SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

A Constituent College of

JSS SCIENCE & TECHNOLOGY UNIVERSITY

Arduino Data Logger (CSV) with Sensors and Python

Project report submitted in partial fulfillment of curriculum prescribed for the event-2 of Internet of Things (20CS630) course for the award of the degree of

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

Submitted by

S.No	Name	Roll.No	USN
1.	Chethan K	34	01JST21CS030
2.	K Ramachandra Shenoy	13	01JCE21CS047

D-SECTION

Under the Guidance of

SHWETHASHREE G C

Assistant Professor

Department of CS & E,

SJCE, JSSSTU, Mysuru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

May 2024

Table of Contents

Introduction

In the modern era, the Internet of Things (IoT) has revolutionized how we interact with and monitor the world around us. IoT enables the interconnection of everyday objects to the internet, allowing them to send and receive data. This capability is transformative for a wide range of applications, including environmental monitoring, industrial automation, smart homes, and healthcare. The cornerstone of IoT lies in the effective collection, analysis, and utilization of data from various sensors. These sensors gather vital information, which can then be analyzed to make informed decisions, automate processes, and improve efficiencies.

This project aims to harness the power of IoT by developing a robust data logger system using an Arduino Uno, a rotary potentiometer, and a photoresistor (light sensor), along with a Python script. The system will capture real-time data from these sensors, display it on the Arduino Serial Monitor, and save it in CSV files using Python. CSV (Comma Separated Values) files are a widely accepted format for data storage and are compatible with numerous data analysis tools such as Microsoft Excel, Google Sheets, and advanced data analytics software.

Objectives

The main objective of this project is to create a reliable data logging system that captures and stores real-time sensor data from an Arduino Uno. The stored data in CSV format can be easily accessed and analyzed for various purposes. The projects guides us through the entire process, including:

1. Sensor Data Acquisition: Setting up an Arduino Uno with a rotary potentiometer and a photoresistor to measure parameters like rotational position and light intensity.
2. Real-time Data Monitoring: Programming the Arduino to read and display sensor values on the Serial Monitor.
3. Data Transmission: Using the Python Serial module to establish communication between the Arduino and a computer.
4. Data Logging: Developing a Python script to capture sensor data from the Serial Monitor and save it into a CSV file.

Literature Review

1. **"A Low-Cost Data Acquisition System for Environmental Monitoring using Arduino and SD Card":**

   By A.A. Adetunji et al. (2017): This paper explores a cost-effective system for environmental data logging using Arduino and SD cards, emphasizing the creation of CSV files for further analysis.

   Challenges faced:

   a) Sensor Accuracy and Calibration: Sensors may have inherent inaccuracies or require calibration for specific environmental conditions.

   b) Power Consumption: Battery life for portable systems can be a constraint, requiring optimization of code and sensor usage.

2. **"Design and Implementation of a Wireless Sensor Network for Environmental Monitoring using Arduino and XBee":**

   By S.R. Kandukuri et al. (2015): This research focuses on a wireless sensor network utilizing Arduino and XBee modules, with data stored in CSV format on SD cards for remote monitoring.

   Challenges faced:

   a) Data Validation: Ensuring the collected data is within expected ranges and free from noise or errors.

   b) Data Loss Prevention: Safeguarding against data loss due to unexpected power outages, SD card malfunctions, or code bugs.

3. **"Development of a Portable Data Logger for Monitoring Agricultural Parameters using Arduino and GSM":**

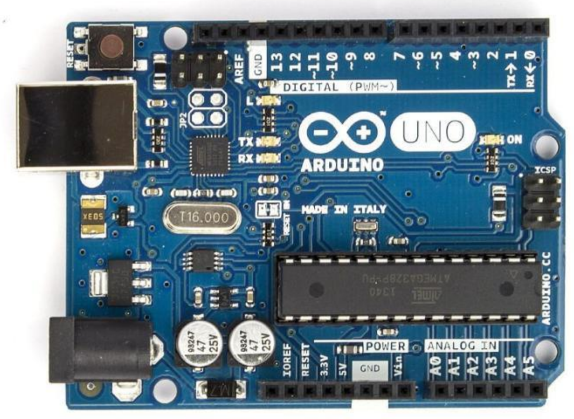
   By M.A. Rahman et al. (2016): This paper details a portable data logger for agricultural applications, utilizing Arduino and GSM communication, with data logging in CSV format for data analysis.

   Challenges faced:

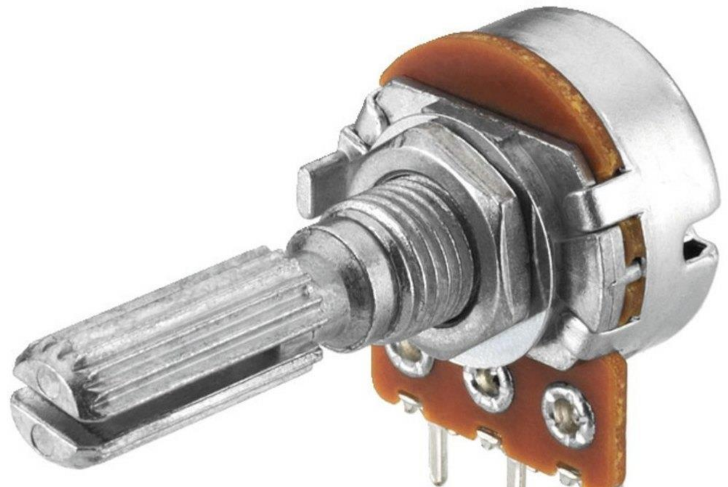
   a) Debugging and Troubleshooting: Identifying and fixing errors or unexpected behaviors in the Arduino code.

   b) Data Analysis Integration: Ensuring the generated CSV files are compatible with intended data analysis software.

Components Used

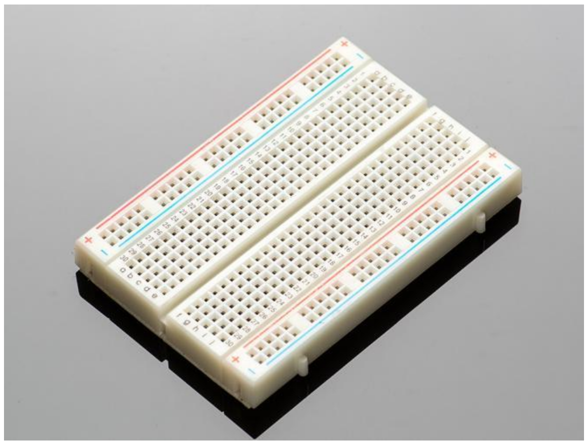


- Arduino UNO: The Arduino Uno is a popular open-source micro controller board based on the ATmega328P chip. It features 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. The board is programmable via the Arduino IDE, which supports a simplified version of C++. Arduino UNO, based on the ATmega328 micro-controller.



Rotary Potentiometer: A rotary potentiometer is a versatile and commonly used electronic component that functions as a variable resistor. It allows for the adjustment of resistance through the rotation of its shaft, which in turn alters the output voltage in a circuit. Potentiometers are widely used in various applications, such as adjusting the volume on audio equipment, dimming lights, and controlling the position of servos in robotics.

- Breadboard: A breadboard is a reusable platform used for building and testing electronic circuits without soldering. It has a grid of holes with internal metal strips to connect components easily. Breadboards facilitate quick prototyping and modifications, making them ideal for experimenting with circuit designs. They come in various sizes and are essential tools in electronics education and development.



- Jumper wires: Jumper wires are short electrical wires with connector pins at each end, used to establish connections between components on a breadboard or other circuit prototypes without soldering. They come in various lengths and types, such as male-to-male, male-to-female, and female-to-female, to suit different connection needs. Jumper wires facilitate quick and flexible modifications in circuit designs, making them essential tools for electronics experimentation and prototyping.

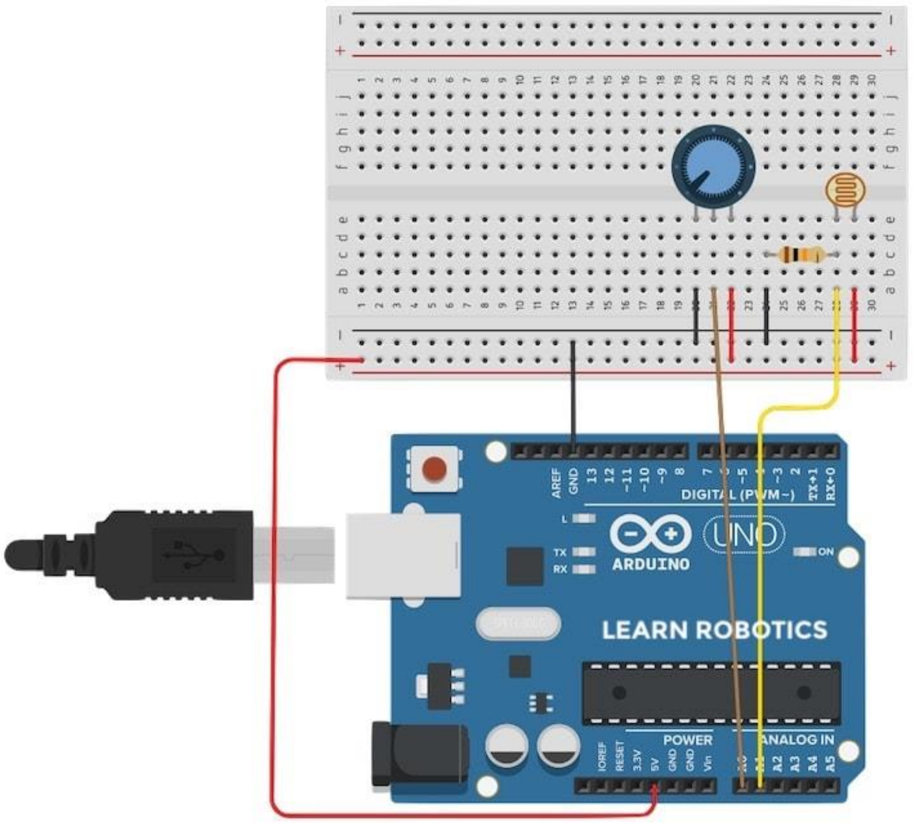
- Photoresistor: A photoresistor, also known as a light-dependent resistor (LDR), is a light-sensitive electronic component that exhibits a change in electrical resistance based on the intensity of light it is exposed to. Photoresistors are widely used in various applications such as light sensors, automatic lighting controls, and light intensity meters.

- USB Cable: USB (Universal Serial Bus) cable is an essential accessory for connecting an Arduino board to a computer. This connection serves multiple purposes, including power supply, programming the Arduino, and enabling serial communication for data exchange between the Arduino and the computer.

Resistor: A 10k ohm resistor is a common and widely used electronic component that provides a resistance of 10,000 ohms. Resistors are fundamental components in electrical and electronic circuits, used to control the flow of current and manage voltage levels within the circuit

Design and Implementation

Fritzing Diagram for the circuit:



The design and implementation of the given project follows the given connections:

1. **Wiring the Arduino and Sensors:**

Components Needed:

- Arduino Uno

- Breadboard
- Rotary potentiometer
- Photoresistor (light sensor)
- Resistors (e.g., 10kΩ for the photoresistor)
- Connecting wires
- USB cable

Wiring Instructions:

1. Connect the Rotary Potentiometer to the Arduino:

- Place the potentiometer on the breadboard.
- Connect the left pin (VCC) to the Arduino’s 5V pin.
- Connect the right pin (GND) to the Arduino’s GND.
- Connect the middle pin (analog output) to one of the Arduino’s analog input pins (e.g., A0).

2. Connect the Photoresistor to the Arduino:

- Place the photoresistor on the breadboard.
- Connect one end of the photoresistor to the Arduino’s 5V pin.
- Connect the other end to an analog input pin (e.g., A1) and also to GND through a resistor (10kΩ).

3. Verify Connections:

Ensure all connections are secure and correctly placed to avoid short circuits and incorrect readings.

2. **Programming the Arduino:**

Arduino Code

```
// Define sensor pins
const int potPin = A0; // Rotary potentiometer connected to A0

const int lightPin = A1; // Photoresistor connected to A1

void setup() {
// Initialize serial communication
Serial.begin(9600);
}
void loop() {
// Read sensor value
int potValue = analogRead(potPin);
int lightValue = analogRead(lightPin);

// Print sensor values to the Serial Monitor
Serial.print("Potentiometer: ");
Serial.print(potValue);
Serial.print(", Light: ");
Serial.println(lightValue);

// Delay for stability
delay(1000); // 1 second delay

}
```

Upload the code to your Arduino Uno using the Arduino IDE. Open the Serial Monitor to verify that the sensor readings are displayed correctly.

3. **<u>Collecting Data Using Python:</u>**

Python Script for Data Logging:

1. Install Required Libraries:

   Ensure you have the `pyserial` library installed. You can install it using pip:

   ```
   pip install pyserial
   ```

   2. Python Code:

```
import serial
import csv
import time

# Configure the serial connection to the Arduino
ser = serial.Serial('COM3', 9600)  # Replace 'COM3' with your Arduino port

# Create a CSV file to store the data

with open('sensor_data.csv', 'w', newline='') as csvfile:

    fieldnames = ['Timestamp', 'Potentiometer', 'Light']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
    try:
        while True:
            # Read data from the Arduino
            line = ser.readline().decode('utf-8').strip()
            print(line)

            # Parse the data (assuming comma-separated values)
            data = line.split(',')
```

```
                if len(data) == 2:
                 # Adjust based on the number of sensors

                    timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
                    pot = data[0].split(':')[1].strip()
                    light = data[1].split(':')[1].strip()


                # Write the data to the CSV file
                    writer.writerow({'Timestamp': timestamp, 'Potentiometer':
                    pot, '   Light': light})

        except KeyboardInterrupt:

            print("Data logging stopped.")
            ser.close()
```

This script reads data from the Arduino via the serial port, parses the sensor values, and writes them into a CSV file with a timestamp.

Importance and Applications of Data Logging Systems

Data logging systems play a crucial role in a multitude of fields, offering invaluable insights and aiding in informed decision-making. These systems, which collect and record data from various sensors, help in monitoring, analyzing, and optimizing processes across different industries. Let's delve into the importance and specific applications of data logging systems in more detail.

Importance of Data Logging Systems:

1. Accurate and Reliable Data Collection: Data loggers provide precise and consistent data over time, ensuring that the information collected is reliable. This accuracy is essential for scientific research, quality control, and any application where data integrity is critical.

2. Real-Time Monitoring: Continuous monitoring of conditions or parameters in real-time allows for immediate responses to any deviations or issues. This capability is particularly important in applications such as environmental monitoring and industrial automation.

3. Long-Term Data Storage: Data logging systems can store data over extended periods, enabling the analysis of trends and long-term patterns. This historical data is valuable for predictive maintenance, climate studies, and any application requiring trend analysis.

4. Automation and Efficiency: By automating the data collection process, data loggers reduce the need for manual monitoring, freeing up human resources for other tasks. This automation increases efficiency and reduces the potential for human error.

5. Enhanced Decision-Making: With accurate, real-time data at hand, decision-makers can make more informed choices. This data-driven approach leads to better outcomes in terms of operational efficiency, safety, and cost-effectiveness.

Applications of Data Logging Systems:

1. Environmental Monitoring

- Agriculture: Farmers use data loggers to monitor soil moisture, temperature, and light intensity, helping them optimize irrigation, planting, and harvesting schedules. This leads to increased crop yields and resource efficiency.
- Climate Research: Scientists use data loggers to collect long-term environmental data, such as temperature, humidity, and light levels. This information is crucial for studying climate change, understanding ecosystems, and developing conservation strategies.

2. Industrial Automation

- Predictive Maintenance: Data loggers monitor machinery and equipment to detect signs of wear and tear. By analyzing trends in the data, maintenance can be scheduled proactively, reducing downtime and preventing costly failures.
- Process Optimization: In manufacturing, data loggers track parameters like temperature, pressure, and flow rates. This data helps in fine-tuning processes for better efficiency, product quality, and safety.

3. Smart Homes

- Energy Management: Sensors in smart homes track light intensity, occupancy, and other parameters to automate lighting and climate control systems. This automation improves energy efficiency, reduces utility costs, and enhances comfort.
- Security: Data loggers integrated with motion sensors and cameras provide continuous monitoring of a home's security status. This real-time data can trigger alarms and notifications, enhancing home security.

4. Healthcare

- Rehabilitation Devices: Potentiometers and other sensors in rehabilitation equipment track patient movements and progress. Data logging helps therapists and doctors to monitor and adjust treatment plans based on accurate data.

- Wearable Health Devices: Wearable sensors that monitor heart rate, activity levels, and other health metrics use data logging to provide continuous health data. This information aids in early detection of health issues and personalized healthcare management.

5. Transportation and Logistics

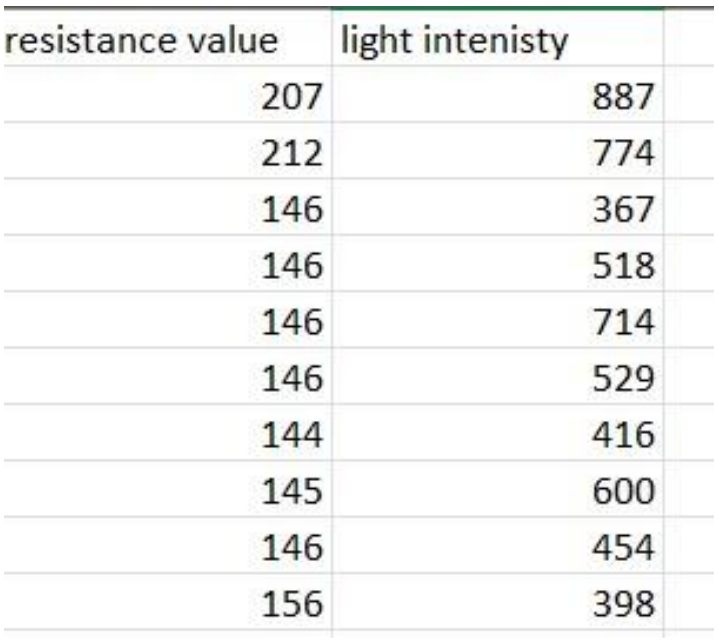
- Fleet Management: Data loggers in vehicles monitor parameters such as speed, location, fuel consumption, and engine health. This data helps in optimizing routes, improving fuel efficiency, and maintaining vehicles.
- Cold Chain Monitoring: In logistics, especially for perishable goods, data loggers track temperature and humidity levels to ensure that products remain within safe conditions during transit.

6. Research and Development

- Scientific Experiments: Data loggers are used in laboratories to monitor experimental conditions and collect data over time. This capability is essential for reproducibility and accuracy in scientific research.
- Product Testing: In product development, data loggers test the durability and performance of products under various conditions. This testing ensures that products meet quality and safety standards.

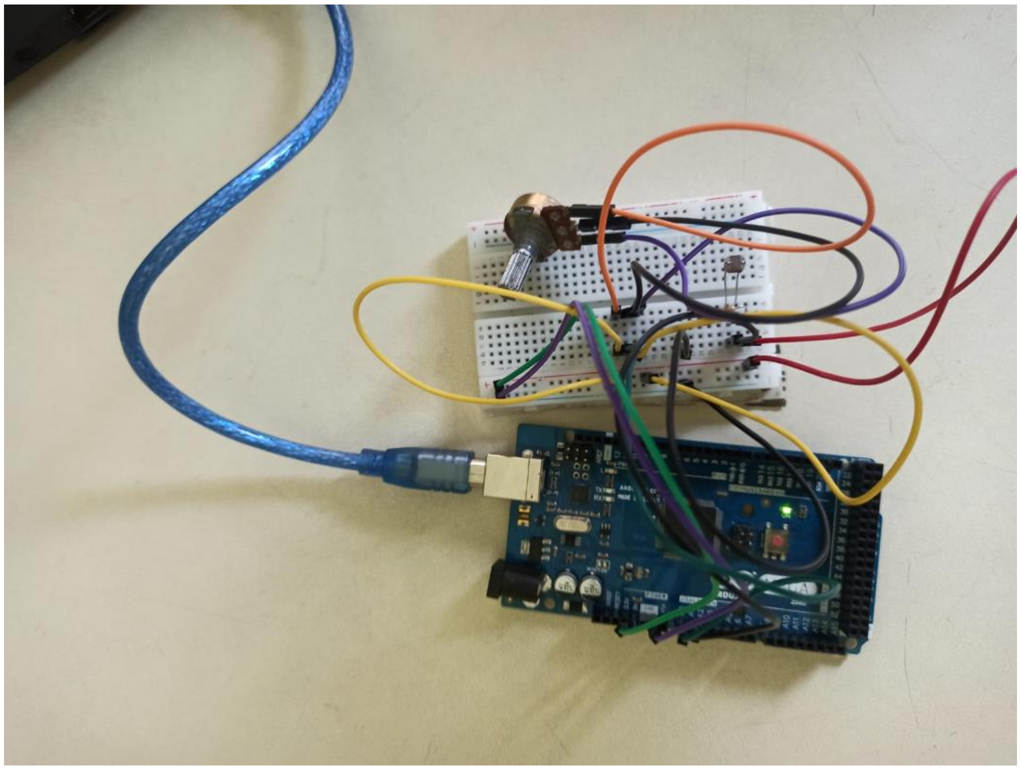
Output

CSV File



resistance value	light intenisty
207	887
212	774
146	367
146	518
146	714
146	529
144	416
145	600
146	454
156	398

Circuit Connection



Conclusion

This project illustrates the integration of hardware and software to create a functional data logging system. By following the steps outlined, you have built a system capable of capturing, monitoring, and storing sensor data for analysis. The skills and knowledge gained from this project are foundational for developing more complex IoT applications. You can expand this project by adding more sensors, incorporating wireless communication modules, or integrating cloud storage solutions, making it highly versatile for various IoT applications.

The successful completion of this project not only demonstrates the practical application of IoT technology but also provides a solid foundation for further exploration and development in the field of data analytics and automation. This system can be a stepping stone for larger, more complex projects, enabling you to leverage the power of IoT to solve real-world problems and enhance operational efficiencies across different domains.

Further Improvements in the Data Logging with Arduino Project

After successfully setting up a basic data logging system using Arduino and Python, several enhancements can be made to improve the system's functionality, efficiency, and versatility. Here are some potential areas for further improvement:

1. Wireless Data Transmission

Implementing wireless communication methods can enhance the flexibility and scalability of the data logging system. This can be achieved using modules such as:

- Wi-Fi (ESP8266 or ESP32): Enable the Arduino to send data to a remote server or cloud platform, allowing for real-time data access and monitoring from anywhere.
- Bluetooth (HC-05 or HM-10): Facilitate wireless data transmission to nearby devices such as smartphones or tablets for localized monitoring and control.
- LoRa (Long Range Radio): Suitable for long-distance data transmission in remote or wide-area applications, such as agricultural fields or industrial sites.

2. Advanced Data Storage Solutions

Upgrading the data storage mechanism can improve data management and accessibility:

- SD Card Module: Integrate an SD card module to locally store large amounts of data, which can be retrieved later for analysis.
- Cloud Integration: Use IoT platforms like ThingSpeak, AWS IoT, or Google Cloud IoT Core to store and visualize data in the cloud, providing remote access and advanced analytics capabilities.

3. Enhanced Data Visualization

Improving how data is presented can make it more useful and easier to interpret:

- Real-Time Dashboards: Create real-time dashboards using web-based platforms such as Grafana or Plotly to visualize data trends and patterns dynamically.
- LCD or OLED Displays: Add a local display to the Arduino setup to show sensor readings directly, making it easier to monitor data without a computer.

4. Power Management and Portability

Making the data logger portable and energy-efficient can broaden its application scope:

- Battery Power: Implement battery power solutions, such as rechargeable Li-ion batteries, to make the system portable.
- Solar Power: Use solar panels for sustainable and continuous power supply, particularly useful for outdoor applications.
- Sleep Modes: Optimize the Arduino code to include sleep modes, reducing power consumption when the system is idle.

5. Sensor Expansion and Diversity

Expanding the variety and number of sensors can provide a more comprehensive data set:

- Additional Sensors: Integrate more sensors, such as humidity, pressure, CO2, or GPS modules, to gather a wider range of environmental data.
- Analog and Digital Sensors: Use a mix of analog and digital sensors to enhance the quality and resolution of the data collected.

6. Data Security

Ensuring the integrity and security of the data collected:

- Encryption: Encrypt data before transmission, especially when using wireless communication methods, to protect against unauthorized access.
- Authentication: Use authentication mechanisms to secure access to the data logging system and the stored data.

7. User Interface and Ease of Use

Improving the user experience can make the system more accessible:

- User-Friendly Software: Develop a graphical user interface (GUI) for the Python script to simplify data logging operations, allowing users to start/stop logging, change settings, and view logs easily.
- Configuration Settings: Allow users to configure settings such as sampling rate, sensor thresholds, and communication parameters through a simple interface.

8. Machine Learning and Advanced Analytics

Leveraging advanced data analysis techniques can extract more insights from the data:

- Machine Learning Models: Integrate machine learning models to predict trends, detect anomalies, or classify data based on the collected sensor readings.
- Data Preprocessing: Implement data preprocessing techniques, such as filtering and normalization, to improve the quality of data before analysis.

By incorporating these improvements, the data logging system based on Arduino can be significantly enhanced, making it more versatile, reliable, and user-friendly. These advancements will not only broaden the range of applications but also improve the overall efficiency and effectiveness of data collection and analysis. Each of these improvements can be tailored to specific project requirements, providing a robust foundation for various IoT applications.

References

- https://forum.arduino.cc/t/creating-csv-file/
- https://www.learnrobotics.org/blog/arduino-data-logger-csv/#step0
- Wikipedia
- https://www.researchgate.net/post/How-can-I-save-data-directly-in-the-pc-using-an-Arduino-UNO