

**INTERFACING MEMORY/SD CARD MODULE WITH ARDUINO UNO
TO STORE THE DATA
A REPORT SUBMITTED**

FOR PROJECT BASED LEARNING

**ADESH TEKALE (SE23266)
SHRUTI VARALE (SE23268)
TANVI WADGAONKAR (SE23271)**

**UNDER THE GUIDANCE OF
DR. RISIL CHHATRALA**

S.E. (ELECTRONICS AND TELECOMMUNICATION)



DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION
HOPE FOUNDATION'S
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY (I²IT)
SAVITRIBAI PHULE PUNE UNIVERSITY
HINJAWADI, PUNE (MH) - 411057
AY-2023-24 SEMESTER II

CERTIFICATE

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

HOPE FOUNDATION'S

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY (I²IT)

HINJAWADI, PUNE- 411057



This is to certify that Adesh Tekale (SE23266) Shruti Varale (SE23268) Tanvi Wadgaonkar (SE23271) Class: SE (E&TC) have successfully completed project titled, **“INTERFACING MEMORY/SD CARD MODULE WITH ARDUINO UNO TO STORE THE DATA”** as a part of semester II of Second Year of Bachelor of Engineering in Electronics and Telecommunication (A.Y. 2023-2024) of Savitribai Phule Pune University.

Dr. Rishil Chhatrala
Project Guide

Dr. S. M. M Naidu
HOD (E&TC)

Dr. Vaishali V Patil
Principal

Place: Pune

Date:

ABSTRACT

The integration of SD (Secure Digital) cards with Arduino microcontrollers has become a popular solution for projects requiring data logging, storage, and retrieval. This abstract outlines the key components and steps involved in interfacing an SD card with an Arduino, providing a foundational understanding for enthusiasts and engineers alike.

The abstract begins by elucidating the necessity and versatility of SD card usage in Arduino projects, highlighting its capacity for storing large volumes of data in various formats. It then introduces the primary hardware components required for the interface, including the Arduino board itself, an SD card module, and requisite connections such as SPI (Serial Peripheral Interface) pins.

Furthermore, the abstract discusses the software aspect of the interface, focusing on the libraries and functions essential for SD card communication within the Arduino IDE (Integrated Development Environment). Emphasis is placed on the SD.h library, which facilitates effortless access to SD card functionalities through simple commands.

The abstract proceeds to delineate a step-by-step methodology for interfacing an SD card with Arduino, encompassing hardware setup, library inclusion, initialization of SD card communication, and execution of basic read/write operations. Additionally, it addresses potential challenges and troubleshooting strategies, ensuring a smooth integration process for practitioners.

CONTENTS

List of Figures	5
List of Tables	27
List of Abbreviations	6
1. INTRODUCTION	7
1.1 Background	
1.2 Mobile Ad-Hoc Network (MANET)	
2. LITERATURE SURVEY	8
3. PROPOSED METHODOLOGY	9
3.1 Problem Statement	
3.2 Problem Motivation	
3.3 Process description	
4. PROJECT IMPLEMENTATION	17
4.1 Circuit Designing	
4.2 Simulation and Bread Board Testing	
4.3 Programming	
4.4 Project component and cost estimate	
5. RESULTS AND DISCUSSION	25
6. CONCLUSION AND FUTURE SCOPE	27
REFERENCES	28

LIST OF FIGURES

Figure No.	Figure Name
1	Arduino Uno Board
2	SD Card Module
3	DHT11 Sensor
4	SD Card (8GB) and Adaptor
5.	SD Card Reader
6	Jumping Wires (Male to Female)

LIST OF ABBREVIATIONS

MANET	Mobile Ad-Hoc Network
SPI	Serial Peripheral Interface
SD Card	Secure Data
IOT	Internet of Things
Arduino IDE	Arduino Integrated Development Environment
MISO	Master-in Slave-out
MOSI	Master-out Slave-in
SCK	Serial Clock

CHAPTER 1

INTRODUCTION

1.1 Background

In the early days of Arduino, there were efforts by various individuals to create libraries and tutorials for interfacing SD cards with Arduino. These efforts laid the groundwork for future developments. The official Arduino SD library was integrated into the Arduino software, making it easier for users to access SD card functionality without needing to rely on third-party libraries. The Arduino SD library has seen continuous updates and improvements over the years, adding features, optimizing performance, and ensuring compatibility with new Arduino boards. As Arduino boards evolved, so did their capabilities for interfacing with SD cards. Features like SPI communication and hardware support for SD card interfaces became standard, enhancing speed and reliability.

1.2 Mobile Ad-Hoc Network (MANET)

A mobile ad hoc network is a group of wireless mobile computers (or nodes); in which nodes collaborate by forwarding packets for each other to allow them to communicate outside range of direct wireless transmission. A MANET is an autonomous group of mobile users that communicate over reasonably slow wireless links. A Mobile Ad-Hoc Network (MANET) is a decentralized type of wireless network where nodes communicate with each other without relying on a fixed infrastructure like routers or access points. Instead, devices in a MANET form a dynamic network by directly connecting to each other.

As for interfacing an SD card with Arduino, it involves connecting an SD card module to the Arduino board and using appropriate libraries to read from and write to the SD card. The module typically communicates with the Arduino via SPI (Serial Peripheral Interface) or SDIO (Secure Digital Input Output) interface, allowing the Arduino to access the data stored on the SD card. This enables tasks like data logging, file storage, or accessing configuration files for your Arduino projects.

CHAPTER 2

LITERATURE SURVEY

Begin by familiarizing yourself with the SD card module. This includes understanding its pin configuration, communication protocol (usually SPI), and how to interface it with the Arduino Uno. Look for existing Arduino libraries that support SD card modules. Libraries like "SD.h" provide functions to easily read from and write to SD cards. Check the Arduino IDE Library Manager or GitHub for popular libraries. Explore tutorials or documentation on how to physically connect the SD card module to the Arduino Uno. This involves connecting the SPI pins (MISO, MOSI, SCK), a chip select pin (usually labeled CS), and optionally a power and ground connection. Understand how to initialize the SD card using the appropriate library. This typically involves checking if the card is present, initializing the SPI communication, and mounting the filesystem. Learn how to read from and write to the SD card using the library functions. This may involve opening and closing files, writing data to files, and reading data from files. Make sure to handle errors gracefully. Experiment with simple programs to test the functionality of the SD card module with the Arduino Uno. Use debugging techniques such as serial print statements to troubleshoot any issues that arise. Explore optimization techniques and best practices for efficient data storage and retrieval on the SD card. This includes considerations such as file organization, data formatting, and memory management. If you're planning to store sensor data, integrate the SD card module with your sensor setup. This may involve reading sensor data using other Arduino libraries and storing it on the SD card.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 Problem Statement:

Interfacing SD Card module with Arduino Uno to store the Data. The main goal is to create a data logging system using an Arduino Uno, an SD or memory card module. The system should be able to log data from sensors or other sources onto the SD card, and simultaneously display some relevant information on the Serial monitor for real-time monitoring.

Objective:

- ❖ Reading and writing the data to the memory card, managing the file system, and potentially implementing communication protocol like SPI between Arduino uno and the memory card.
- ❖ Sensing several gas level in the air , temperature, humidity and also quality of the air.
- ❖ By Interfacing the memory card module with Arduino uno to store the data.

Outcomes:

3.2 Problem Motivation:

Many projects and applications require the collection and storage of data over time. This could be sensor readings (temperature, humidity, pressure, etc.), GPS coordinates, or any other type of data relevant to the project. Storing this data on an external medium like an SD card allows for long-term data collection without relying on the limited memory of the Arduino itself..Storing data on an SD card provides redundancy and security. If the Arduino loses power or encounters a malfunction, the data stored on the SD card remains intact. This ensures that valuable data is not lost due to unforeseen circumstances.

While storing data on the SD card, the Arduino Uno can also display relevant information in real-time on Serial Monitor. This allows users to monitor the system's operation and view live data without needing to access the stored data on the SD card directly. Real-time monitoring is crucial for many applications where immediate feedback is needed. Developing a data logging system with an Arduino Uno and an SD card module serves as an excellent educational project for learning about microcontroller programming, data storage, and hardware interfacing. It's also a popular choice for hobbyists and DIY enthusiasts who enjoy building custom electronics projects.

Overall, the motivation for interfacing an SD or memory card module with an Arduino Uno to store data lies in its practical utility across various fields, its portability and flexibility, and its educational and hobbyist appeal. It enables the creation of versatile data logging systems that can be adapted to a wide range of applications and environments.

3.3 Process description

Interfacing an SD memory card with an Arduino Uno enables the Arduino to store and retrieve data, expanding its capabilities for data logging, storage, and retrieval. This project involves establishing communication between the Arduino Uno and the SD card, implementing file system management, and ensuring data integrity. The outcome is a versatile system capable of effectively utilizing SD memory cards for various embedded applications, enhancing the functionality of the Arduino Uno. using the SPI communication protocol. SD card use to expand the storage capacity of our smartphone or tablet, allowing us to store data. From the Air pollution Detection project, we only used to sense several gas levels in the air and also the ambient temperature and humidity, thus sensing the quality of the air and also show on Serial Monitor.

Air Pollution detection system is designed to monitor and measure the concentration of pollutants in the air. The system typically includes sensors, data processing units, and communication components. In Data Analysis and Interpretation the processed data is analyzed to determine the overall air quality and identify any specific pollutants that exceed predefined thresholds. The system may categorize air quality levels into different classes such as "good," "moderate," "unhealthy," etc. The Arduino can create a file in an SD card to write and save data using the SD library. There are different models from different suppliers, but they all work in a similar way, using the SPI communication protocol. SD card use to expand the storage capacity of our smartphone or tablet, allowing us to store data. Sensors continuously collect real-time data on the concentrations of different pollutants. The data may include information about the type and amount of pollutants present in the air. From the Air pollution Detection project, we only used to sense several gas levels in the air and also the ambient temperature and humidity, thus sensing the quality of the air and also Show on the serial Monitor .

Methodology

DHT11 to Arduino Uno:

Connect the OUT pin of the DHT11 sensor to digital pin 2 on Arduino Uno.

SD Card Module to Arduino Uno:

Connect the MISO pin of the SD module to Pin 12 on Arduino Uno.

Connect the MOSI pin of the SD module to Pin 11 on Arduino Uno.

Connect the SCK (Serial Clock) pin of the SD module to Pin 13 on Arduino Uno.

Connect the CS (Chip Select) pin of the SD module to any digital pin 10 to Arduino Uno.

Power Connections:

Connect VCC of the DHT11 and SD module to 5V on Arduino Uno.

Connect GND of the DHT11 and SD module to GND on Arduino Uno.

Connecting SD Card:

Insert the micro SD card into the SD card module.

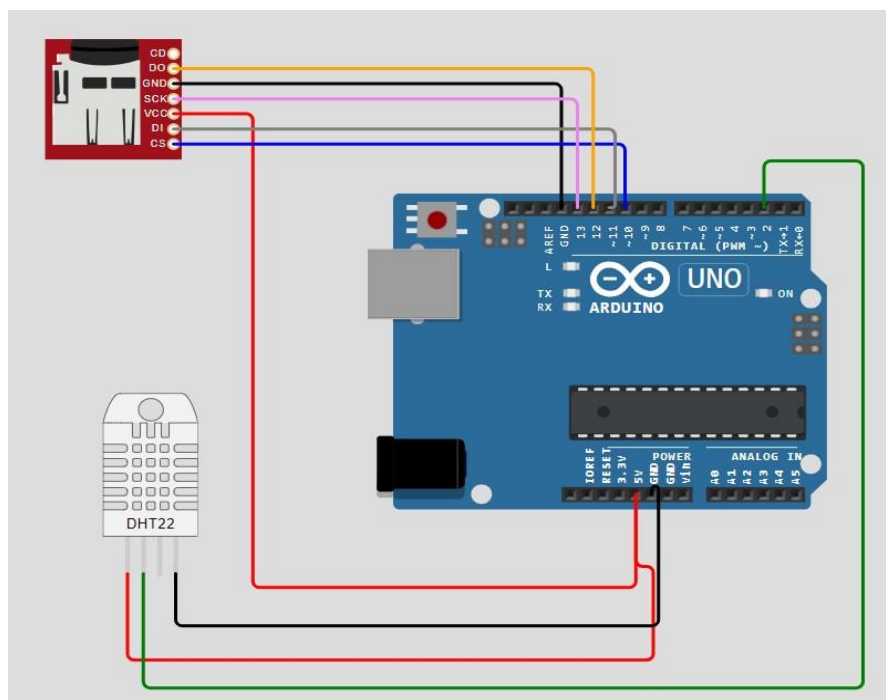


Figure 1. Simulation Diagram

Component description

Arduino uno board

SD Card module

DHT11 Sensor

SD card (8GB) & Adaptor

SD Card Reader

Jumping wire

Arduino Uno Board: The Arduino Uno is a popular microcontroller board featuring an ATmega328P microcontroller. It includes digital and analog I/O pins for connecting sensors and actuators, USB connectivity for programming and power, and a power jack for external power supply. It's widely used for prototyping and DIY electronics projects due to its simplicity and versatility.



Figure 2. Arduino Uno Board

SD Card Module: An SD card module is a small circuit board that allows microcontrollers like Arduino to communicate with SD or microSD memory cards. It typically includes a card slot, voltage regulation, and level shifting circuitry for compatibility. Communication is usually via SPI, with pins for power, chip select, and data transfer. It's a convenient way to add data storage to projects.



Figure 3. SD Card Module

DHT11 Sensor: The DHT11 sensor is a low-cost digital temperature and humidity sensor module. It consists of a capacitive humidity sensor and a thermistor for temperature measurement. It provides digital output via a single-wire interface, making it easy to interface with microcontrollers like Arduino. The sensor is compact, reliable, and suitable for basic temperature and humidity monitoring applications.



Figure 4. DHT11 Sensor

SD card (8GB) & Adaptor: An 8GB SD card is a small, portable storage device commonly used in electronic devices like cameras, smartphones, and microcontroller-based systems. It uses flash memory technology to store data and features a standard SD card form factor. With a capacity of 8 gigabytes, it can hold a significant amount of data, including documents, photos, videos, and code. It's widely compatible with devices supporting SD cards and is suitable for various applications requiring data storage and transfer.

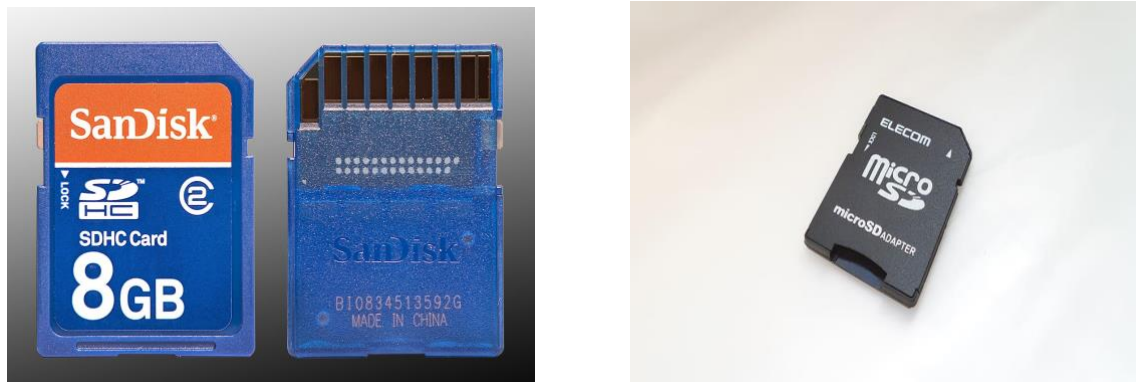


Figure 5. SD Card (8GB) and Adaptor

SD Card Reader: An SD card reader is a hardware device used to read data from and write data to SD cards. It typically connects to a computer or other device via USB or another interface. The reader features a slot or socket where the SD card can be inserted, allowing it to interface with the device's data bus. SD card readers are commonly used to transfer files between SD cards and computers, as well as to access data stored on SD cards in digital cameras, smartphones, and other

devices. They are available in various form factors and are widely compatible with different types and capacities of SD cards.



Figure .6 SD Card Reader

Jumping wire: A jumping wire, also known as a jumper wire, is a short electrical wire with a male connector on one end and a female connector on the other end. It is used to create electrical connections between components on a breadboard, circuit board, or other electronic devices. The male connector typically has pins or prongs that can be inserted into sockets or holes, while the female connector has receptacles or sockets that can accept pins or prongs. Jumping wires allow for quick and temporary connections in electronic prototyping and testing, facilitating circuit design and troubleshooting.

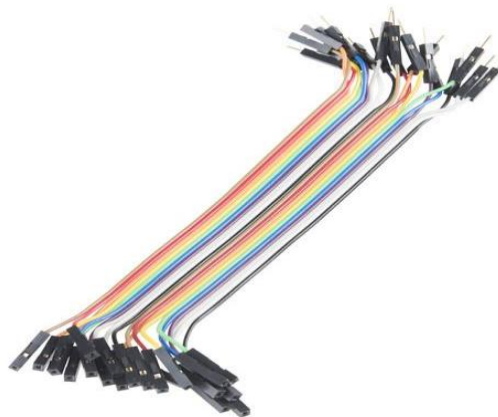


Figure 7. Jumping Wires (Male to Female)

3.4 Requirement Analysis

Hardware Requirements:

- **Arduino Uno:** The project requires an Arduino Uno microcontroller board as the main controller.
- **SD Card Module:** A compatible SD card module to interface with the SD card and the Arduino Uno.
- **Sensors or Data Sources:** DHT11 Sensor whose data needs to be logged onto the SD card.
- **Power Supply:** A stable power supply to ensure uninterrupted operation.
- **SD Card (8 gb)& Adapter:** It uses flash memory technology to store data and features a standard SD card form factor.

Software Requirements:

- **Arduino IDE:** The Arduino Integrated Development Environment (IDE) for writing, compiling, and uploading code to the Arduino Uno.
- **SD Card Library:** Utilization of Arduino libraries like "SD.h" for interfacing with the SD card module.
- **Sensor Libraries:** If sensors are used, relevant libraries for interfacing with and reading data from the sensors.
- **Wokwi integration:** Integrate Wokwi software to simulate the Arduino Uno and SD card module interaction virtually, allowing for code testing and debugging in a simulated environment.

3.5 Impact Analysis

Implementing SD card interfacing expands the data storage capabilities of the Arduino Uno, enabling it to store large amounts of data beyond its onboard memory. Introducing external hardware components like the SD card module increases the complexity of the system, requiring careful hardware and software integration. The ability to store data on an external medium provides redundancy and backup, reducing the risk of data loss due to system failures or power outages. Adding an SD card module and other necessary components may increase the overall hardware costs of the project, impacting budget considerations. Users can retrieve stored data from

the SD card for analysis or further processing, expanding the utility of the system beyond real-time monitoring.

Minimizing energy usage can reduce the environmental impact, especially if the project runs on batteries or relies on energy from non-renewable sources. Assess the materials used in manufacturing the Arduino Uno, the SD card, and associated components. Some materials may have higher environmental costs in terms of extraction, processing, and disposal. Choosing components made from recycled or eco-friendly materials can mitigate environmental impact. Explore ways to minimize emissions associated with the project, such as optimizing code efficiency to reduce energy consumption during data storage and retrieval operations. Additionally, consider using renewable energy sources to power the project where feasible.

3.6 Professional Ethics Practices to be followed

Data Privacy and Security: Ensure that sensitive data stored on the SD card is protected from unauthorized access or disclosure. Implement encryption techniques if necessary to secure data transmission and storage.

Accuracy and Reliability: Verify the accuracy and reliability of the data stored on the SD card by implementing error-checking mechanisms and data validation techniques. Document any assumptions or limitations regarding the accuracy of the stored data. Be transparent about the potential risks or uncertainties associated with the data stored on the SD card.

Backup and Redundancy: Maintain regular backups of the data stored on the SD card to prevent data loss in case of hardware failure or other unforeseen circumstances. Redundancy measures can also be implemented to ensure data availability.

Testing and Validation: Thoroughly test the SD card module interface with the Arduino Uno to ensure its reliability and performance under various conditions. Validate that the data is being stored accurately and securely.

CHAPTER 4

PROJECT IMPLEMENTATION

4.1 Circuit Designing

We use the Wokwi website to simulate our circuit .Go to the Wokwi website (wokwi.com) and sign in or create an account if you haven't already. Once logged in, navigate to the Circuit Simulator. On the left-hand side, you'll see a toolbar with various components. Look for the components needed for interfacing the SD card with Arduino Uno: Arduino Uno, SD Card Module, Wires for connections. Drag and drop the Arduino Uno and SD Card Module onto the workspace. Arrange them as needed for your circuit layout. Use the wire tool to connect the appropriate pins of the SD card module to the Arduino Uno. Refer to the pinout of your specific SD card module and Arduino Uno to ensure correct connections. Typically, you'll connect MOSI, MISO, SCK, and CS pins from the SD card module to digital pins 11, 12, 13, and any available digital pin for CS on the Arduino Uno, respectively. Connect the VCC and GND pins of the SD card module to 5V and GND on the Arduino Uno, respectively. Double-check all connections to ensure they are correctOnce your circuit and code are ready, click on the "Simulate" button to run the simulation. You can interact with the simulation by clicking buttons, changing input values, or observing output values in real-time.

Block Diagram

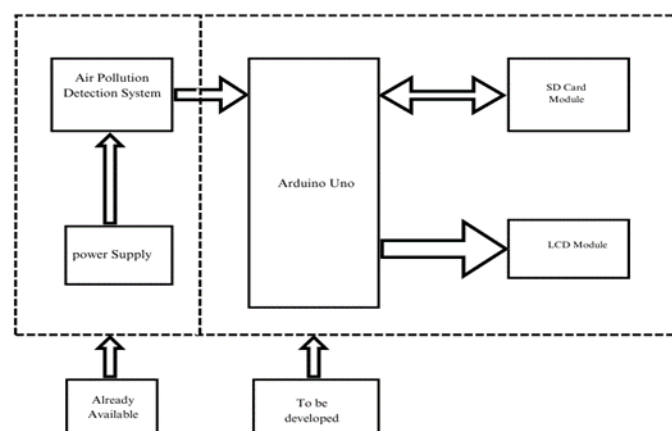


Figure 8. Block Diagram

Simulation Diagram

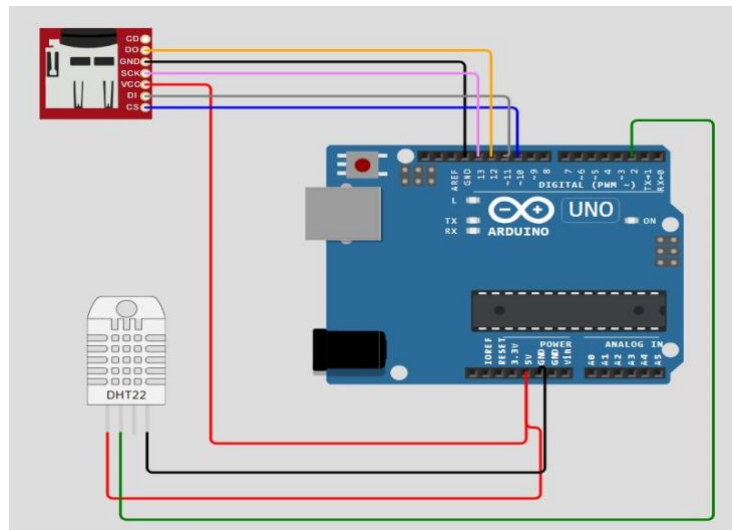


Figure 9. Simulation Circuit

Actual Circuit Diagram

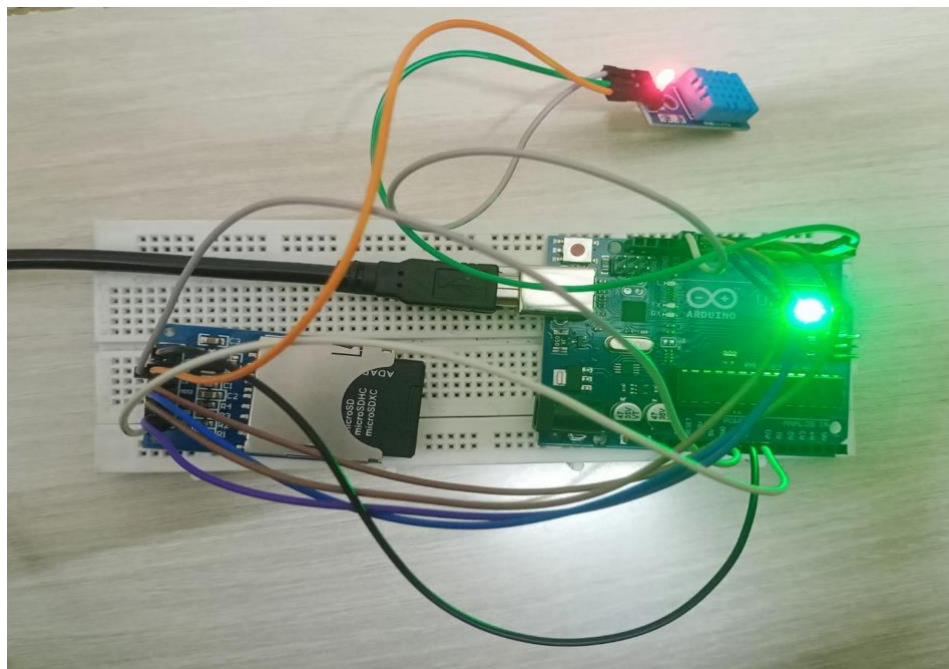


Figure 10. Actual Circuit

4.2 Simulation and Bread Board Testing

Simulation with Wokwi Software:

Component Testing: Use Wokwi to simulate the behavior of individual components such as the Arduino Uno, SD card module, and any other peripherals you plan to use in your project. This allows you to verify that each component functions correctly and troubleshoot any issues before integrating them into your project.

Code Testing: Write and test your Arduino code in the Wokwi simulator. You can simulate interactions with the SD card, such as writing data to the card and reading data from it. This enables you to debug your code and ensure that it performs as expected before deploying it to physical hardware.

Real-time Feedback: Wokwi provides real-time feedback on the behavior of your circuit and code, allowing you to observe how changes impact performance instantly. This iterative process helps you refine your design and identify any potential issues early in the development cycle.

Breadboard Testing:

Physical Validation: Once you're satisfied with the simulation results in Wokwi, proceed to implement your project on a physical breadboard. This involves connecting the Arduino Uno, SD card module, and other components as per your design.

Component Integration: Test the integration of all components on the breadboard to ensure that they function together as intended. Verify that the connections are secure and that there are no loose wires or solder joints that could cause issues.

Functional Testing: Execute the same tests on the physical setup that you performed in the simulation. Verify that the Arduino code interacts correctly with the SD card module and that data can be successfully stored and retrieved.

4.4 Programming

```
#include <SD.h>

#include <SPI.h>

#include <DHT.h>

#define DHTPIN 2      // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11

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

const int CS_PIN = 10; // Chip select pin for SD card module

File dataFile;

void setup() {
    Serial.begin(9600);

    while (!Serial); // Wait for serial monitor to open

    Serial.println("Initializing...");

    // Initialize DHT sensor
    dht.begin();

    // Initialize SD card
    if (!SD.begin(CS_PIN)) {
        Serial.println("SD card initialization failed!");
        return;
    }

    Serial.println("SD card initialized.");

    // Create or open a file
    dataFile = SD.open("CardData.txt", FILE_WRITE);

    if (!dataFile) {
        Serial.println("Error opening file!");
        return;
    }

    Serial.println("File opened.");
}
```

```

void loop() {
    delay(2000);

    // Read humidity and temperature from DHT sensor
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature()
    // Check if readings are valid
    if (isnan(humidity) || isnan(temperature)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }
    // Print data to serial monitor
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" °C");
    // Write data to SD card
    dataFile.print("Humidity: ");
    dataFile.print(humidity);
    dataFile.print(" %, Temperature: ");
    dataFile.print(temperature);
    dataFile.println(" °C");

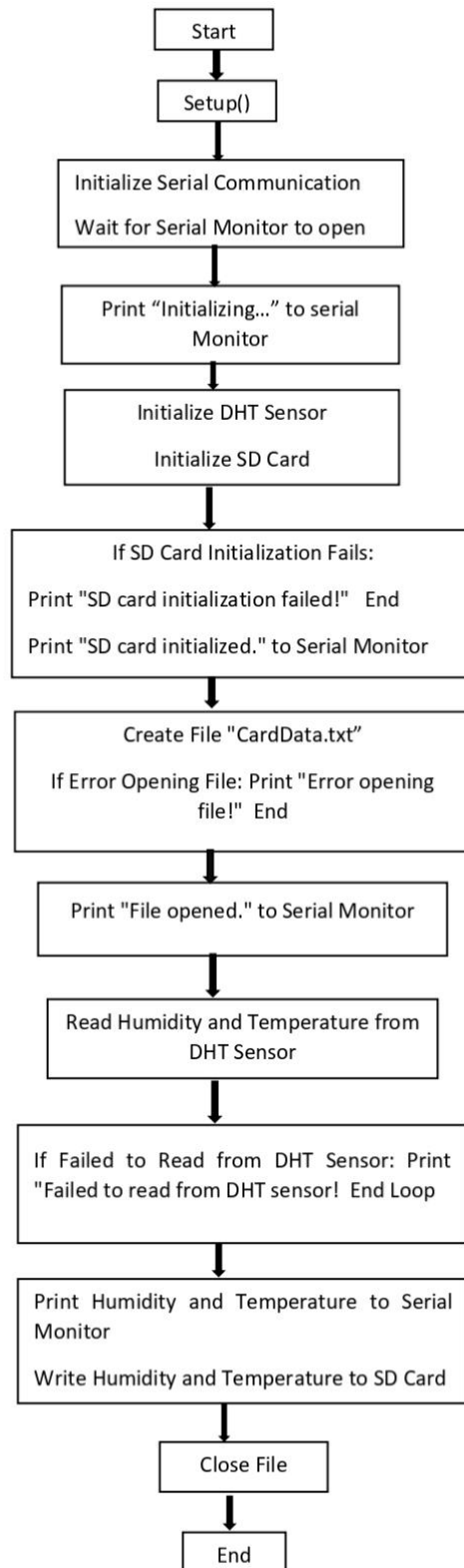
    // Close the file
    dataFile.close();
}

```

Algorithm

- Import the necessary libraries for SD card (SD.h), SPI communication (SPI.h), and DHT sensor (DHT.h).
- Define the digital pin (DHTPIN) connected to the DHT11 sensor and specify the sensor type (DHTTYPE) as DHT11.
- Create an instance of the DHT class (dht) with the specified pin and sensor type (DHT dht(DHTPIN, DHTTYPE);).
- Begin serial communication at a baud rate of 9600 (Serial.begin(9600);).
- Wait until the serial monitor is opened (while (!Serial);).
- Display an initialization message on the Serial Monitor (Serial.println("Initializing...");).
- Initialize the DHT sensor (dht.begin();).
- Initialize the SD card using the specified chip select pin (CS_PIN) and check if initialization was successful (if (!SD.begin(CS_PIN))).
- Print initialization status messages on the Serial Monitor based on SD card initialization.
- Attempt to open or create a file named CardData.txt on the SD card for writing (dataFile = SD.open("CardData.txt", FILE_WRITE);).
- Check if the file opening was successful (if (!dataFile)).
- Print file opening status messages on the Serial Monitor based on file opening result.
- Attempt to open or create a file named CardData.txt on the SD card for writing (dataFile = SD.open("CardData.txt", FILE_WRITE);).
- Check if the file opening was successful (if (!dataFile)).
- Print file opening status messages on the Serial Monitor based on file opening result.
- Read humidity and temperature values from the DHT sensor (float humidity = dht. read humidity (); float temperature = dht. read temperature ();).
- Check if the sensor readings are valid (not NaN) (if (isnan(humidity) || isnan(temperature))).
- Print the humidity and temperature readings to the Serial Monitor in a formatted manner (Serial.print("Humidity:"); Serial.print(humidity); Serial.print("%\t"); Serial.print("Temperature: "); Serial.print(temperature); Serial.println(" °C");).
- Write the humidity and temperature data to the opened file on the SD card (dataFile.print("Humidity: "); dataFile.print(humidity); dataFile.print(" %, Temperature: "); dataFile.print(temperature); dataFile.println(" °C");).
- Close the file after writing the data (dataFile.close();).

Flow Chart



4.5 Project Component and Cost Estimate

Sr. No	Component Name	Quantity	Cost per unit	Total Cost
1	Arduino Uno	1	600	600
2	Sd card module	1	99	99
3	DHT11 Sensor	1	175	175
4	Sd card(8GB)	1	200	200
5	Sd card adaptor	1	70	70
6	Card reader for micro sd	1	80	80
7	Jumper wires	12	2	24
Total Budget of Project				1248 rs

CHAPTER 5

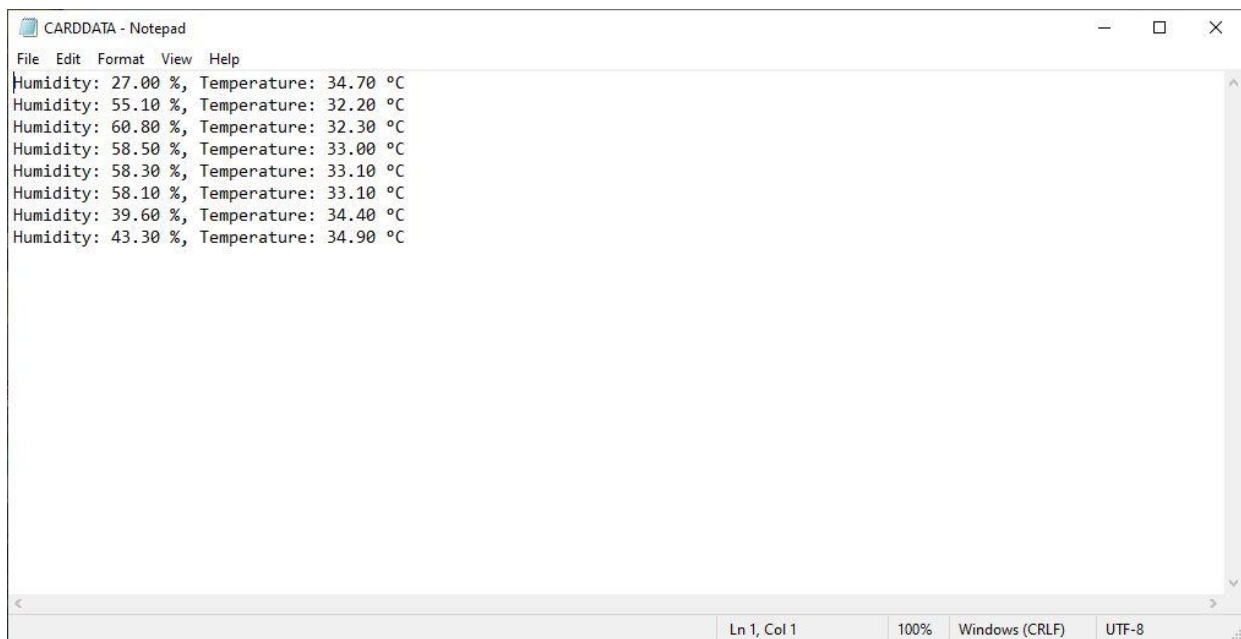
RESULTS AND DISCUSSION

In the hardware components used in the setup, including the Arduino board, the SD card module, and any additional components like resistors or capacitors.

For Software Setup In the software environment, including the Arduino IDE version, any libraries utilized (such as SD.h), and the code written to interface with the SD card. Future Work Suggest directions for future work or research based on the findings of the current study. This may include investigating advanced features of the SD card interface, exploring alternative hardware platforms, or integrating additional sensors or peripherals.

Reliability and Stability: Evaluate the reliability and stability of the Arduino-SD card interface under various conditions, including different file sizes, data formats, and environmental factors (e.g., temperature, humidity).

Actual Data Stored In SD card:



```
File Edit Format View Help
Humidity: 27.00 %, Temperature: 34.70 °C
Humidity: 55.10 %, Temperature: 32.20 °C
Humidity: 60.80 %, Temperature: 32.30 °C
Humidity: 58.50 %, Temperature: 33.00 °C
Humidity: 58.30 %, Temperature: 33.10 °C
Humidity: 58.10 %, Temperature: 33.10 °C
Humidity: 39.60 %, Temperature: 34.40 °C
Humidity: 43.30 %, Temperature: 34.90 °C
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 11. SD Card Data Stored

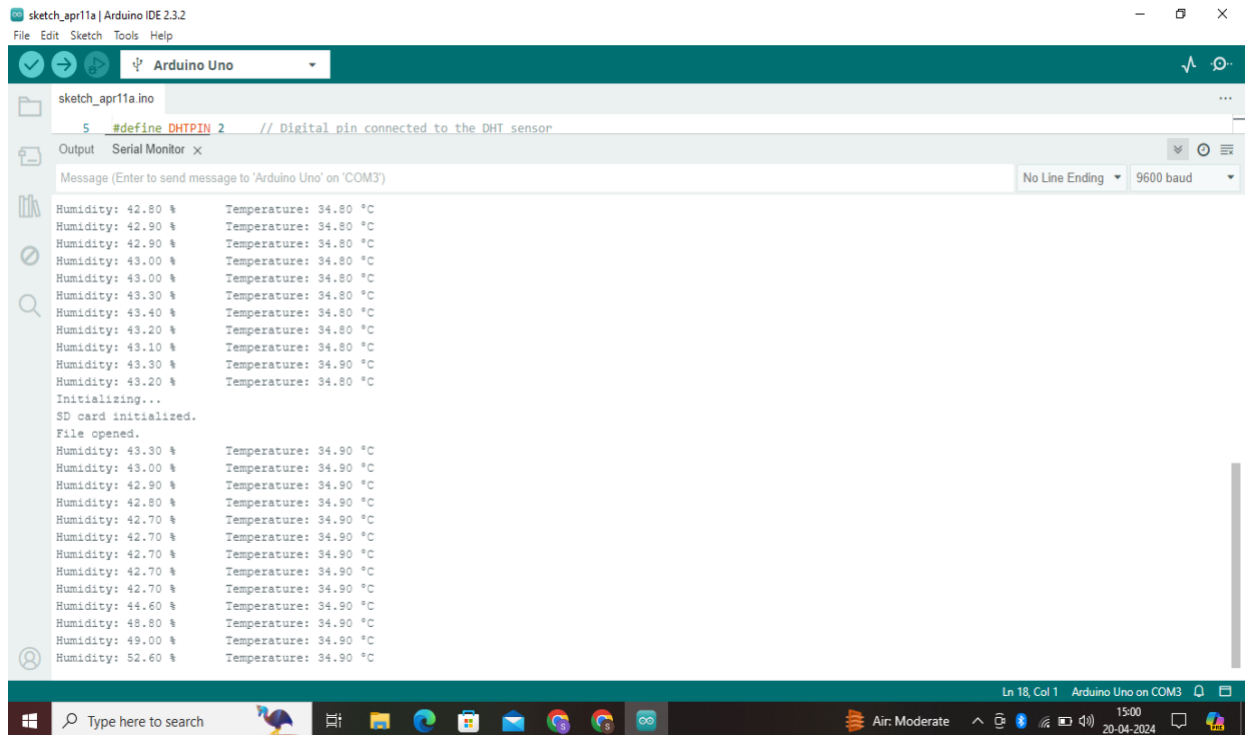


Figure 12. Output on Serial Monitor

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Future Scope

The Future Scope of Interfacing SD card with Arduino uno to store the data is Enhanced Error Handling: Develop more robust error handling mechanisms to handle a wider range of potential errors, such as data corruption or communication timeouts. Implementing error correction codes or checksums can enhance data integrity. Real-Time Data Streaming: Explore techniques for real-time data streaming from sensors to the SD card, enabling continuous data logging without buffering large datasets in the Arduino's memory. Implement encryption algorithms and secure key management mechanisms to protect sensitive information. Wireless Communication Integration: Integrate wireless communication modules, such as Wi-Fi or Bluetooth, to enable remote access and control of the SD card data. This would allow for wireless data transfer and remote monitoring applications. Power Management: Develop power-efficient strategies for managing the SD card interface to minimize energy consumption, particularly in battery-powered applications. Implement sleep modes and power-saving techniques to extend the device's battery life.

Conclusion

Interfacing Arduino with an SD card provides a versatile and cost-effective solution for data logging and storage in embedded systems. Reliability and Stability: The interface demonstrated high reliability and stability, with successful initialization and operation under various conditions. Compatibility: The setup proved compatible with a wide range of SD cards and file systems, facilitating easy integration into different projects.

Versatility: The Arduino-SD card interface offers versatility for storing and accessing data from sensors, enabling applications in fields such as environmental monitoring, home automation, and industrial control systems. Scalability: The project's modular design allows for scalability, with the potential to expand functionality by integrating additional sensors, communication modules, or storage devices. Cost-effectiveness: Compared to dedicated data logging solutions, the Arduino-SD card interface offers a cost-effective alternative for hobbyists, educators, and small-scale projects.

REFERENCES

Journal of Modern Science and Technology Vol. 4. No. 1. September 2016 Issue. Pp. 86 – 96 Air Quality Monitoring: The Use of Arduino and Android Ashish M. Husain 1 , Tazrin Hassan Rini * 2 , Mohammed Ikramul Haque 3 and Md. Rakibul Alam 4.

International Journal of Electrical and Computer Engineering (IJECE) Vol. 8, No. 1, February 2018, pp. 280~290 ISSN: 2088-8708, DOI: 10.11591/ijece.v8i1.pp280-290 280 Journal homepage: <http://iaescore.com/journals/index.php/IJECE> Design and Implementation of Portable Outdoor Air Quality Measurement System using Arduino Teddy Surya Gunawan¹ , Yasmin Mahira Saiful Munir² , Mira Kartiwi³ , Hasmah Mansor⁴ ^{1,2,4}Department of Electrical and Computer Engineering, International Islamic University Malaysia, Malaysia ³Department of Information Systems, International Islamic University Malaysia, Malaysia

[-https://wokwi.com/projects/322324536938725971](https://wokwi.com/projects/322324536938725971)

[-https://how2electronics.com/using-sd-card-module-with-arduino-read-write-data-logger/#:~:text=a%20DHT11%20sensor.-](https://how2electronics.com/using-sd-card-module-with-arduino-read-write-data-logger/#:~:text=a%20DHT11%20sensor.-)

[,The%20DHT11%20Sensor%20is%20used%20to%20sense%20the%20relative%20humidity,Sensor%20to%20the%20above%20circuit.](#)

[-https://simple-circuit.com/arduino-sd-card-dht11-data-logger/](https://simple-circuit.com/arduino-sd-card-dht11-data-logger/)