPHYSIOLOGICAL MONITORING OF PLANTS, INTEGRATION OF IOT FOR CONTINUOUS MONITORING, ARTIFICIAL INTELLIGENCE

Electrophysiological monitoring of plants has gained increasing attention as a method to study plant responses to environmental and physiological stress. The paper "Identifying General Stress in Commercial Tomatoes Based on Machine Learning Applied to Plant Electrophysiology" (Applied Science 2021, 10.3390); provides a critical framework for understanding how plant electrophysiology can be integrated with advanced technologies like machine learning to identify stress factors. This paper serves as a reference to position the current project within the broader context of plant health monitoring.

Electrophysiology has been recognized as a useful approach to understand plant responses to environmental changes. This involves measuring bioelectric signals that plants produce in response to factors such as water stress, nutrient deficiencies, or pest attacks. These signals serve as early indicators of stress, offering a predictive advantage over traditional methods like visual inspection or chemical analysis.

The study on commercial tomatoes highlights the potential of electrophysiological signals to detect stress conditions effectively. By analyzing bioelectric activity, the researchers demonstrated that stress could be quantified and classified using machine learning models, enabling proactive management of crop health.

Building on the concepts from the paper, this project takes advantage of IoT technology for real-time data acquisition. The use of Arduino MKR WiFi 1010 and environmental sensors enables the continuous collection of soil and atmospheric parameters alongside electrophysiological data. Unlike the paper, which focuses on tomatoes, this system is designed

for broader applications in agriculture, enabling the integration of diverse environmental and soil conditions such as pH, temperature, conductivity, and nutrient levels.

The paper emphasizes machine learning as a tool to analyze complex electrophysiological data for stress detection. Although the current project does not implement machine learning directly, it sets the stage for future integration by providing a robust platform for data collection and visualization. The recorded data in Google Sheets could be further analyzed using machine learning models to classify and predict plant stress more accurately.

The paper highlights the importance of reliable data acquisition methods for electrophysiology. Similarly, this project uses RS485 and RS232 protocols to ensure accurate and noise-resistant communication between sensors and the microcontroller. The integration of environmental monitoring shields and soil sensors complements electrophysiological monitoring by providing a comprehensive picture of plant health.

The study demonstrates the importance of translating electrophysiological data into actionable insights. The current project builds upon this by creating real-time data visualization capabilities in Google Sheets. Users can interact with the data, generate custom graphs, and analyze trends, thereby facilitating decision-making for crop management.

While the paper focuses primarily on the classification of stress in a controlled environment using machine learning, this project addresses gaps by providing a scalable, low-cost system that integrates environmental, soil, and electrophysiological parameters. The ability to visualize real-time data extends the usability of such systems beyond academic research to practical applications in commercial farming.

METHODOLOGY

The investigation method for the Real-Time Monitoring System for Electrophysiological Plant Status involved a systematic and multidisciplinary approach. It integrates hardware design, software development, and real-time monitoring of the plant. The methodology can be outlined as follows:

1. **Literature Review and Background Research**

It was conducted a thorough review of existing studies on electrophysiological plant monitoring, soil health sensors, and environmental monitoring technologies. It was studied the principles of plant electrophysiology and stress responses, taking advantage of insights from companies like Vivent and relevant academic papers.

A literature review of previous research on plant bioelectrical signals, sensor technologies, and data acquisition methods, have demonstrated that plants generate electrical responses to environmental stimuli, including light, temperature, humidity, and stress factors such as drought or disease.

2. **System Design and Development**

a. **Hardware Selection:**

It was decided to use the Arduino MKR WiFi 1010 for its Wi-Fi connectivity and compatibility with various sensors. Also, the employment of the Arduino MKR Environmental Shield for capturing temperature, humidity, pressure, and light intensity.

And the integration of a soil sensor using RS485 and RS232 protocols to measure parameters such as pH, soil temperature, moisture, conductivity, nitrogen, phosphorus, and potassium.

b. **Software Development:**

It was decided for the development of the firmware to interface with sensors, collect data, and transmission the use of Google Sheets using Apps Script. This setup allows continuous data logging, automatic timestamping, and cloud storage, making it accessible from any device.

This software development approach takes advantages of the scalability and accessibility of Google's cloud ecosystem, allowing remote access and real-time collaboration while ensuring a reliable and automated workflow for plant monitoring and analysis.

3. **Data Collection and Integration**

The system was deployed in a controlled environment to monitor plant status under varying conditions. And capture data every 6 seconds and store it in Google Sheets for analysis and visualization.

This cloud-based approach enhances remote monitoring capabilities, ensuring real-time, scalable, and easily accessible data management for plant research and precision agriculture applications.

4. **Visualization and User Interaction**

Apps Script functionalities were created to allow users to generate graphs of specific parameters versus time and the development of tools for filtering and analyzing data based on user input, such as date ranges or parameter selection.

Creates dynamic visualizations, helping researchers identify trends and anomalies in plant responses. Users can notice significant changes in plant bioelectrical activity, aiding in early stress detection

CONTENT

1. **Vivent silver electrode for electrophysiological signal measurement**

To monitor the plant's electrophysiological status, the Vivent silver electrode cable was used to establish connection with the plant. The cable captures the electrical signals generated by the plant in response to various environmental conditions. The signal was initially observed using an oscilloscope, revealing a voltage range between 1 mV and 30 mV. Due to the low magnitude of these signals, an instrumentation amplifier was employed to amplify the signal and make it suitable for further processing.

The AD620 is a high-performance instrumentation amplifier known for its precision, low power consumption, and compact design. It is ideal for applications requiring the amplification of small differential signals, such as electrophysiological monitoring.

In this project, an AD620 instrumentation amplifier module was used to measure plant electrophysiological signals. The module is equipped with two variable resistors, one for setting the gain and the other for adjusting the offset.

The AD620 offset was set to 1.5V to center the amplified signal within the range of the ADC used in subsequent processing.

AD620 amplified the plant's electrophysiological signal from the initial range of 1 mV–30 mV to a range of 10 mV–300 mV. This amplified signal was then fed into the ADC of the Arduino MKR WiFi 1010 for digitization and further analysis.

This design step ensured that the plant's weak electrophysiological signals were accurately amplified and prepared for digital processing, forming a critical part of the real-time monitoring system.

2. **Environmental Data Monitoring with MKR ENV Shield**

The MKR ENV Shield is used to capture essential environmental parameters, including temperature, humidity, atmospheric pressure, and light intensity. This shield integrates seamlessly with the Arduino MKR WiFi 1010, providing a robust solution for real-time monitoring.

The MKR ENV Shield includes an onboard SD card slot, enabling local data storage. This feature was employed to store measurements from various sensors.

The data was recorded in a structured format, including:

Timestamp: Generated using an I2C-connected real-time clock (RTC).

Plant Signal: Processed ADC values of electrophysiological measurements.

Environmental Data: Temperature, humidity, pressure, and light intensity.

Soil Parameters: pH, temperature, humidity, conductivity, nitrogen, phosphorus, and potassium levels.

The integration of the SD card ensures redundancy in data logging. By taking advantage of the MKR ENV shield, the project integrates critical environmental data into the real-time monitoring system. This data complements the plant electrophysiological measurements, providing a comprehensive overview of plant-environment interactions.

3. **Soil Sensor Integration for Detailed Analysis**

The system incorporates a soil sensor capable of measuring multiple soil parameters such as pH, temperature, humidity, electrical conductivity, nitrogen, phosphorus, and potassium

levels. This integration adds valuable insights into the plant's immediate environment, crucial for assessing plant health and behavior.

The sensor operates with a polarization voltage between 12 and 24 volts, supplied via two dedicated power lines. And the other two lines the sensor uses the RS485 protocol for transmitting data. RS485 is robust and suited for long-distance communication in noisy environments.

A conversion module is used to convert differential RS485 signals to single-ended RS232 signals, which are compatible with the MKR WiFi 1010 serial ports.

4. **Integration of Arduino MKR WiFi 1010 with Google Sheets**

For the software development, Google Apps Script was used to facilitate seamless communication between the Arduino MKR WiFi 1010 and Google Sheets. This approach enabled the real-time storage and visualization of environmental, soil, and plant electrophysiological data in the cloud.

Secure data transmission was implemented using the WiFiNINA library for HTTPS connectivity. The Apps Script endpoint was set up to handle POST requests and parse incoming JSON data. And receive the data from the Arduino in JSON format. The parsed data was appended to specific rows in Google Sheets, maintaining a chronological order with the timestamp given by the Apps Script software.

Buttons in the Google Sheet interface allow users to generate charts dynamically. Users could visualize any parameter (e.g., environmental, soil, or electrophysiological) versus time, using dedicated scripts that accessed and plotted the data.

Figure 1 represents the MKR ADC vs. time, which is used to calculate the plant's voltage. The ADC data from the Arduino MKR WiFi 1010 provides insight into the electrical

activity of the plant. This information is crucial as variations in plant voltage can indicate physiological changes related to stress, hydration, or overall health.

The next four figures correspond to the MKR Environmental Shield, which measures different environmental conditions. The MKR Environmental Shield supports multiple sensors, with its key ranges including temperature (15°C to 40°C), humidity (0% to 100%), atmospheric pressure (260 hPa to 1260 hPa), and light intensity (0 to 650 Lux). Figure 2 shows environmental temperature, an important factor affecting plant metabolism and growth. Figure 3 illustrates environmental humidity, which influences water absorption through the roots and transpiration through the leaves. Figure 4 displays atmospheric pressure, a parameter that can impact gas exchange and overall environmental stability. Figure 5 represents environmental light intensity, which is essential for photosynthesis, as light availability directly affects the plant's energy production and growth.

The remaining seven graphs focus on soil conditions, crucial for nutrient uptake and plant development. Figure 6 shows soil pH, which ranges from 3 to 9, with a resolution of 0.01 and an accuracy of ±0.3. Soil pH determines nutrient availability, as extreme values can limit the absorption of essential elements. Figure 7 represents soil moisture, measured from 0% to 100% with a resolution of 0.1%. Proper moisture levels are necessary for root function and microbial activity. Figure 8 tracks soil temperature, ranging from -40°C to 80°C, with a resolution of 0.1°C and an accuracy of ±0.5°C. Soil temperature affects root growth and enzymatic activity, playing a key role in nutrient cycling.

Figure 9 displays soil electrical conductivity, measured in micro Siemens per centimeter (µS/cm) with a range of 0 to 10,000 µS/cm. This parameter indicates soil salinity and nutrient concentration, helping to assess the balance of dissolved ions. The last three graphs relate to

the concentration of essential nutrients in the soil. Figure 10 shows nitrogen (N) concentration, figure 11 represents phosphorus (P) levels, and figure 12 displays potassium (K) concentration. Each of these macronutrients is measured in mg/Kg, ranging from 0 to 1999 mg/Kg, with a resolution of 1 mg/Kg. Nitrogen is crucial for leaf and stem growth, phosphorus supports root development and energy transfer, and potassium enhances disease resistance and fruit quality.

Together, these measurements provide a comprehensive analysis of the plant's environment and internal electrical activity. By correlating plant voltage with soil and environmental conditions, it becomes possible to detect stress factors early and optimize growth conditions effectively.

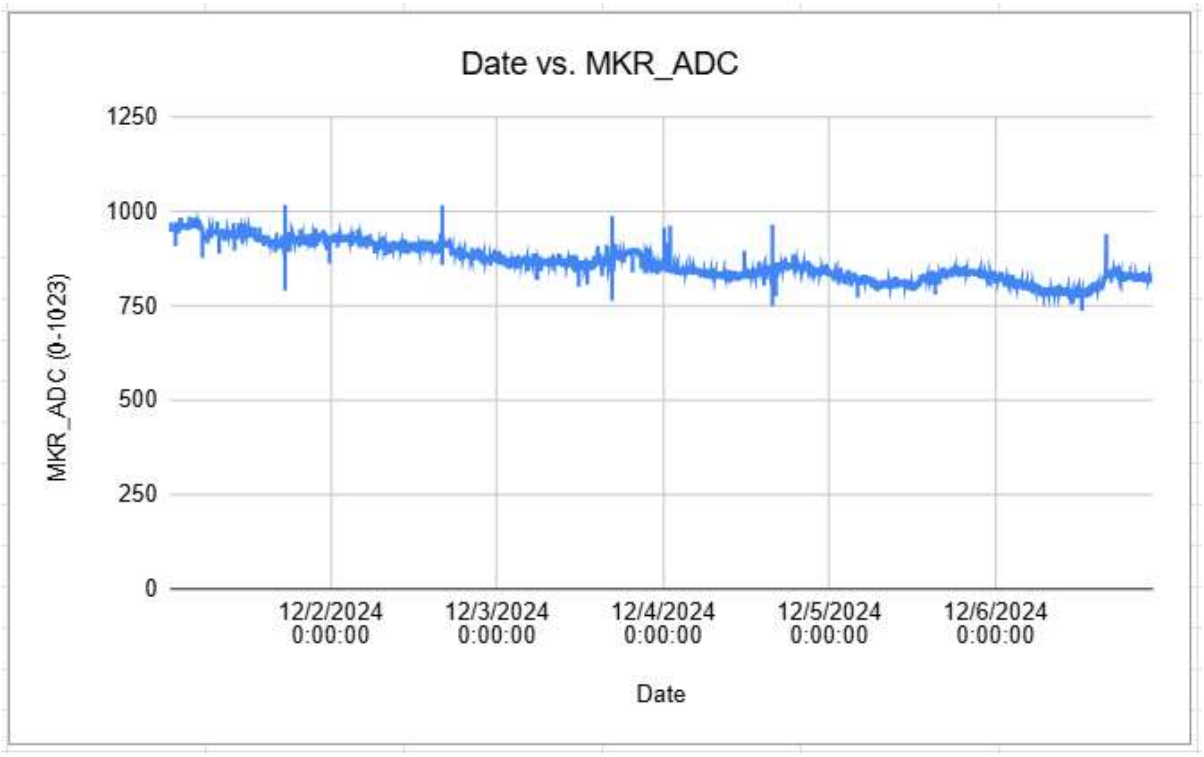

Figure 1. Date vs. MKR ADC

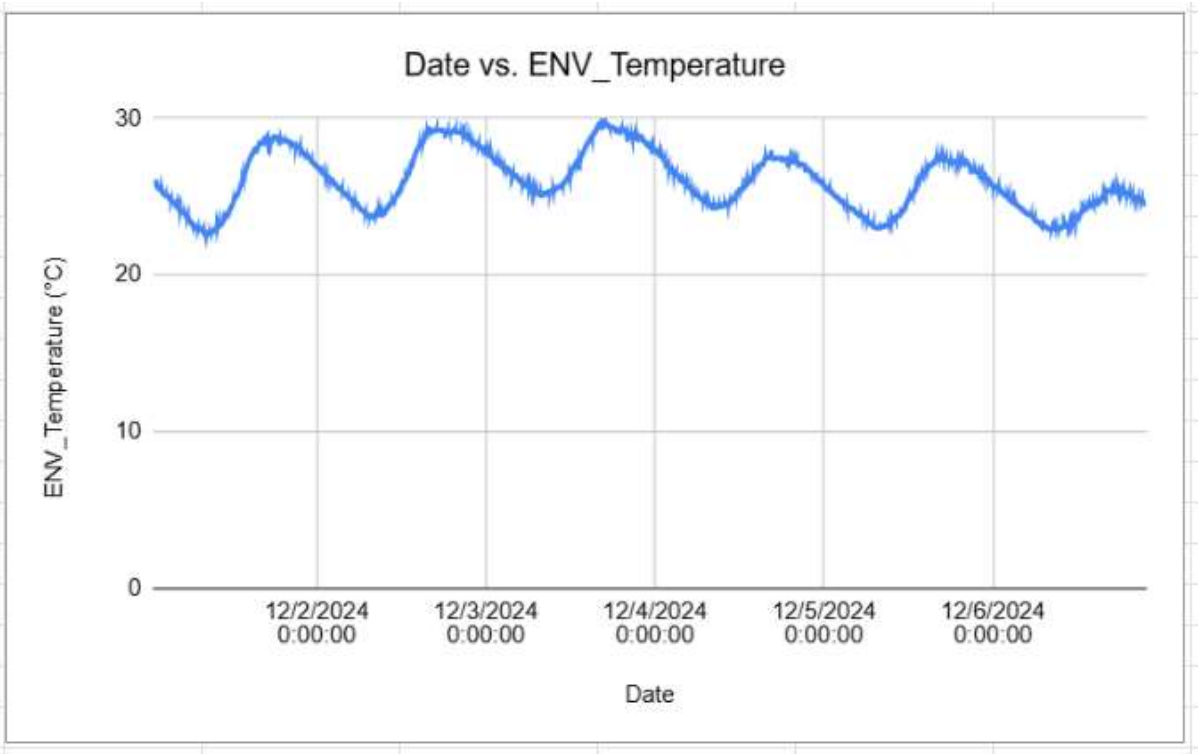

Figure 2. Date vs. Environmental Temperature

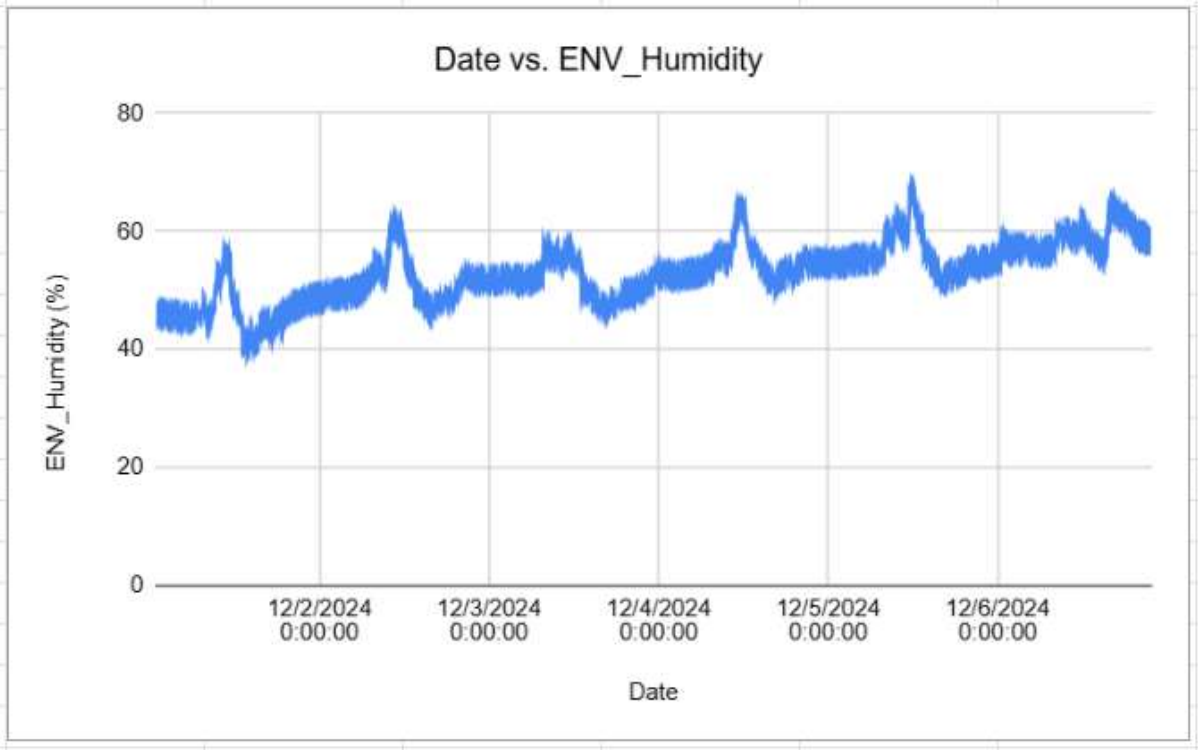

Figure 3. Date vs. Environmental Humidity

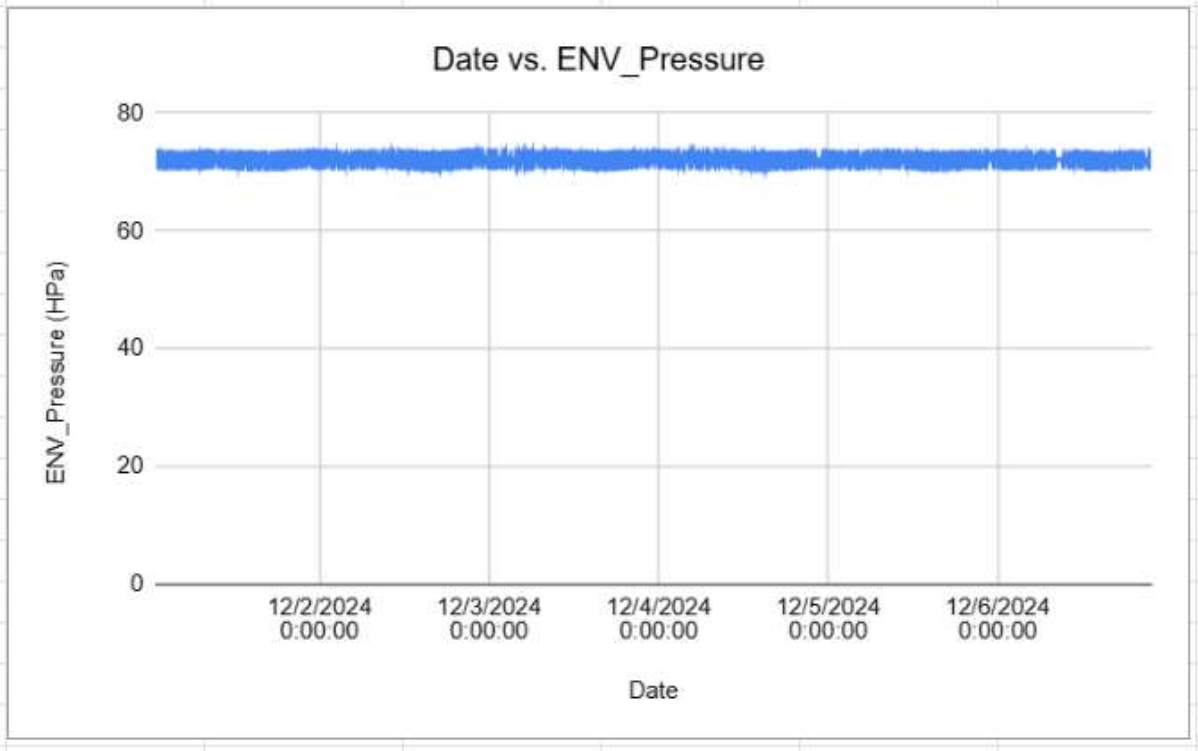

Figure 4. Date vs. Environmental Pressure

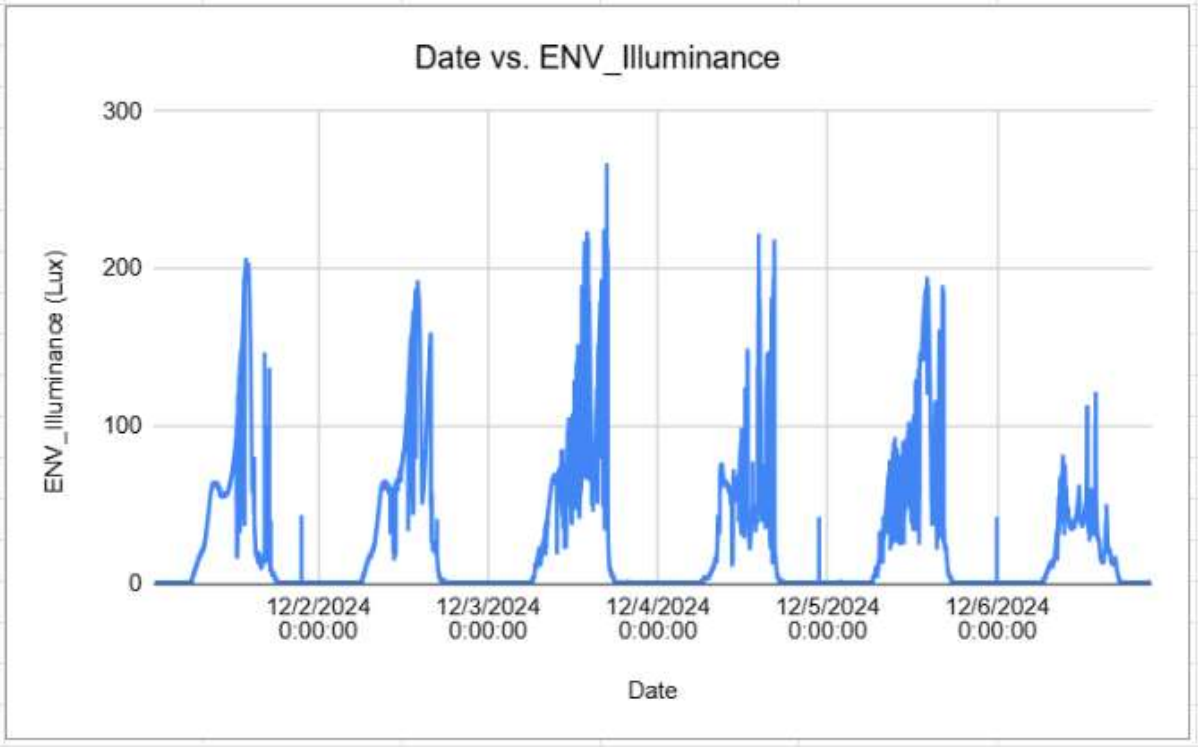

Figure 5. Date vs. Environmental Light Intensity

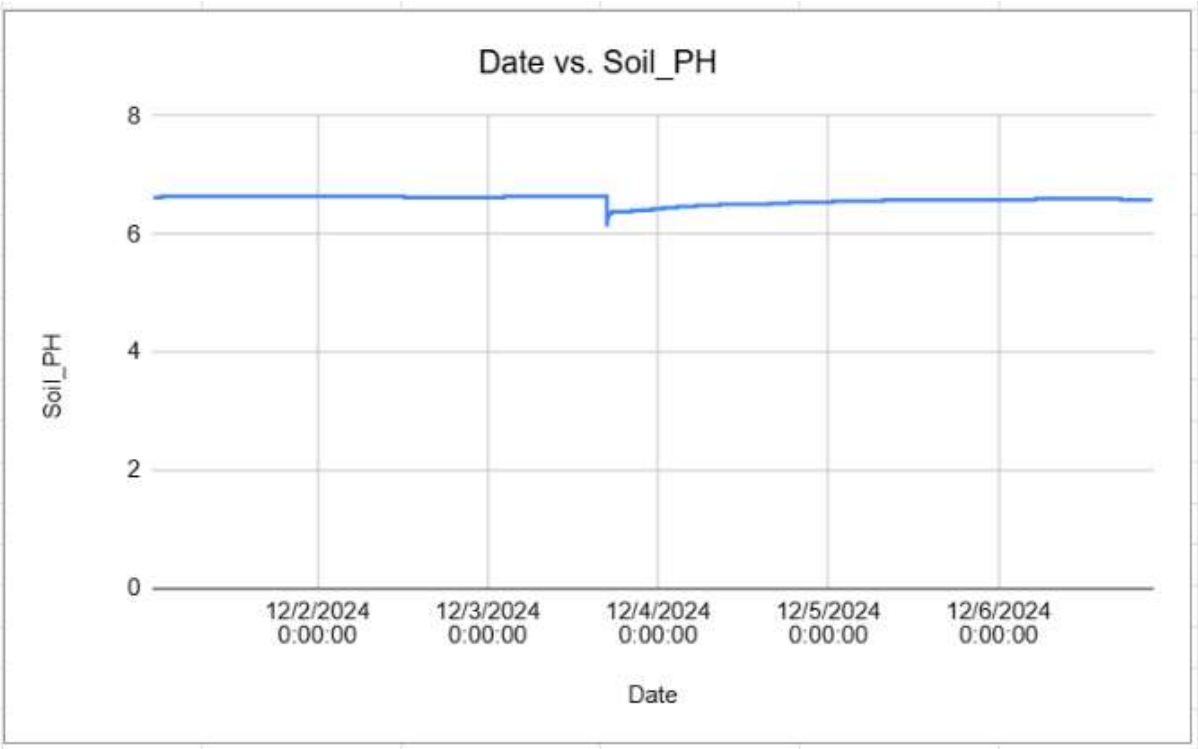

Figure 6. Date vs. Soil pH

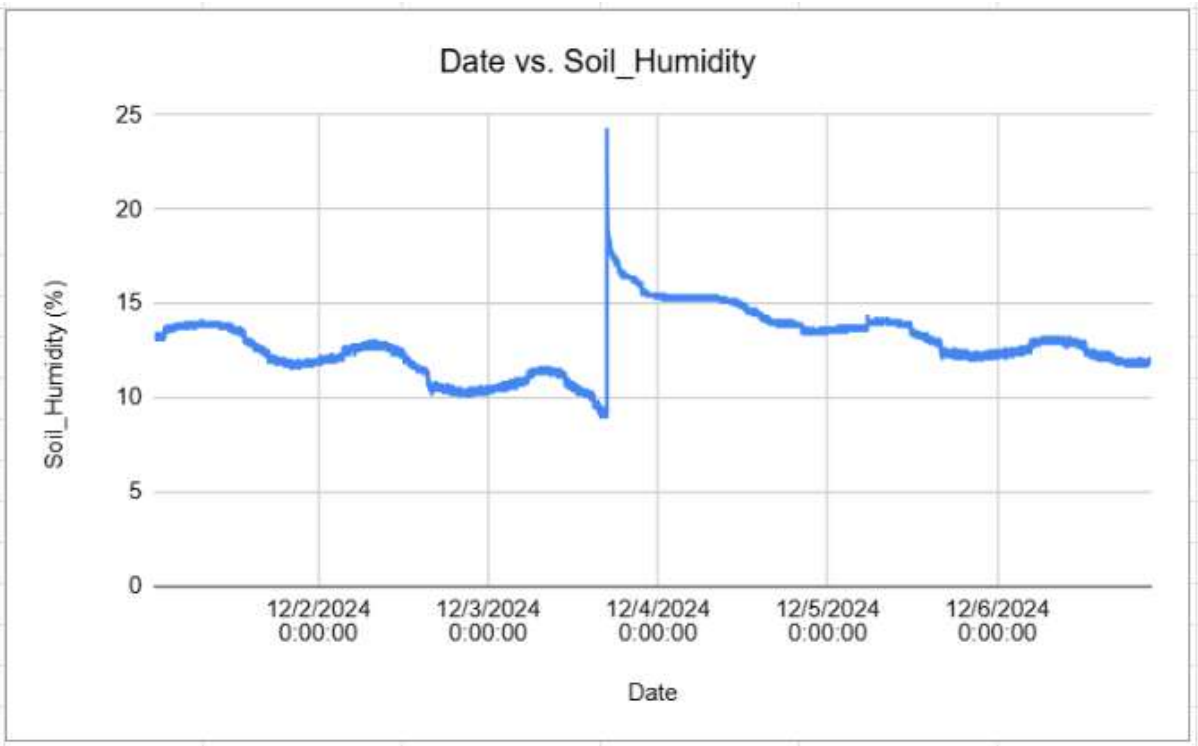

Figure 7. Date vs. Soil Humidity

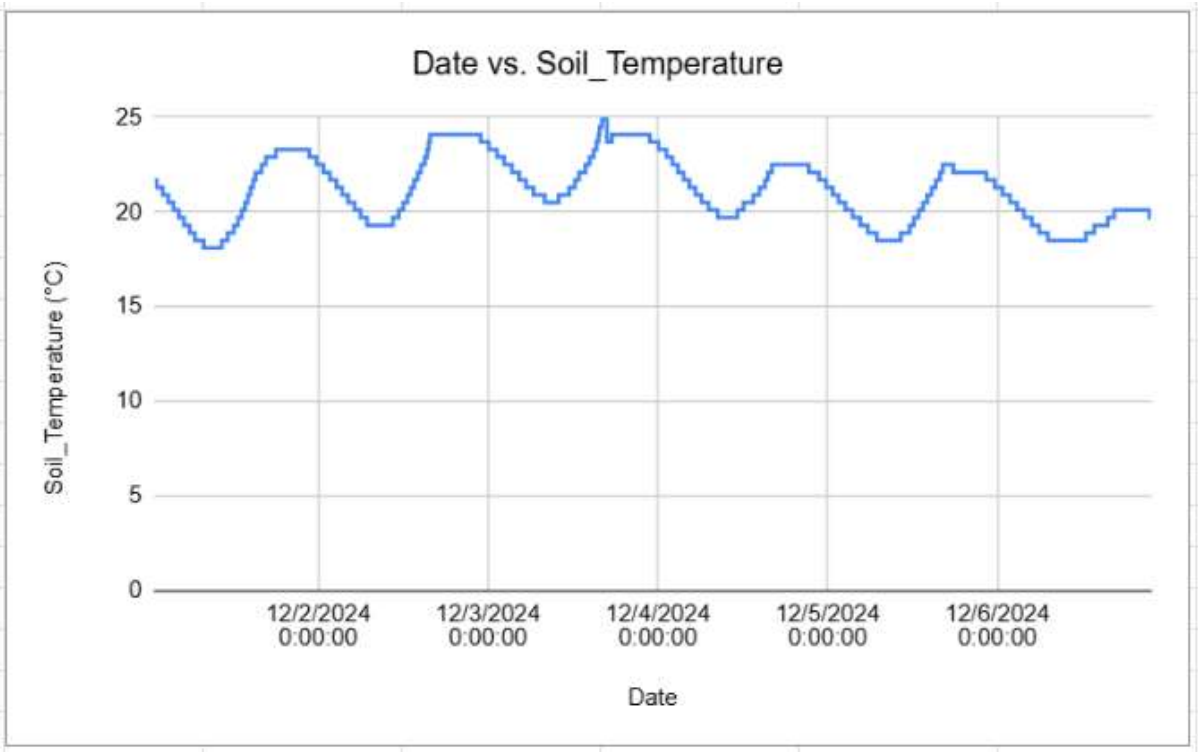

Figure 8. Date vs. Soil Temperature

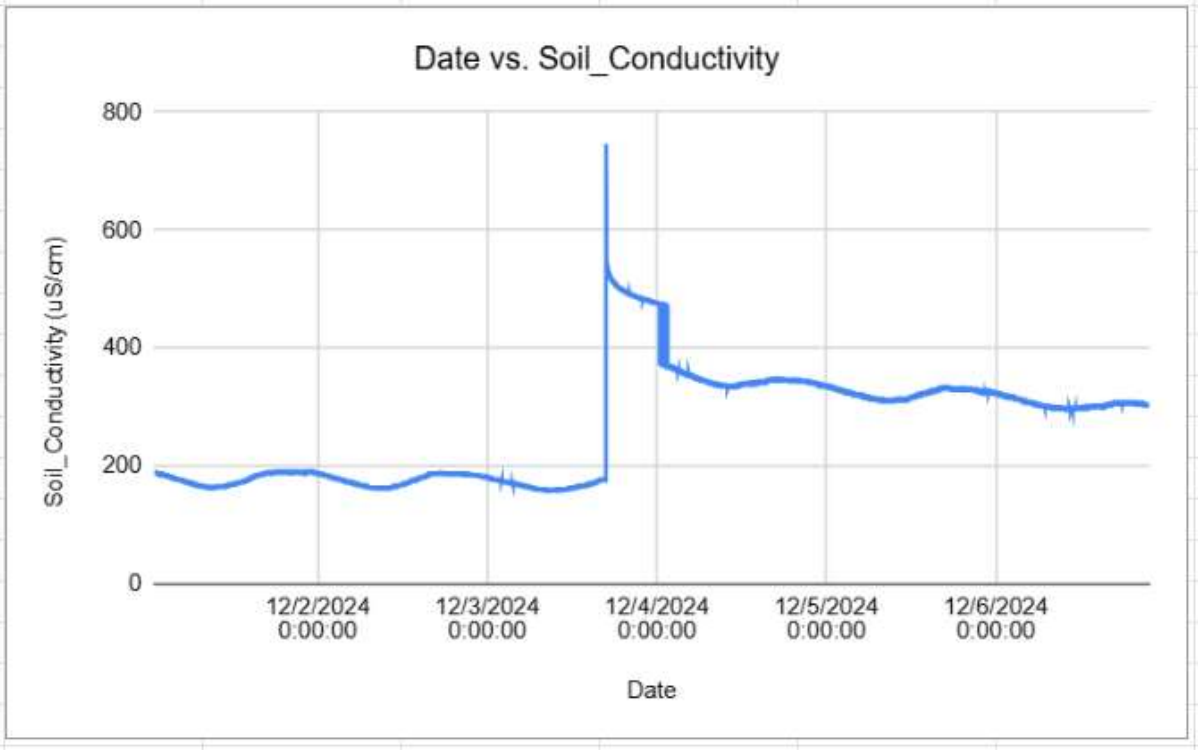

Figure 9. Date vs. Soil Conductivity

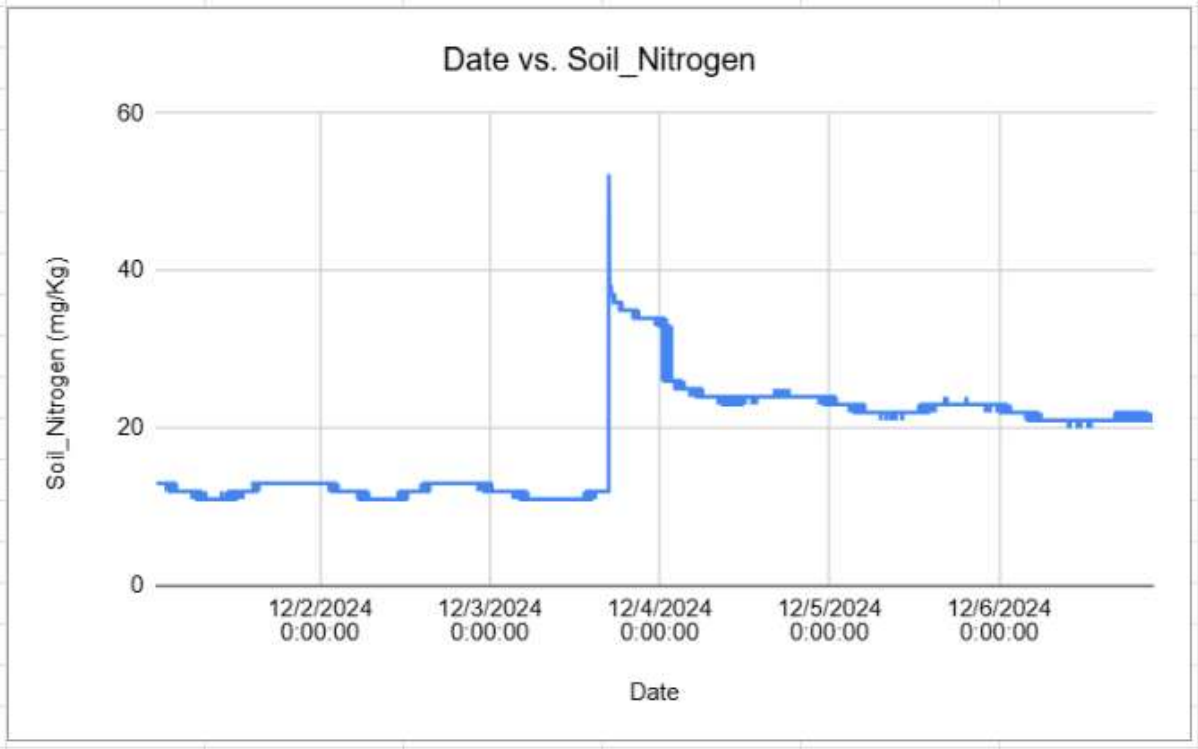

Figure 10. Date vs Soil Nitrogen

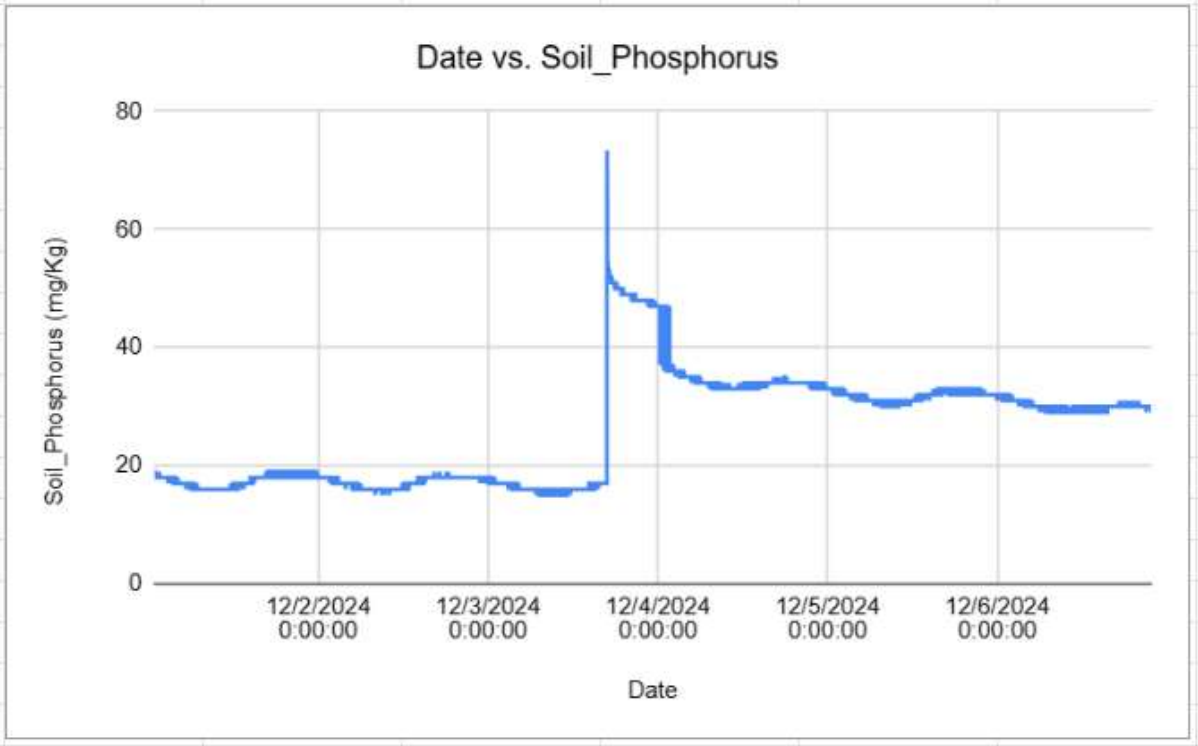

Figure 11. Date vs Soil Phosphorus

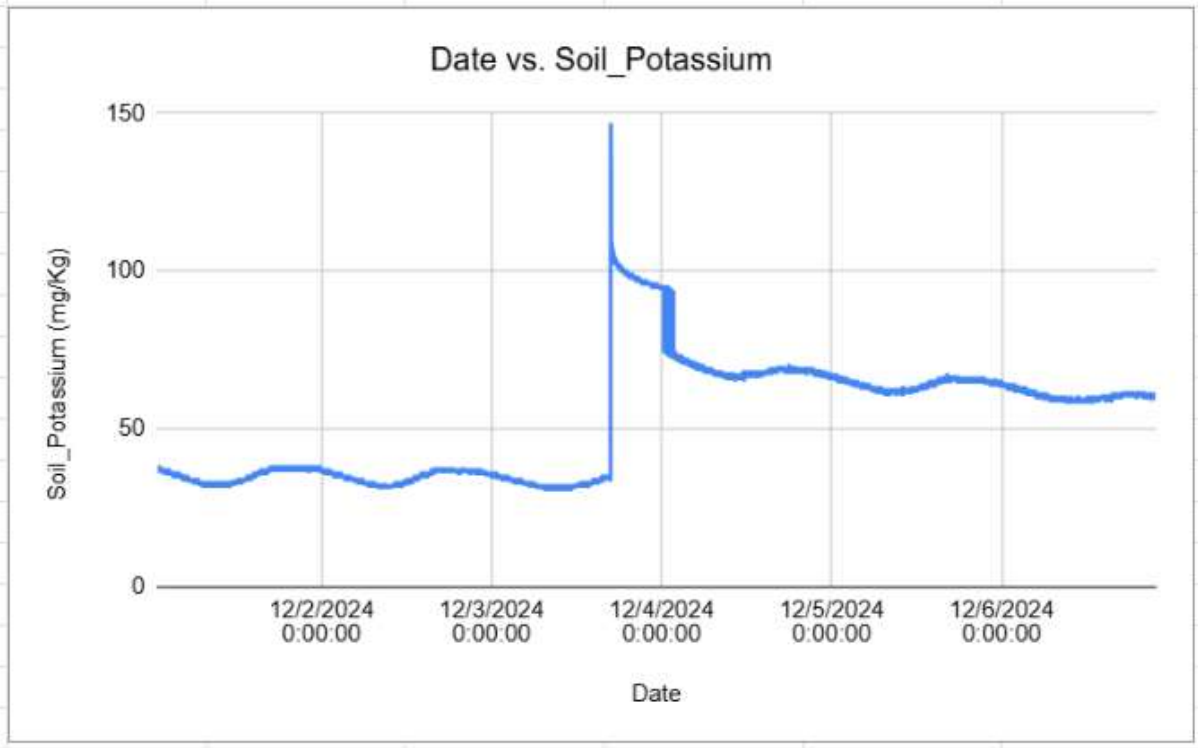

Figure 12. Date vs. Soil Potassium

The system has demonstrated robust performance, collecting and recording data consistently for 18 days without any issues. During this period, various events were monitored and analyzed through the generated graphics.

One significant event occurred when the plant was watered. The system displayed a clear and immediate increase in soil humidity, conductivity, and NPK levels, aligning with expected behavior after watering. These changes were effectively captured by the soil sensors, showcasing the accuracy and reliability of the integrated monitoring system.

However, identifying corresponding changes in the plant's voltage signals proved to be challenging. Unlike soil parameters, the plant's voltage signals did not exhibit an obvious or easily discernible pattern related to the watering event. This highlights the complexity of

interpreting electrophysiological signals and suggests that more advanced data analysis techniques may be required to extract meaningful insights from these signals.

To address this limitation, the potential use of artificial intelligence (AI) is proposed. AI techniques, such as machine learning models, could analyze the multidimensional dataset and identify subtle correlations and patterns in plant voltage signals that may indicate stress, disease, or other physiological changes. By taking advantage of AI, the system could evolve into a powerful diagnostic tool capable of providing actionable insights into the plant's health and status.

The budget used for this project is detailed in Table 1.

Component	Qty	Unit Price (USD)	Total Price (USD)	Description
Arduino MKR WiFi 1010	1	40	40	Main microcontroller for data acquisition and wireless communication.
MKR Environmental Shield	1	40	40	Measures environmental data (temperature, humidity, pressure, and light intensity).
SD Card (16 GB)	1	10	10	Used for data storage.
Soil Sensor (RS485)	1	80	80	Measures soil pH, temperature, humidity, conductivity, and NPK levels.
RS485 to RS232 Converter Module	1	12	12	Converts RS485 protocol to RS232 for soil sensor communication.
AD620 Instrumentation Amplifier Module	1	10	10	Amplifies plant voltage signals.
Vivent Silver Cable	1	50	50	Electrode cable for plant voltage measurement.
Power Supply (12-24V)	1	20	20	Provides voltage for soil sensor polarization.

Table 1. Estimated Budget for the project

The total estimated value is 262.00 US dollars.

The relatively low budget for this system highlights its potential for widespread adoption due to affordability. Unlike expensive proprietary systems, this custom-designed setup provides a cost-effective alternative while maintaining robust functionality. By leveraging accessible components and open-source platforms, this solution offers an easy implementation pathway for researchers and agricultural professionals.

Its economic design demonstrates that advanced plant monitoring systems can be developed without requiring substantial financial resources, making it particularly suitable for small-scale farmers, educational institutions, or low-budget research projects. Additionally, the modular nature of the system enables scalability and customization, further enhancing its practical appeal.

CONCLUSIONS

Current methods in electrophysiological plant monitoring often involve renting specialized equipment, such as those provided by companies like Vivent. These systems focus primarily on analyzing the plant's voltage signals, which require substantial computational resources, including advanced artificial intelligence algorithms, to interpret the data accurately.

This project introduces a comprehensive monitoring system that integrates environmental sensors, soil information, and plant voltage measurements. By combining these diverse data points, the system offers a more holistic view of the plant's condition.

The inclusion of environmental factors such as temperature, humidity, pressure, and light intensity, alongside soil parameters like pH, temperature, humidity, conductivity, nitrogen, phosphorus, and potassium, creates a multidimensional dataset. This approach has the potential to significantly enhance the detection and understanding of plant diseases or stress factors by correlating multiple variables, rather than relying solely on plant voltage signals.

This innovative system not only reduces reliance on high-cost, resource-intensive methodologies but also opens the door for real-time, low-cost, and scalable solutions in agricultural monitoring and management. It paves the way for a new era of precision agriculture that can proactively address plant health challenges with enhanced efficiency and accuracy.

REFERENCES

Najdenovska, E., Dotoit, F., Tran, D., Rochat, A., Vu, B., Mazza, M., Camps, C., Plummer, C., Wallbridge, N. & Raileanu, L., E. (2021). *Identifying Stress in Commercial Tomatoes Bases on Machine Learning Applied to Plant Electrophysiology*. Applied Sciences, https://doi.org/10.3390/app11125640.

González, D., Najdenovska, E., Dutoit, F. & Raileanu, L., E. (2023). *Detecting stress caused by nitrogen deficit using deep learning techniques applied on plant electrophysiological data*. Scientific Reports, https://doi.org/10.1038/s41598-023-36683-3.

Najdenovska, E., Dutoit, F., Tran, D., Plummer, C., Wallbridge, N., Camps, C. & Raileanu, L., E. (2021). *Classification of Plant Electrophysiology Sygnals for Detection of Spider Mites Infestation in Tomatoes.* Applied Sciences, https://doi.org/10.3390/app11041414.

Tran, D. & Camps, C. (2021). *Early Diagnosis of Iron Deficiency in Commercial Tomato Crop Using Electrical Signals*. Frontiers in Sustainable Food Systems, https://doi.org/10.3389/fsufs.2021.631529.

Arduino (2025), *Arduino Integrated Development Environment (IDE)*, Version 1.8.19, Retrieve January 7, 2025 from https://www.arduino.cc/en/software.

Arduino (2025), MKR ENV Shield rev2 (2025). Retrieve January 7, 2025 from https://docs.arduino.cc/tutorials/mkr-env-shield/mkr-env-shield-basic/

Arduino (2025), WiFiNiNa library (2025), Retrieve January 7, 2025 from https://docs.arduino.cc/libraries/wifinina/

Arduino (2025), Watchdog Library for Arduino MKR. Retrieve January 7, 2025 from https://docs.arduino.cc/libraries/adafruit-sleepydog-library/

Arduino (2025), Data Logger with MKR SD Proto Shield, Retrieve January 7, 2025 from https://docs.arduino.cc/tutorials/mkr-sd-proto-shield/mkr-sd-proto-shield-data-logger/

Arduino (2025), Adding more Seria Interfaces to SAMD microcontrollers (SERCOM), Retrieve January 7, 2025 from https://docs.arduino.cc/tutorials/communication/SamdSercom/

Atmel Corporation, ATmega 328/P Datasheet. Retrieve January 7, 2025 from https://www.microchip.com/en-us/product/atmega328p

Google, Apps Script, Retrieve January 7, 2025 from https://developers.google.com/apps-script/reference

INDEX OF ANNEXES

ANNEX A: ARDUINO CODE

```
//Library for MKR Environmental SHIELD
#include <Arduino_MKRENV.h>
//Libraries for WIFI Connection
#include <SPI.h>
#include <WiFiNINA.h>
//Library for WATCHDOG
#include <Adafruit_SleepyDog.h>
//Library to manage SD Card
#include <SD.h>
//Library to create additional UART
#include <Arduino.h>
#include "wiring_private.h"
//Library for protocol I2C
#include <Wire.h>
// DS1307 I2C address
#define DS1307_ADDRESS 0x68
// SS pin for the SD card on the ENV Shield
const int SDCARD_SS_PIN = 4;
//Instance of the SD Card
File myFile;
//Instance UART
Uart mySerial (&sercom3, 1, 0, SERCOM_RX_PAD_1, UART_TX_PAD_0);
// Buffer to store received UART data of the SOIL SENSOR
byte receivedData[7];
float temperature = 0.00;
float humidity    = 0.00;
float pressure    = 0.00;
float illuminance = 0.00;
const int analogPin = A0;
unsigned long startTime = 0;
//WIFI and SERVER information
char* ssid = "YOUR_SSID";
char* pass = "YOUR_PASSWORD";
int keyIndex = 0;
const char* urlPath = "/macros/s/YOUR_SCRIPT_ID/exec";
int status = WL_IDLE_STATUS;
char* server = "script.google.com";
//Instance of the WIFI SSL
WiFiSSLClient client;
```

```
void setup() {
  //Initialization Serial Console and Serial SERCOM
  Serial.begin(9600);
  mySerial.begin(9600);
  pinPeripheral(1, PIO_SERCOM); //Assign RX function to pin 1
  pinPeripheral(0, PIO_SERCOM); //Assign TX function to pin 0
  delay(2000);
  //Initialization MKR ENV Shield
  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
  }
  //Initialization SD Card
  Serial.print("Initializing SD card...");
  if (!SD.begin(SDCARD_SS_PIN)) {
    Serial.println("SD card initialization failed!");
    return;
  }
  Serial.println("SD card initialized.");
  //I2C Initialization
  Wire.begin();
  // Check wifi module
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
  }
  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }
  // Attempt to connect to WiFi network
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);
    // Wait 10 seconds for connection
    delay(10000);
  }
  Serial.println("Connected to wifi");
  printWiFiStatus();
  //Set WATCHDOG at 1 minute
  int timeout = Watchdog.enable(120000);
  Serial.print("Watchdog enabled with timeout: ");
  Serial.print(timeout);
  Serial.println(" milliseconds");
  Watchdog.reset();
}
```

```
void loop() {

  // Read all the sensor values
  temperature = ENV.readTemperature();
  humidity    = ENV.readHumidity();
  pressure    = ENV.readPressure();
  illuminance = ENV.readIlluminance();

  //GET DATA FROM DS1307
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(0x00); //
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ADDRESS, 7);

  byte seconds = bcdToDec(Wire.read());
  byte minutes = bcdToDec(Wire.read());
  byte hours = bcdToDec(Wire.read());
  Wire.read();
  byte day = bcdToDec(Wire.read());
  byte month = bcdToDec(Wire.read());
  byte year = bcdToDec(Wire.read());

  //GET DATA FROM SOIL SENSOR
  //Sending commands to get PH
  byte hexValues[] = {0x01, 0x03, 0x00, 0x06, 0x00, 0x01, 0x64, 0x0B};
  mySerial.write(hexValues, sizeof(hexValues));
  delay (100);
  if (mySerial.available() >= sizeof(receivedData)) {
    mySerial.readBytes(receivedData, sizeof(receivedData));
  }
  float PH =  ((receivedData[3] << 8) | receivedData[4]) / 100.0;

  //Sending commands to get Humidity
  byte hexValues1[] = {0x01, 0x03, 0x00, 0x12, 0x00, 0x01, 0x24, 0x0F};
  mySerial.write(hexValues1, sizeof(hexValues1));
  delay (100);
  if (mySerial.available() >= sizeof(receivedData)) {
    mySerial.readBytes(receivedData, sizeof(receivedData));
  }
  float Humidity =  ((receivedData[3] << 8) | receivedData[4]) / 10.0;
```

```
//Sending commands to get Temperature
byte hexValues2[] = {0x01, 0x03, 0x00, 0x13, 0x00, 0x01, 0x75, 0xCF};
mySerial.write(hexValues2, sizeof(hexValues2));
delay (100);
if (mySerial.available() >= sizeof(receivedData)) {
  mySerial.readBytes(receivedData, sizeof(receivedData));
}
float Temperature =  ((receivedData[3] << 8) | receivedData[4]) / 10.0;

//Sending commands to get Conductivity
byte hexValues3[] = {0x01, 0x03, 0x00, 0x15, 0x00, 0x01, 0x95, 0xCE};
mySerial.write(hexValues3, sizeof(hexValues3));
delay (100);
if (mySerial.available() >= sizeof(receivedData)) {
  mySerial.readBytes(receivedData, sizeof(receivedData));
}
float Conductivity =  ((receivedData[3] << 8) | receivedData[4]);

//Sending commands to get Nitrogen
byte hexValues4[] = {0x01, 0x03, 0x00, 0x1E, 0x00, 0x01, 0xE4, 0x0C};
mySerial.write(hexValues4, sizeof(hexValues4));
delay (100);
if (mySerial.available() >= sizeof(receivedData)) {
  mySerial.readBytes(receivedData, sizeof(receivedData));
}
float Nitrogen =  ((receivedData[3] << 8) | receivedData[4]);

//Sending commands to get Phosphorus
byte hexValues5[] = {0x01, 0x03, 0x00, 0x1F, 0x00, 0x01, 0xB5, 0xCC};
mySerial.write(hexValues5, sizeof(hexValues5));
delay (100);
if (mySerial.available() >= sizeof(receivedData)) {
  mySerial.readBytes(receivedData, sizeof(receivedData));
}
float Phosphorus =  ((receivedData[3] << 8) | receivedData[4]);

//Sending commands to get Potassium
byte hexValues6[] = {0x01, 0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xC0};
mySerial.write(hexValues6, sizeof(hexValues6));
delay (100);
if (mySerial.available() >= sizeof(receivedData)) {
  mySerial.readBytes(receivedData, sizeof(receivedData));
}
float Potassium =  ((receivedData[3] << 8) | receivedData[4]);
```

```
//GET DATA FROM ADC
int adcValue = analogRead(analogPin);
//STORE DATE, SOIL SENSOR DATA AND ADC IN SD CARD
myFile = SD.open("AGRO5.txt", FILE_WRITE);
if (myFile) {
  myFile.print(day);          myFile.print("/");
  myFile.print(month);        myFile.print("/");
  myFile.print(year);         myFile.print(" ");
  myFile.print(hours);        myFile.print(":");
  myFile.print(minutes);      myFile.print(":");
  myFile.print(seconds);      myFile.print(" , ");

  myFile.print(temperature);  myFile.print(" , ");
  myFile.print(humidity);     myFile.print(" , ");
  myFile.print(pressure);     myFile.print(" , ");
  myFile.print(illuminance);  myFile.print(" , ");

  myFile.print(PH);           myFile.print(" , ");
  myFile.print(Humidity);     myFile.print(" , ");
  myFile.print(Temperature);  myFile.print(" , ");
  myFile.print(Conductivity); myFile.print(" , ");
  myFile.print(Nitrogen);     myFile.print(" , ");
  myFile.print(Phosphorus);   myFile.print(" , ");
  myFile.print(Potassium);    myFile.print(" , ");

  myFile.println(adcValue);

  myFile.close();
}
else {
  Serial.println("error opening AGRO5.txt");
}
//CREATE JSON DATA TO SEND TO THE SERVER
String jsonPayload = "{\"value1\":  "  + String(temperature) +
                     ",\"value2\": "  + String(humidity) +
                     ",\"value3\": "  + String(pressure) +
                     ",\"value4\": "  + String(illuminance) +
                     ",\"value5\": "  + String(PH) +
                     ",\"value6\": "  + String(Humidity) +
                     ",\"value7\": "  + String(Temperature) +
                     ",\"value8\": "  + String(Conductivity) +
                     ",\"value9\": "  + String(Nitrogen) +
                     ",\"value10\": " + String(Phosphorus) +
                     ",\"value11\": " + String(Potassium) +
                     ",\"value12\": " + String(adcValue) +
                     "}";
Serial.println(jsonPayload);
delay(200);
```

```
  //START CONNECTION WITH THE SERVER
  Serial.println("\nStarting connection to server...");
  if (client.connect(server, 443)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    client.println("POST " + String(urlPath) + " HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(jsonPayload.length());
    client.println();
    client.println(jsonPayload);
  }
  //WAIT FOR THE SERVER RESPONSE
  delay(1000);
 //IF NOT RESPONSE FROM THE SERVER ACTIVATES WATCHDOG
  startTime = millis();
  while ((millis() - startTime < 6000) && (client.available())) {
    char c = client.read();
    Serial.write(c);
    Watchdog.reset();
  }
  delay(200);
  //SERVER DESCONNECTION
  Serial.println("disconnecting from server.");
  client.stop();
  delay(200);
}

// Attach the interrupt handler to the SERCOM
void SERCOM3_Handler()
{
  mySerial.IrqHandler();
}

// Convert BCD to decimal
byte bcdToDec(byte val) {
  return (val / 16 * 10) + (val % 16);
}
```

```
void printWiFiStatus() {
  //SSID of the network
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  //IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  //Received signal strength
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

ANEXO B: APPS SCRIPT CODE

```
//Function to listen, obtain and append the data
function doPost(e) {
  try {
    // Parse the incoming JSON payload from the POST request
    var payload = JSON.parse(e.postData.contents);

    // Access the Google Sheet by ID
    const spreadsheet = SpreadsheetApp.openById("YOUR_SHEET_ID");
    const dataSheet  = spreadsheet.getSheetByName("Sheet1");

    // Extract the data from the payload
    var MKR_Temperature = payload.value1 || "";
    var MKR_Humidity = payload.value2 || "";
    var MKR_Pressure = payload.value3 || "";
    var MKR_Illuminance = payload.value4 || "";
    var Soil_PH = payload.value5 || "";
    var Soil_Humidity = payload.value6 || "";
    var Soil_Temperature = payload.value7 || "";
    var Soil_Conductivity = payload.value8 || "";
    var Soil_Nitrogen = payload.value9 || "";
    var Soil_Phosphorus = payload.value10 || "";
    var Soil_Potassium = payload.value11 || "";
    var MKR_ADC = payload.value12 || "";

    // Append the data as a new row
    dataSheet.appendRow([new Date(), MKR_Temperature, MKR_Humidity,
    MKR_Pressure, MKR_Illuminance, Soil_PH, Soil_Humidity, Soil_Temperature,
    Soil_Conductivity, Soil_Nitrogen, Soil_Phosphorus, Soil_Potassium,
    MKR_ADC]);

    // Return a success response
    return ContentService.createTextOutput("Row appended successfully");

  } catch (error) {
    // Handle errors and return an error message
    return ContentService.createTextOutput("Error: " + error.message);
  }
}

```

```
40  //This code is to make graphics from the DATA
41  function updateChart() {
42
43    const spreadsheet = SpreadsheetApp.openById("YOUR_SHEET_ID")
44    const dataSheet  = spreadsheet.getSheetByName("Sheet1");
45    const chartSheet  = spreadsheet.getSheetByName("Sheet2");
46    const lastRow = dataSheet.getLastRow();
47
48    // Define the range for the graphic base on value I4
49    const cellValue = chartSheet.getRange("I4").getValue();
50    //1 day 14400 2 days 28800
51    const startRow = Math.max(1, lastRow - cellValue);
52    const endRow = lastRow;
53
54    // For the graphis is used the date and the respective data column
55    const xRange = dataSheet.getRange(`A${startRow}:A${endRow}`);
56    const yRange = dataSheet.getRange(`M${startRow}:M${endRow}`);
57
58    const xRange1 = dataSheet.getRange(`A${startRow}:A${endRow}`);
59    const yRange1 = dataSheet.getRange(`B${startRow}:B${endRow}`);
60
61    const xRange2 = dataSheet.getRange(`A${startRow}:A${endRow}`);
62    const yRange2 = dataSheet.getRange(`C${startRow}:C${endRow}`);
63
64    const xRange3 = dataSheet.getRange(`A${startRow}:A${endRow}`);
65    const yRange3 = dataSheet.getRange(`D${startRow}:D${endRow}`);
66
67    const xRange4 = dataSheet.getRange(`A${startRow}:A${endRow}`);
68    const yRange4 = dataSheet.getRange(`E${startRow}:E${endRow}`);
69
70    const xRange5 = dataSheet.getRange(`A${startRow}:A${endRow}`);
71    const yRange5 = dataSheet.getRange(`F${startRow}:F${endRow}`);
72
73    const xRange6 = dataSheet.getRange(`A${startRow}:A${endRow}`);
74    const yRange6 = dataSheet.getRange(`G${startRow}:G${endRow}`);
75
76    const xRange7 = dataSheet.getRange(`A${startRow}:A${endRow}`);
77    const yRange7 = dataSheet.getRange(`H${startRow}:H${endRow}`);
78
79    const xRange8 = dataSheet.getRange(`A${startRow}:A${endRow}`);
80    const yRange8 = dataSheet.getRange(`I${startRow}:I${endRow}`);
```

```
81
82    const xRange9 = dataSheet.getRange(`A${startRow}:A${endRow}`);
83    const yRange9 = dataSheet.getRange(`J${startRow}:J${endRow}`);
84
85    const xRange10 = dataSheet.getRange(`A${startRow}:A${endRow}`);
86    const yRange10 = dataSheet.getRange(`K${startRow}:K${endRow}`);
87
88    const xRange11 = dataSheet.getRange(`A${startRow}:A${endRow}`);
89    const yRange11 = dataSheet.getRange(`L${startRow}:L${endRow}`);
90
91    // Are there graphics on the sheet?
92    const charts = chartSheet.getCharts();
93    if (charts.length > 0) {
94
95      //GRAPHIC MKR_ADC
96      const chart = charts[0];
97      const newChart = chart
98        .modify()
99        .setChartType(Charts.ChartType.LINE)
100       .clearRanges()
101       .addRange(xRange)
102       .addRange(yRange)
103       .setPosition(2, 2, 0, 0)
104       .setOption('title', 'Date vs. MKR_ADC')
105       .setOption('titleTextStyle', {
106        bold: false,
107        fontSize: 16,
108        alignment: 'center'
109        })
110       .setOption('hAxis.title', 'Date')
111       .setOption('vAxis.title', 'MKR_ADC (0-1023)')
112       .build();
113     chartSheet.updateChart(newChart);
114
```

```
115     //GRAPHIC ENVIRONMENTAL TEMPERATURE
116     const chart1 = charts[1];
117     const newChart1 = chart1
118       .modify()
119       .setChartType(Charts.ChartType.LINE)
120       .clearRanges()
121       .addRange(xRange1)
122       .addRange(yRange1)
123       .setPosition(21, 2, 0, 0)
124       .setOption('title', 'Date vs. ENV_Temperature')
125       .setOption('titleTextStyle', {
126        bold: false,
127        fontSize: 16,
128        alignment: 'center'
129        })
130       .setOption('hAxis.title', 'Date')
131       .setOption('vAxis.title', 'ENV_Temperature (°C)')
132       .build();
133     chartSheet.updateChart(newChart1);
134
135     //GRAPHIC ENVIRONMENTAL HUMIDITY
136     const chart2 = charts[2];
137     const newChart2 = chart2
138       .modify()
139       .setChartType(Charts.ChartType.LINE)
140       .clearRanges()
141       .addRange(xRange2)
142       .addRange(yRange2)
143       .setPosition(40, 2, 0, 0)
144       .setOption('title', 'Date vs. ENV_Humidity')
145       .setOption('titleTextStyle', {
146        bold: false,
147        fontSize: 16,
148        alignment: 'center'
149        })
150       .setOption('hAxis.title', 'Date')
151       .setOption('vAxis.title', 'ENV_Humidity (%)')
152       .build();
153     chartSheet.updateChart(newChart2);
154
```

```
155       //GRAPHIC ENVIRONMENTAL PRESSURE
156       const chart3 = charts[3];
157       const newChart3 = chart3
158         .modify()
159         .setChartType(Charts.ChartType.LINE)
160         .clearRanges() // Clear previous ranges
161         .addRange(xRange3)
162         .addRange(yRange3)
163         .setPosition(59, 2, 0, 0)
164         .setOption('title', 'Date vs. ENV_Pressure')
165         .setOption('titleTextStyle', {
166          bold: false,
167          fontSize: 16,
168          alignment: 'center'
169          })
170         .setOption('hAxis.title', 'Date')
171         .setOption('vAxis.title', 'ENV_Pressure (HPa)')
172         .build();
173       chartSheet.updateChart(newChart3);
174
175       //GRAPHIC ENVIRONMENTAL ILLUMINANCE
176       const chart4 = charts[4];
177       const newChart4 = chart4
178         .modify()
179         .setChartType(Charts.ChartType.LINE)
180         .clearRanges()
181         .addRange(xRange4)
182         .addRange(yRange4)
183         .setPosition(78, 2, 0, 0)
184         .setOption('title', 'Date vs. ENV_Illuminance')
185         .setOption('titleTextStyle', {
186          bold: false,
187          fontSize: 16,
188          alignment: 'center'
189          })
190         .setOption('hAxis.title', 'Date')
191         .setOption('vAxis.title', 'ENV_Illuminance (Lux)')
192         .build();
193       chartSheet.updateChart(newChart4);
194
```

```
195       //GRAPHIC SOIL PH
196       const chart5 = charts[5];
197       const newChart5 = chart5
198         .modify()
199         .setChartType(Charts.ChartType.LINE)
200         .clearRanges()
201         .addRange(xRange5)
202         .addRange(yRange5)
203         .setPosition(97, 2, 0, 0)
204         .setOption('title', 'Date vs. Soil_PH')
205         .setOption('titleTextStyle', {
206          bold: false,
207          fontSize: 16,
208          alignment: 'center'
209          })
210         .setOption('hAxis.title', 'Date')
211         .setOption('vAxis.title', 'Soil_PH')
212         .build();
213       chartSheet.updateChart(newChart5);
214
215       //GRAPHIC SOIL HUMIDITY
216       const chart6 = charts[6];
217       const newChart6 = chart6
218         .modify()
219         .setChartType(Charts.ChartType.LINE)
220         .clearRanges()
221         .addRange(xRange6)
222         .addRange(yRange6)
223         .setPosition(116, 2, 0, 0)
224         .setOption('title', 'Date vs. Soil_Humidity')
225         .setOption('titleTextStyle', {
226          bold: false,
227          fontSize: 16,
228          alignment: 'center'
229          })
230         .setOption('hAxis.title', 'Date')
231         .setOption('vAxis.title', 'Soil_Humidity (%)')
232         .build();
233       chartSheet.updateChart(newChart6);
234
```

```
235     //GRAPHIC SOIL TEMPERATURE
236     const chart7 = charts[7];
237     const newChart7 = chart7
238       .modify()
239       .setChartType(Charts.ChartType.LINE)
240       .clearRanges()
241       .addRange(xRange7)
242       .addRange(yRange7)
243       .setPosition(135, 2, 0, 0)
244       .setOption('title', 'Date vs. Soil_Temperature')
245       .setOption('titleTextStyle', {
246        bold: false,
247        fontSize: 16,
248        alignment: 'center'
249        })
250       .setOption('hAxis.title', 'Date')
251       .setOption('vAxis.title', 'Soil_Temperature (°C)')
252       .build();
253     chartSheet.updateChart(newChart7);
254
255     //GRAPHIC SOIL CONDUCTIVITY
256     const chart8 = charts[8];
257     const newChart8 = chart8
258       .modify()
259       .setChartType(Charts.ChartType.LINE)
260       .clearRanges()
261       .addRange(xRange8)
262       .addRange(yRange8)
263       .setPosition(154, 2, 0, 0)
264       .setOption('title', 'Date vs. Soil_Conductivity')
265       .setOption('titleTextStyle', {
266        bold: false,
267        fontSize: 16,
268        alignment: 'center'
269        })
270       .setOption('hAxis.title', 'Date')
271       .setOption('vAxis.title', 'Soil_Conductivity (uS/cm)')
272       .build();
273     chartSheet.updateChart(newChart8);
274
```

```
275     //GRAPHIC SOIL NITROGEN
276     const chart9 = charts[9];
277     const newChart9 = chart9
278       .modify()
279       .setChartType(Charts.ChartType.LINE)
280       .clearRanges()
281       .addRange(xRange9)
282       .addRange(yRange9)
283       .setPosition(173, 2, 0, 0)
284       .setOption('title', 'Date vs. Soil_Nitrogen')
285       .setOption('titleTextStyle', {
286        bold: false,
287        fontSize: 16,
288        alignment: 'center'
289        })
290       .setOption('hAxis.title', 'Date')
291       .setOption('vAxis.title', 'Soil_Nitrogen (mg/Kg)')
292       .build();
293     chartSheet.updateChart(newChart9);
294
295     //GRAPHIC SOIL PHOSPHORUS
296     const chart10 = charts[10];
297     const newChart10 = chart10
298       .modify()
299       .setChartType(Charts.ChartType.LINE)
300       .clearRanges()
301       .addRange(xRange10)
302       .addRange(yRange10)
303       .setPosition(192, 2, 0, 0)
304       .setOption('title', 'Date vs. Soil_Phosphorus')
305       .setOption('titleTextStyle', {
306        bold: false,
307        fontSize: 16,
308        alignment: 'center'
309        })
310       .setOption('hAxis.title', 'Date')
311       .setOption('vAxis.title', 'Soil_Phosphorus (mg/Kg)')
312       .build();
313     chartSheet.updateChart(newChart10);
314
```

```
315       //GRAPHIC SOIL POTASSIUM
316       const chart11 = charts[11];
317       const newChart11 = chart11
318         .modify()
319         .setChartType(Charts.ChartType.LINE)
320         .clearRanges()
321         .addRange(xRange11)
322         .addRange(yRange11)
323         .setPosition(211, 2, 0, 0)
324         .setOption('title', 'Date vs. Soil_Potassium')
325         .setOption('titleTextStyle', {
326          bold: false,
327          fontSize: 16,
328          alignment: 'center'
329          })
330         .setOption('hAxis.title', 'Date')
331         .setOption('vAxis.title', 'Soil_Potassium (mg/Kg)')
332         .build();
333       chartSheet.updateChart(newChart11);
334
335     } else {
336
337     //CREATING GRAPHIC MKR_ADC FOR THE FIRST TIME
338     const xRange = dataSheet.getRange(`A${startRow}:A${endRow}`);
339     const yRange = dataSheet.getRange(`M${startRow}:M${endRow}`);
340     const chart = chartSheet.newChart()
341       .setChartType(Charts.ChartType.LINE)
342       .addRange(xRange)
343       .addRange(yRange)
344       .setPosition(2, 2, 0, 0)
345       .setOption('title', 'Date vs. MKR_ADC')
346       .setOption('titleTextStyle', {
347        bold: false,
348        fontSize: 16,
349        color: '#000000',
350        alignment: 'center'
351     })
352       .setOption('hAxis.title', 'Date')
353       .setOption('vAxis.title', 'MKR_ADC (0-1023)')
354       .build();
355     chartSheet.insertChart(chart);
356     Logger.log("MKR_ADC was created.");
357
```

```
358    //CREATING GRAPHIC ENV TEMPERATURE FOR THE FIRST TIME
359    const xRange1 = dataSheet.getRange(`A${startRow}:A${endRow}`);
360    const yRange1 = dataSheet.getRange(`B${startRow}:B${endRow}`);
361    const chart1 = chartSheet.newChart()
362      .setChartType(Charts.ChartType.LINE)
363      .addRange(xRange1)
364      .addRange(yRange1)
365      .setPosition(21, 2, 0, 0)
366      .setOption('title', 'Date vs. ENV_Temperature')
367      .setOption('titleTextStyle', {
368       bold: false,
369       fontSize: 16,
370       color: '#000000',
371       alignment: 'center'
372    })
373      .setOption('hAxis.title', 'Date')
374      .setOption('vAxis.title', 'ENV_Temperature (°C)')
375      .build();
376    chartSheet.insertChart(chart1);
377    Logger.log("ENV_Temp was created.");
378
379    //CREATING GRAPHIC ENV HUMIDITY FOR THE FIRST TIME
380    const xRange2 = dataSheet.getRange(`A${startRow}:A${endRow}`);
381    const yRange2 = dataSheet.getRange(`C${startRow}:C${endRow}`);
382    const chart2 = chartSheet.newChart()
383      .setChartType(Charts.ChartType.LINE)
384      .addRange(xRange2)
385      .addRange(yRange2)
386      .setPosition(40, 2, 0, 0)
387      .setOption('title', 'Date vs. ENV_Humidity')
388      .setOption('titleTextStyle', {
389       bold: false,
390       fontSize: 16,
391       color: '#000000',
392       alignment: 'center'
393    })
394      .setOption('hAxis.title', 'Date')
395      .setOption('vAxis.title', 'ENV_Humidity (%)')
396      .build();
397    chartSheet.insertChart(chart2);
398    Logger.log("ENV_Humidity was created.");
399
```

```
400   //CREATING GRAPHIC ENV PRESSURE FOR THE FIRST TIME
401   const xRange3 = dataSheet.getRange(`A${startRow}:A${endRow}`);
402   const yRange3 = dataSheet.getRange(`D${startRow}:D${endRow}`);
403   const chart3 = chartSheet.newChart()
404     .setChartType(Charts.ChartType.LINE)
405     .addRange(xRange3)
406     .addRange(yRange3)
407     .setPosition(59, 2, 0, 0)
408     .setOption('title', 'Date vs. ENV_Pressure')
409     .setOption('titleTextStyle', {
410      bold: false,
411      fontSize: 16,
412      color: '#000000',
413      alignment: 'center'
414   })
415     .setOption('hAxis.title', 'Date')
416     .setOption('vAxis.title', 'ENV_Pressure (HPa)')
417     .build();
418   chartSheet.insertChart(chart3);
419   Logger.log("ENV_Pressure was created.");
420
421   //CREATING GRAPHIC ENV ILLUMINANCE FOR THE FIRST TIME
422   const xRange4 = dataSheet.getRange(`A${startRow}:A${endRow}`);
423   const yRange4 = dataSheet.getRange(`E${startRow}:E${endRow}`);
424   const chart4 = chartSheet.newChart()
425     .setChartType(Charts.ChartType.LINE)
426     .addRange(xRange4)
427     .addRange(yRange4)
428     .setPosition(78, 2, 0, 0)
429     .setOption('title', 'Date vs. ENV_Illuminance')
430     .setOption('titleTextStyle', {
431      bold: false,
432      fontSize: 16,
433      color: '#000000',
434      alignment: 'center'
435   })
436     .setOption('hAxis.title', 'Date')
437     .setOption('vAxis.title', 'ENV_Illuminance (Lux)')
438     .build();
439   chartSheet.insertChart(chart4);
440   Logger.log("ENV_Illuminance was created.");
441
```

```
442    //CREATING GRAPHIC SOIL PH FOR THE FIRST TIME
443    const xRange5 = dataSheet.getRange(`A${startRow}:A${endRow}`);
444    const yRange5 = dataSheet.getRange(`F${startRow}:F${endRow}`);
445    const chart5 = chartSheet.newChart()
446      .setChartType(Charts.ChartType.LINE)
447      .addRange(xRange5)
448      .addRange(yRange5)
449      .setPosition(97, 2, 0, 0)
450      .setOption('title', 'Date vs. Soil_PH')
451      .setOption('titleTextStyle', {
452       bold: false,
453       fontSize: 16,
454       color: '#000000',
455       alignment: 'center'
456    })
457      .setOption('hAxis.title', 'Date')
458      .setOption('vAxis.title', 'Soil_PH')
459      .build();
460    chartSheet.insertChart(chart5);
461    Logger.log("SOIL PH was created.");
462
463    //CREATING GRAPHIC SOIL HUMIDITY FOR THE FIRST TIME
464    const xRange6 = dataSheet.getRange(`A${startRow}:A${endRow}`);
465    const yRange6 = dataSheet.getRange(`G${startRow}:G${endRow}`);
466    const chart6 = chartSheet.newChart()
467      .setChartType(Charts.ChartType.LINE)
468      .addRange(xRange6)
469      .addRange(yRange6)
470      .setPosition(116, 2, 0, 0)
471      .setOption('title', 'Date vs. Soil_Humidity')
472      .setOption('titleTextStyle', {
473       bold: false,
474       fontSize: 16,
475       color: '#000000',
476       alignment: 'center'
477    })
478      .setOption('hAxis.title', 'Date')
479      .setOption('vAxis.title', 'Soil_Humidity (%)')
480      .build();
481    chartSheet.insertChart(chart6);
482    Logger.log("SOIL HUMIDITY was created.");
483
```

```
484   //CREATING GRAPHIC SOIL TEMPERATURE FOR THE FIRST TIME
485   const xRange7 = dataSheet.getRange(`A${startRow}:A${endRow}`);
486   const yRange7 = dataSheet.getRange(`H${startRow}:H${endRow}`);
487   const chart7 = chartSheet.newChart()
488     .setChartType(Charts.ChartType.LINE)
489     .addRange(xRange7)
490     .addRange(yRange7)
491     .setPosition(135, 2, 0, 0)
492     .setOption('title', 'Date vs. Soil_Temperature')
493     .setOption('titleTextStyle', {
494      bold: false,
495      fontSize: 16,
496      color: '#000000',
497      alignment: 'center'
498   })
499     .setOption('hAxis.title', 'Date')
500     .setOption('vAxis.title', 'Soil_Temperature (°C)')
501     .build();
502   chartSheet.insertChart(chart7);
503   Logger.log("SOIL TEMPERATURE was created.");
504
505   //CREATING GRAPHIC SOIL CONDUCTIVITY FOR THE FIRST TIME
506   const xRange8 = dataSheet.getRange(`A${startRow}:A${endRow}`);
507   const yRange8 = dataSheet.getRange(`I${startRow}:I${endRow}`);
508   const chart8 = chartSheet.newChart()
509     .setChartType(Charts.ChartType.LINE)
510     .addRange(xRange8)
511     .addRange(yRange8)
512     .setPosition(154, 2, 0, 0)
513     .setOption('title', 'Date vs. Soil_Conductivity')
514     .setOption('titleTextStyle', {
515      bold: false,
516      fontSize: 16,
517      color: '#000000',
518      alignment: 'center'
519      })
520     .setOption('hAxis.title', 'Date')
521     .setOption('vAxis.title', 'Soil_Conductivity (uS/cm)')
522     .build();
523   chartSheet.insertChart(chart8);
524   Logger.log("SOIL CONDUCTIVITY was created.");
525
```

```
526    //CREATING GRAPHIC SOIL NITROGEN FOR THE FIRST TIME
527    const xRange9 = dataSheet.getRange(`A${startRow}:A${endRow}`);
528    const yRange9 = dataSheet.getRange(`J${startRow}:J${endRow}`);
529    const chart9 = chartSheet.newChart()
530      .setChartType(Charts.ChartType.LINE)
531      .addRange(xRange9)
532      .addRange(yRange9)
533      .setPosition(173, 2, 0, 0)
534      .setOption('title', 'Date vs. Soil_Nitrogen')
535      .setOption('titleTextStyle', {
536       bold: false,
537       fontSize: 16,
538       color: '#000000',
539       alignment: 'center'
540    })
541      .setOption('hAxis.title', 'Date')
542      .setOption('vAxis.title', 'Soil_Nitrogen (mg/Kg)')
543      .build();
544    chartSheet.insertChart(chart9);
545    Logger.log("SOIL NITROGEN was created.");
546
547    //CREATING GRAPHIC SOIL PHOSPHORUS FOR THE FIRST TIME
548    const xRange10 = dataSheet.getRange(`A${startRow}:A${endRow}`);
549    const yRange10 = dataSheet.getRange(`K${startRow}:K${endRow}`);
550    const chart10 = chartSheet.newChart()
551      .setChartType(Charts.ChartType.LINE)
552      .addRange(xRange10)
553      .addRange(yRange10)
554      .setPosition(192, 2, 0, 0)
555      .setOption('title', 'Date vs. Soil_Phosphorus')
556      .setOption('titleTextStyle', {
557       bold: false,
558       fontSize: 16,
559       color: '#000000',
560       alignment: 'center'
561    })
562      .setOption('hAxis.title', 'Date')
563      .setOption('vAxis.title', 'Soil_Phosphorus (mg/Kg)')
564      .build();
565    chartSheet.insertChart(chart10);
566    Logger.log("SOIL PHOSPHORUS was created.");
567
```

```
568     //CREATING GRAPHIC SOIL POTASSIUM FOR THE FIRST TIME
569     const xRange11 = dataSheet.getRange(`A${startRow}:A${endRow}`);
570     const yRange11 = dataSheet.getRange(`L${startRow}:L${endRow}`);
571     const chart11 = chartSheet.newChart()
572       .setChartType(Charts.ChartType.LINE)
573       .addRange(xRange11)
574       .addRange(yRange11)
575       .setPosition(211, 2, 0, 0)
576       .setOption('title', 'Date vs. Soil_Potassium')
577       .setOption('titleTextStyle', {
578        bold: false,
579        fontSize: 16,
580        color: '#000000',
581        alignment: 'center'
582     })
583       .setOption('hAxis.title', 'Date')
584       .setOption('vAxis.title', 'Soil_Potassium (mg/Kg)')
585       .build();
586     chartSheet.insertChart(chart11);
587     Logger.log("SOIL POTASSIUM was created.");
588   }
589   }
590
591   //Function to delete all Charts
592   function deleteAllCharts() {
593     const spreadsheet = SpreadsheetApp.openById("YOUR_SHEET_ID")
594     const chartSheet  = spreadsheet.getSheetByName("Sheet2");
595     const chartsToDelete = chartSheet.getCharts();
596     // Remove each chart
597       chartsToDelete.forEach((chart) => {
598         chartSheet.removeChart(chart);
599       });
600     SpreadsheetApp.getUi().alert("All charts have been deleted.");
601   }
602
```

ANEXO C: ELECTRONIC SCHEMATIC