Faculty of Engineering

University of Ottawa

GNG 1103 C04 – Engineering Design

Fall 2024

Project Deliverable H: Prototype III & Customer Feedback

Course Coordinator: Dr. Muslim Majeed

Group Members: Adam Sirota, Brayden Fitzgerald, Hala Aldahoun, Rami Mosleh, Jordan Draper

Deliverable Date: November 24, 2024

Abstract

This document will display the team's last prototype as well as the analysis and the results gathered during the tests. In addition, it will include a test plan done for the three prototypes and lists all the feedback the team received during the last prototype.

Table of Contents

1 Introduction

In this technical document, we will start our last prototype that is based on our first and second prototypes and based on feedback and comments we gathered from our clients and users. We will include also the analysis and the results we gathered during the tests for each subsystem. In addition, we will have our last updated tables which are: target specifications, detailed design, BOM and prototyping test plan. Thus, with that approach, we will help bat conservationists in Canada to see the effectiveness of the bat boxes.

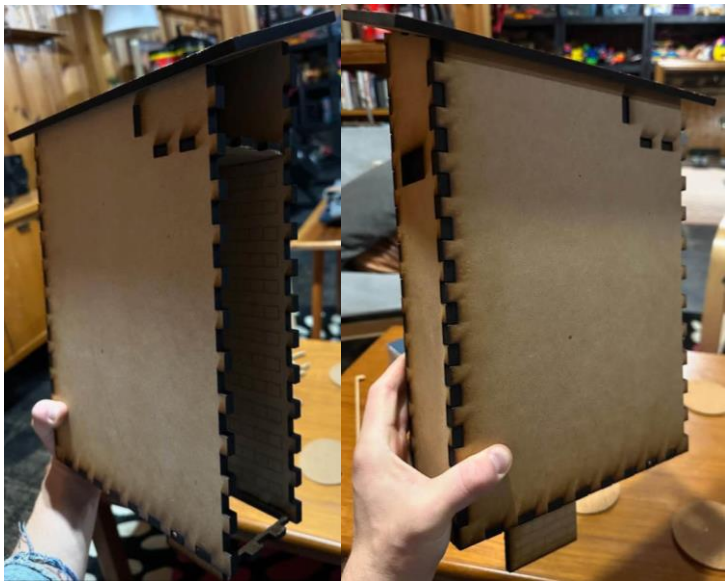
2 Prototype III

In this part, we are going to divide the bat box design into three subsystems, which are the bat box structure, the tracking/storing device, and the power and coding functionality. From these subsystems, our design will track bat visits to that box and will help bat conservationists to see the efficacity of that box.

2.1 Bat Box Structure

This subsystem, bat box structure, needs to be designed to give those bats a home that considers their safety, comfort, and health and withstands any change in weather.

2.1.1 Design



2.1.2 Analysis

The bat box enclosure was built using ¼" MDF plywood. Each piece was laser cut to include finger joints or slits, allowing the structure to be simply and securely connected. In areas where finger joints were determined to be too loose, woods glue was used to strengthen the joints. The dimensions of the box are approximately 16.875" x 11" x 3" including the length of the landing pad and the width of the roof. The weight of the box, not including the Arduino and power supply is 2.338 lb. The use of MDF plywood and the chosen thickness ensure that the box will withstand harsh weather conditions such as snow and rain. The following features are included in the bat box:

Landing pad: The landing pad, located at the bottom of the bat box is designed so bats enter one at a time, ensuring proper data tracking. Additionally, its width was chosen so that adequate space is left at the bottom of the bat box to allow guano to escape

Meshing: On either side of the entrance, the bottom is left open, being covered only by a mesh. This provides protection from predators and weather while allowing guano to come out the bottom of the enclosure

Ventilation: On each side of the bat box, square vents ensure proper air circulation and ventilation

Laser sensor opening located just beneath the entrance on the landing pad, this opening provides an ideal position for the laser sensor to track bat visits

Arduino housing: Located at the top right of the enclosure, this built in housing ensures the Arduino is protected from the elements and the bats themselves while allowing for easy maintenance

Wire channel: To ensure the wires running from the Arduino to the laser sensor, as well as the power supply, are protected, a protective cover is used

Roof: Sloped roof ensures rain and snow buildup do not damage the enclosure

The laser cutting machine was not able to etch grooves into the landing pad or interior, however, the laser cutter was used to detail guides into the wood, allowing the grooves to be chiseled by hand while ensuring dimensioning accuracy.

2.1.3 Results

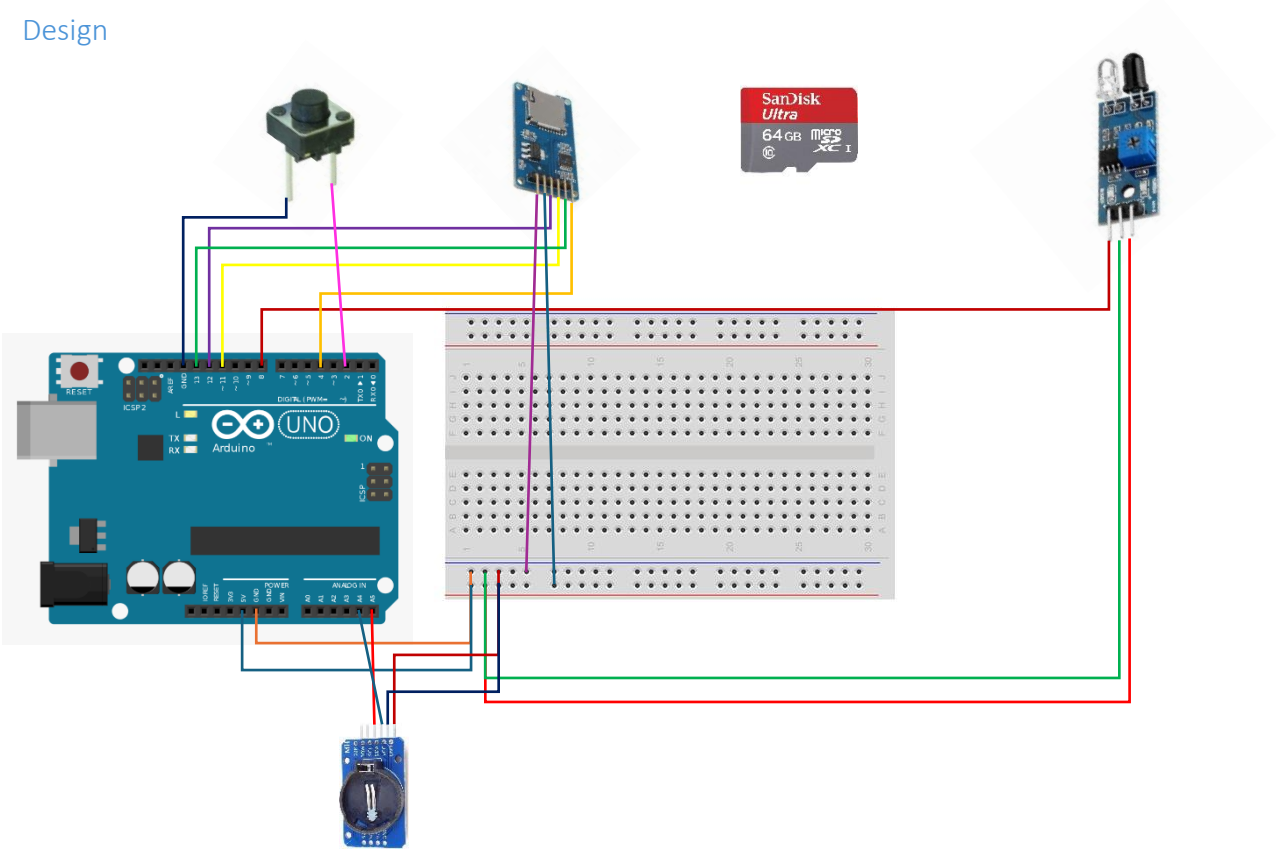
Several tests were conducted to ensure the effectiveness of the bat box. Due to time constraints and issues with the laser cutter, the bat box pieces could not be cut on time so not all tests could be completed. One significant test was to ensure the structural stability and integrity of the bat box.

As shown, the bat box is capable of withstanding forces in excess of 45 lb, which is far heavier than any weight the box will have to withstand.

2.2 Tracking Device

This subsystem, the tracking device, needs to be designed to count the number of bats using this bat box. These data can be stored in a device or transferred to a device that this user always use.

2.2.1 Design



2.2.2 Analysis

- **<u>The Analysis of the Functionality of the Laser Sensor:</u>**
  - For the laser sensor, we added some code lines to the code built for our project. For example, we programmed the Arduino to print "motion detected" in the serial monitor when the sensor detects an obstacle, and "motion ended" when there is no obstacle in front of the laser. In addition, the laser sensor features a green LED, which lights on when there is presence of an obstacle and turns off when there is absence of that same obstacle.
  - Thus, during the test, once we place, for example, our hand in front of our sensor, we see from the LED and from the Serial Monitor the results we obtained and compare them.
  - Finally, with our hand placed in front of the laser, we adjust the distance to 2cm with the potentiometer knob and measure it using a 15cm ruler.

Image 1: The Potentiometer Knob of the Laser Sensor

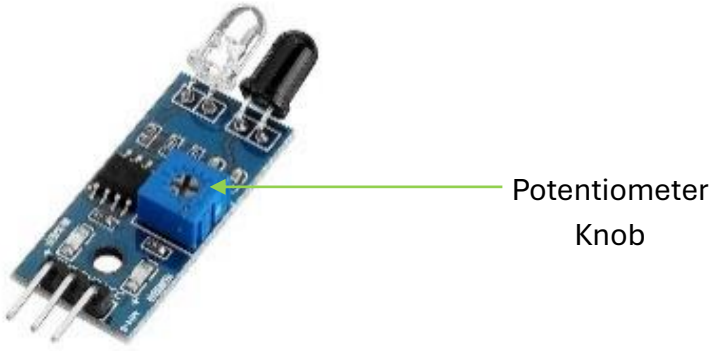


Image 2: The Detection of Laser Sensor at 2cm and above

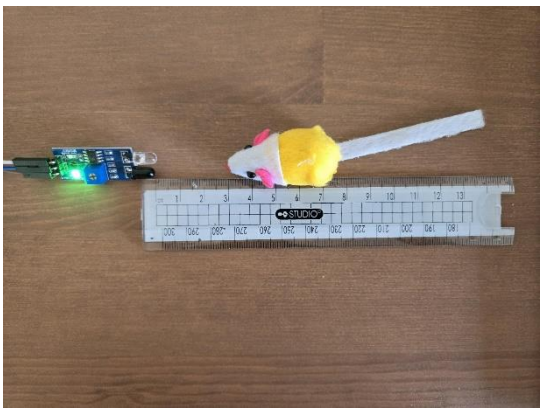


Image 3: The Detection of Laser Sensor at 2cm and below

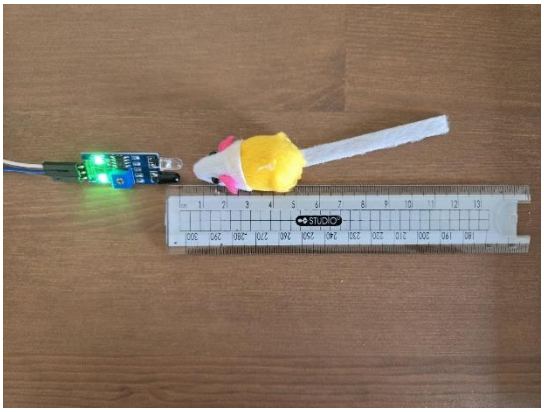
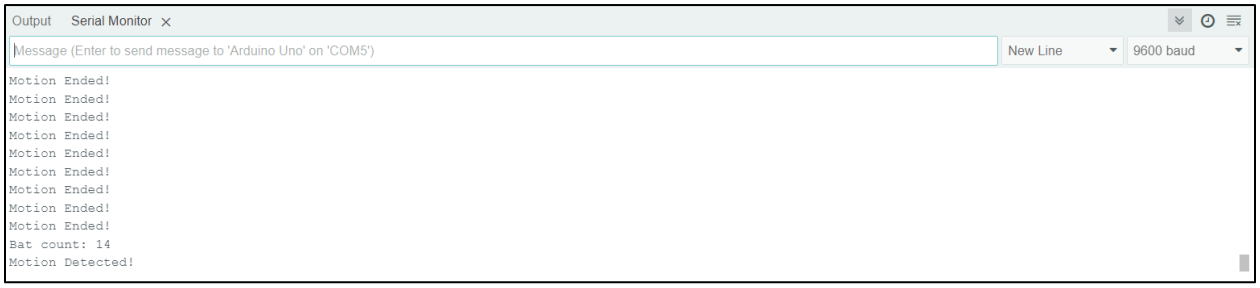


Image 4: The Detection of Laser Shown in the Serial Monitor of Arduino IDE



- **The Analysis of the Functionality of the MicroSD Card:**
  - Before we start with our analysis, just a reminder that the microSD card should have at last 4 files (.txt): one for detection log, one for daily detection, one for monthly detection and one for total bat counts in all the 6 months in general.
  - For the SD card, we added some code lines to the code built for our project. For example, it will display on the serial monitor whether the microSD card is properly initialized and whether the files (.txt) are easy to open and to save data. If not, it would give an error that explains at which level the problem is. It also checks if the file already exists; if it does, the code deletes the existing file and creates a new one to store fresh data. Additionally, it will display on the serial monitor the number of bats counted.
  - Regarding the insertion and ejection of the microSD card, some information needs to be taken into consideration. For example, if the microSD card is ejected from the adapter, it will send a message on serial monitor saying that. However, once the microSD card is reinserted, the program deletes any existing file and starts fresh with new data storage.
  - Finally, during the test, once we simulate some visits using the laser sensor, we take off the microSD card from the adapter from Arduino Uno and then insert it to the computer to see the results. These results (the number of bats) will be compared to the results we obtained from the serial monitor.

    Image 5: The Initialisation of MicroSD Card and All the 4 Files in the Serial Monitor of Arduino IDE

Image 6: The Number of Bat Visits Detected by the Laser Sensor in the Serial Monitor of Arduino Ide

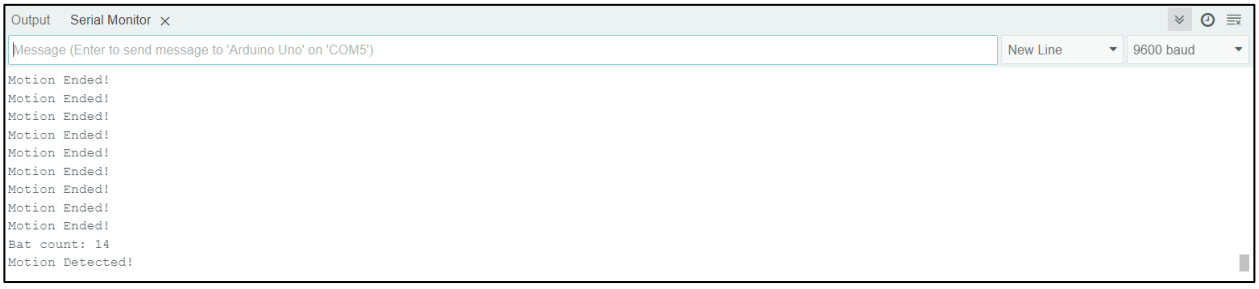
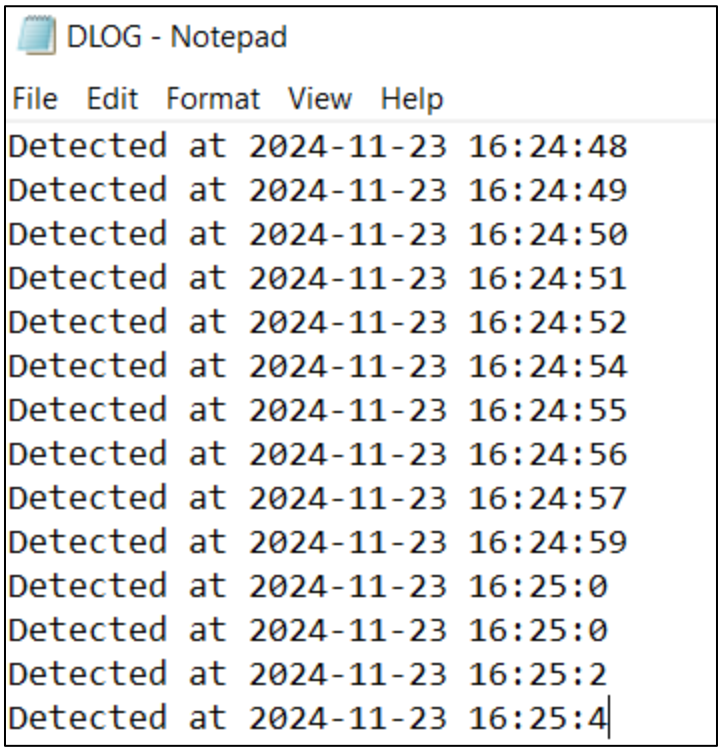


Image 7: The Number of Visits Detected by the Laser Sensor and Shown in dLog.txt



- **The Analysis of the Functionality of the RTC Module:**
  - For the RTC module, we added some code lines to the code built for our project. For example, it will display on the serial monitor whether the microSD card is properly initialized.
  - Finally, during the test, we simulate some visits using the laser sensor and we manually observe the time using a clock when the laser detects an obstacle. In addition, once we finish simulation, we eject the microSD card from Arduino Uno and insert it to the computer to see the real time and date at which the values are stored. We will compare the results we obtained from the clock and from the microSD card.

    Image 7: The Real Date and Time Shown in the Computer

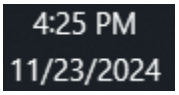
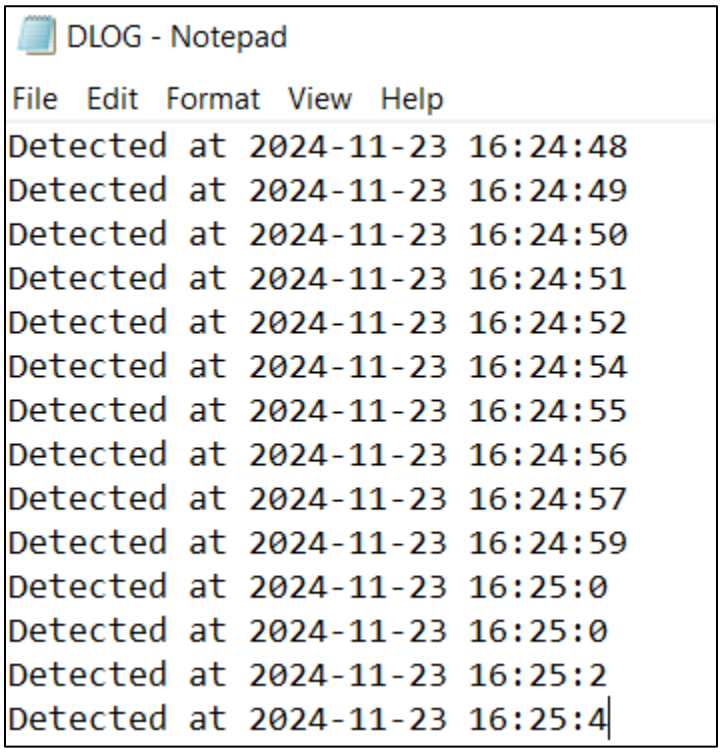


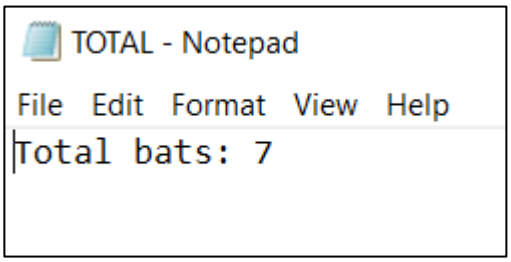
Image 8: The Detection Date and Time Shown in dLog.txt



- **The Analysis of the Functionality of the Push Button:**
  - For the Push Button, we added some code lines to the code built for our project. For example, it will display on the serial monitor the total number of bats when the push button is pressed twice.
  - Finally, during the test, once we simulate some visits using the laser sensor, we press the push button twice to see the results. After that, we eject the microSD card from Arduino Uno and then insert it to the computer to see the results stored at one of the files (which is called total.txt). These results (the number of bats) will be compared to the results we obtained from the serial monitor.

    Image 9: The Real Number of Total Bats Used the Bat Box Shown in the Serial Monitor of Arduino IDE

Image 10: The Real Number of Total Bats Used the Bat Box Shown in the total.txt



2.2.3 Results

Laser Sensor Functionality
The laser sensor consistently detected bat-like movements during the simulations. Each detection triggered accurate logging of bat counts on the serial monitor and the microSD card. The system displayed the bat count in real-time and stored the data for later retrieval.
When comparing the logged data on the microSD card to the serial monitor's display, the results were identical, confirming the precision and reliability of the laser sensor in detecting bat movements.

MicroSD Card Functionality
The microSD card was successfully initialized, and all necessary files (detection log, daily detection, monthly detection, and total bat counts) were created without any issues during testing. The program effectively deleted and recreated files when needed, ensuring fresh data storage. Simulated errors during initialization or file access were displayed clearly on the serial monitor, providing useful feedback for troubleshooting.
The system correctly managed the insertion and ejection of the microSD card. When the card was removed, the serial monitor displayed a message confirming the ejection. Upon reinsertion, all files were cleared, and fresh data storage began without errors.
Data retrieved from the microSD card after simulations matched the bat counts recorded on the serial monitor, confirming accurate logging during laser sensor detections.

RTC Module Functionality
The RTC module successfully recorded timestamps for each bat detection during the simulations. Real-time clock values stored on the microSD card were consistent with manual observations of the time during the laser sensor simulations.
The timestamps recorded for each detection accurately reflected the real-time events, with no discrepancies observed when compared to the clock readings during testing. This demonstrated that the RTC module reliably logged the correct time and date for every bat detection.

Push Button Functionality

The push button worked as intended during testing. Pressing it twice successfully triggered the display of the total bat count on the serial monitor. This count was also correctly logged on the microSD card.

The push button demonstrated consistent functionality across multiple tests, providing reliable responses and confirming its role as an effective way to query bat count data in real-time.

Overall Functionality of the System

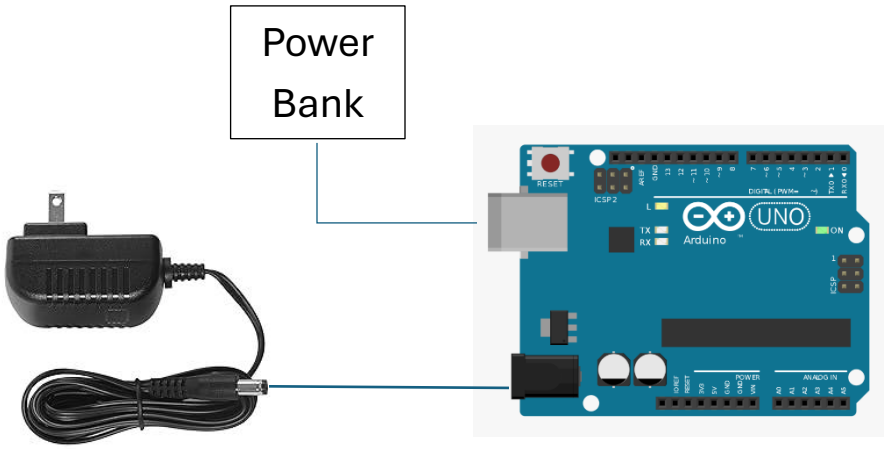
The system operated cohesively, integrating the laser sensor, microSD card, RTC module, and push button effectively. The laser sensor accurately detected bat movements, while the RTC module reliably recorded the corresponding timestamps. The microSD card managed the data storage without any issues, and the push button offered an intuitive way to query total bat counts.

Test simulations confirmed that the system was robust and reliable, with all components working seamlessly to deliver accurate data. The system also handled unexpected scenarios, such as microSD card ejection, without any interruptions to functionality.

2.3 Power and Coding Functionality

The last subsystem, power and coding functionality, ensures that our tracking device has electricity to function and works autonomously without relying on a computer.

2.3.1 Design



```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <RTClib.h>

const int irSensorPin = 8;   // Pin connected to the IR Sensor OUT
const int chipSelect = 4;    // Pin connected to the SD card's chip select (CS)
const int buttonPin = 2;     // Pin connected to the push button

int batCount = 0;           // Daily bat count
int monthlyBatCount = 0;     // Monthly bat count
int totalBatCount = 0;       // Total cumulative bat count
bool lastSensorState = HIGH;  // Last state of the IR sensor
bool currentSensorState = HIGH;
bool lastButtonState = HIGH;  // Last state of the button

File dataFile;
File totalFile;
File detectionFile;          // File to log detection times
RTC_DS3231 rtc;              // Create an RTC object

bool checkSDCard() {
        if (!SD.begin(chipSelect)) {
                Serial.println("SD card initialization failed!");
                return false;  // Return false if SD card initialization failed
        }
        return true;  // Return true if SD card is successfully initialized
}

void setup() {
        Serial.begin(9600);
        pinMode(irSensorPin, INPUT);       // Set the IR sensor pin as input
        pinMode(buttonPin, INPUT_PULLUP);    // Set button pin with pull-up resistor
        // Initialize the SD card
        if (!SD.begin(chipSelect)) {
                Serial.println("SD card initialization failed!");
                return;
        }
        Serial.println("SD card initialized.");
        // Initialize the RTC
        if (!rtc.begin()) {
                Serial.println("Couldn't find RTC");
                while (1);
        }
        Serial.println("RTC initialized.");
        // Set RTC to compile time (use only once, then comment this out)
        rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // Set RTC to the time of compilation
        Serial.println("RTC time set.");
        // Check if files exist and are not empty, then erase their content
        deleteAndCreateFile("dLog.txt");
        deleteAndCreateFile("daily.txt");
        deleteAndCreateFile("monthly.txt");
        deleteAndCreateFile("total.txt");
}
void loop() {
        if (!checkSDCard()) {
                Serial.println("SD card removed, re-initializing...");
                while (!checkSDCard()) {
                        Serial.println("Waiting for SD card to be inserted...");
                        delay(1000);  // Wait for 1 second and retry SD card initialization
                }
```

```
                // Check if files exist and are not empty, then erase their content
                deleteAndCreateFile("dLog.txt");
                deleteAndCreateFile("daily.txt");
                deleteAndCreateFile("monthly.txt");
                deleteAndCreateFile("total.txt");
                // Reset bat count variables after SD card reinsertion
                batCount = 0;
                monthlyBatCount = 0;
                totalBatCount = 0;
        }
        currentSensorState = digitalRead(irSensorPin);
        // Detect when the IR beam is interrupted (bat passes)
        if (lastSensorState == HIGH && currentSensorState == LOW) {
                batCount++;          // Increment daily count
                monthlyBatCount++;     // Increment monthly count
                totalBatCount++;       // Increment total count
                // Get the current time from the RTC
                DateTime now = rtc.now();
                // Log detection time and date
                detectionFile = SD.open("dLog.txt", FILE_WRITE);
                if (detectionFile) {
                        detectionFile.print("Detected at ");
                        detectionFile.print(now.year());
                        detectionFile.print("-");
                        detectionFile.print(now.month());
                        detectionFile.print("-");
                        detectionFile.print(now.day());
                        detectionFile.print(" ");
                        detectionFile.print(now.hour());
                        detectionFile.print(":");
                        detectionFile.print(now.minute());
                        detectionFile.print(":");
                        detectionFile.print(now.second());
                        detectionFile.close();
                } else {
                        Serial.println("Error opening dLog.txt");
                }
                Serial.print("Bat count: ");
                Serial.println(batCount);
                Serial.println("Motion Detected!");
        } else if (currentSensorState == HIGH) {
                Serial.println("Motion Ended!");
        }
        // Get the current time from the RTC
        DateTime now = rtc.now();
        // Check if a day has passed
        static DateTime lastDay = now; // Store last recorded day
        if (now.day() != lastDay.day() || now.month() != lastDay.month() || now.year() != lastDay.year()) {
                // Write daily bat count to SD card
                dataFile = SD.open("daily.txt", FILE_WRITE);
                if (dataFile) {
                        dataFile.print("Bat count for ");
                        dataFile.print(now.year());
                        dataFile.print("-");
                        dataFile.print(now.month());
                        dataFile.print("-");
                        dataFile.println(now.day());
                        dataFile.println(batCount);
                        dataFile.close();
                } else {
                        Serial.println("Error opening daily.txt");
                }
```

```
                        // Write monthly bat count to a separate file
                        dataFile = SD.open("monthly.txt", FILE_WRITE);
                        if (dataFile) {
                                dataFile.print("Monthly count for ");
                                dataFile.print(now.year());
                                dataFile.print("-");
                                dataFile.print(now.month());
                                dataFile.println(":");
                                dataFile.println(monthlyBatCount);
                                dataFile.close();
                        } else {
                                Serial.println("Error opening monthly.txt");
                        }
                        // Write total bat count to a separate file
                        totalFile = SD.open("total.txt", FILE_WRITE);
                        if (totalFile) {
                                totalFile.print("Total bats recorded until now: ");
                                totalFile.println(totalBatCount/2);
                                totalFile.close();
                        } else {
                                Serial.println("Error opening total.txt");
                        }
                        // Reset daily, monthly, and total counts for the next day
                        batCount = 0;
                        monthlyBatCount = 0; // Reset monthly count
                        totalBatCount = 0;   // Reset total count if desired
                        lastDay = now; // Update last recorded day
                }
                // Button press logic
                int buttonState = digitalRead(buttonPin);
                if (lastButtonState == HIGH && buttonState == LOW) {
                        // Button pressed
                        int currentTotalBats = totalBatCount; // Capture the total bat count
                         // Save the total bat count to a separate file
                        totalFile = SD.open("total.txt", FILE_WRITE);
                        if (totalFile) {
                                totalFile.print("Total bats: ");
                                totalFile.println(currentTotalBats/2); // Store the total bat count
                                totalFile.close();
                        } else {
                                Serial.println("Error opening total.txt");
                        }
                        Serial.print("Total bats: ");
                        Serial.println(currentTotalBats/2);
                }
                lastButtonState = buttonState; // Update the last button state
                lastSensorState = currentSensorState;
                // Small delay to avoid excessive reads
                delay(50);
}
void deleteAndCreateFile(const char* filename) {
                if (SD.exists(filename)) {
                        Serial.print("Deleting file: ");
                        Serial.println(filename);
                        SD.remove(filename);  // Delete the file
                } else {
                        Serial.print("File does not exist: ");
                        Serial.println(filename);
                }
                // Create the file (this also clears its content if it exists)
                File file = SD.open(filename, FILE_WRITE);
```

```
        if (file) {
                Serial.print("File created: ");
                Serial.println(filename);
                file.close();
        } else {
                Serial.print("Error creating file: ");
                Serial.println(filename);
        }
}
```

2.3.2 Analysis

- **<u>The analysis of the functionality of Code Written in Arduino IDE:</u>**
  - During the test, we check if the code compiles and then upload it to Arduino to see how different components interact with each other. Depending on all the tests we did for the tracking system, we will evaluate our code as either a success or a failure.
- **<u>The analysis of the functionality of Power Alimentation:</u>**
  - During the test, we checked if the 9V adapter supplies the Arduino Uno with the presence of a backup power. In addition, we test the power failure by unplugging the cord from the wall outlet and by seeing if the Arduino Uno continues with the backup power coming from the power bank.

    <u>Image 11:</u> The Power Supplied from the Wall Outlet

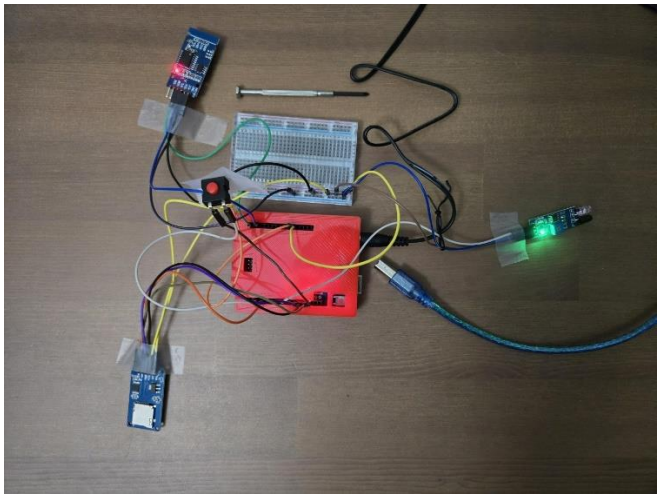


Image 12: The Power Supplied from the Power Bank in Accordance with a Power Failure

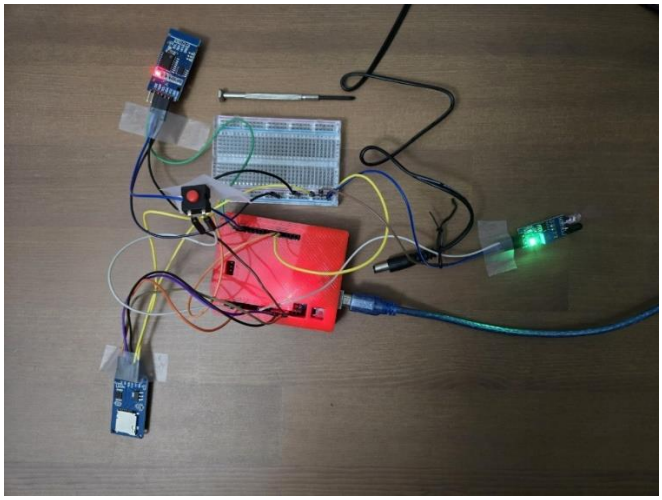
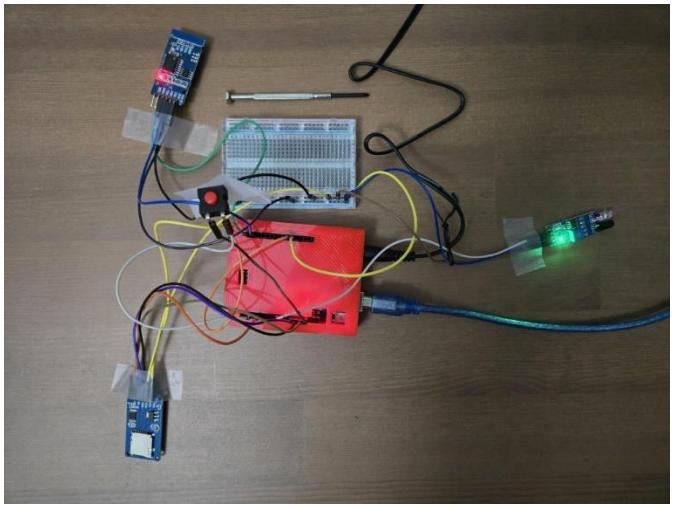


Image 13: The Power Supplied from the Wall Outlet with the Presence of a Power Backup



2.3.3 Results

Code Functionality in Arduino IDE

The code written for the system compiled successfully and was uploaded to the Arduino without any errors. During testing, the interactions between various components, including the laser sensor, microSD card, RTC module, and push button, were observed. Each subsystem performed its intended functions seamlessly, indicating that the code effectively integrated all components.

The tests demonstrated the reliability of the written code, which ensured accurate data logging, real-time functionality, and proper handling of unexpected events like microSD card ejection. The results of the tracking system tests confirmed that the code was a success, supporting the overall functionality of the project.

Power Alimentation Functionality

The power alimentation system provided consistent power to the Arduino Uno during testing. When the 9V adapter was connected to the wall outlet, it supplied sufficient power to the system. The presence of a backup power source, such as a power bank, was tested to ensure uninterrupted operation during power failures.

During the power failure test, the 9V adapter was unplugged from the wall outlet, and the system seamlessly switched to the backup power from the power bank. The Arduino Uno continued to operate without any interruptions, demonstrating the effectiveness of the power alimentation design.

Visual observations (as represented in Images 11, 12, and 13) confirmed:

- **Image 11:** The system received power from the wall outlet via the 9V adapter.
- **Image 12:** The system successfully switched to power supplied by the power bank during a simulated power failure.
- **Image 13:** The system-maintained functionality with power from the wall outlet while the power bank was connected as a backup.

These results indicate that the power alimentation system is reliable and ensures continuous operation of the tracking system under various power conditions.

3 Feedback & Comments

During our Thursday lab, we gathered feedback from TAs and neighboring groups to refine our project and ensure it aligned well with practical needs. One key point of feedback was about the laser sensor's detection range. Initially, we discovered it was limited to 45 degrees, but after researching solutions, we determined that adjusting the angle would broaden the sensor's range, resolving this limitation and ensuring more accurate tracking of bat entries.

Other groups provided helpful comments on our mountable roof design, expressing appreciation for the idea and noting that it seemed both functional and efficient for tree mounting. Some questions arose regarding our design's handling of bat guano collection. We had initially included small openings at the bottom of the box, (like airflow), but realized they could potentially serve as drainage for the guano as well. This feedback prompted us to enhance the opening so that airflow wouldn't be obstructed, and the mesh would remain clear of blockages.

In addition, the client provided valuable feedback regarding the bat box design. They suggested including a backup battery for the tracking devices to ensure uninterrupted data collection in case of power failure. Additionally, the client recommended positioning the laser sensor near the entrance of the bat box, but specifically below it, and not exactly on the entrance, to improve the accuracy of bat detection as they enter. These adjustments will help optimize the sensor's functionality and enhance the overall reliability of the tracking system.

4 Updated Design Framework

4.1 Target Specifications

Table 1: Target Specifications Table

	Design Specifications	Relation (=, < or >)	Value	Units	Verification Method
	Functional Requirements				
1	Track the number of bat visits	=	yes	N/A	Test
2	Attractive Bat Box Shape	=	yes	N/A	Test
3	Predator-Proof	=	yes	N/A	Test
4	Number of bats supported (Bats)	>	15	Bats	Analysis
5	Power Source for Sensor	=	yes	N/A	Test
6	Data Stored Device	=	yes	N/A	Test
	Non-Functional Requirements				
7	Aesthetics	=	yes	N/A	Test
8	Product life (years)	<	8	years	Analysis
9	Lightweight for installation	=	yes	N/A	Test
10	Low-Maintenance	=	yes	N/A	Test
11	Removable parts for cleaning	=	yes	N/A	Test
12	Weatherproof	=	yes	N/A	Test
	Constraints				
13	Bat Box Size (m^3)	<	0.12	m^3	Analysis
14	Eco-Friendly Materials	=	yes	N/A	Test
15	Operating Conditions: Temperature (°C)	=	10 to 35	°C	Test
16	Operating Condition: Humidity (%)	=	40 to 60	%	Test
17	Entry/Exit Opening (m)	<	0.0254	m	Analysis
18	Cost ($)	<	150	$	Estimate, Final check

4.2 Bill of Materials

Table 2: Bill of Materials

Parts	Quantity	Unit of Measure	Unit Cost ($)	Estimate Cost ($)	Link
Hardware Required					
1/4" MDF (18in × 24in)	2	inches	4	8	Link Sections
8 oz. Black Waterproofing Stain	1	ounce	5.98	5.98	Link Sections
11 mm Aperture Wire Mesh (15 cm x 15 cm)	2	centimeters	5.28	10.56	Link Sections
Wood Glue 8 oz.	1	ounce	0	0	Owned
Wood Screws #6 1-5/8"	10	inches	0	0	Owned
Folding Butt Hinges	2	inches	0	0	Owned
Duck-Billed Hasp Lock	2	inches	0	0	Owned
Electrical Components Required					
Arduino Uno	1	EA	15.25	15.25	Link Sections
Breadboard	1	EA	5	5	Link Sections
IR Sensor Emitter/Receiver	1	EA	8.99	8.99	Link Sections
MicroSD card Adapter	1	EA	8.9	8.9	Link Sections
MicroSD card	1	EA	9.99	9.99	Link Sections
DS3231 RTC Module	1	EA	2.99	2.99	Link Sections
Power Supply Adapter 12V	1	EA	13.99	13.99	Link Sections
Power Bank	1	EA	0	0	Owned
Push Button	1	EA	2.25	2.25	Link Sections
4 Jumper Wires (Male-Male)	1	EA	1	1	Link Sections
16 Jumper Wires (Male-Female)	2	EA	1	2	Link Sections
1/4" - 25ft Protective Sleeve	1	EA	9.99	9.99	Link Sections
Junction Box	1	EA	12.79	12.79	Link Sections
Software Required					
OnShape	1	EA	0	0	Owned
Trello	1	EA	0	0	Owned
Arduino IDE	1	EA	0	0	Owned
Libraries					
SPI	1	EA	0	0	Link Sections
SD	1	EA	0	0	Link Sections
Wire	1	EA	0	0	Link Sections
RTClib	1	EA	0	0	Link Sections
Total Cost (Without Taxes)				118.68	
Total Cost (With Taxes)				134.1	

5 Typical Objectives

The objectives for this bat box project include addressing client feedback, ensuring subsystem functionality, verifying design feasibility, and reducing maintenance requirements. Based on the client's input, the entry/exit mechanism must always remain open, ensuring that bats can move freely. This change minimizes maintenance requirements and reduces the risk of obstructing bat movement or impacting tracking accuracy. Another objective is the inclusion of a mesh-emptying solution, proposed as a drawer system that allows easy access for cleaning without disturbing the bats. This reduces the risk of the mesh filling up, which could hinder bat movement or interfere with data collection.

The design's flexibility and feasibility are also prioritized, particularly in the mountable roof, which, while not a primary concern for the client, may require testing in various mounting environments, such as on trees or other structures, to ensure reliability. Testing for ease of access in the drawer system is also planned to ensure it meets the project's minimal maintenance goals. These objectives work together to create a durable, low-maintenance bat box that meets client specifications and environmental needs.

6 Prototype Test Plan

Table 3: Prototype Test Plan

ID	Test Objective	Description of Prototype	Results to be Recorded	Duration of Test
1	Determine if bat box is effective	Test and observe the temperature within the bat box to ensure it is suitable for bats, and also test how the bat box may be affected by weather such as snow. Snow will be put on the roof of the bat box to test how it handles that type of weather.	-The temperature within the bat box to ensure it is safe for the bats inside -Check to see if the bat box can support the weight of snow buildup (Surface area of the roof times height typical snowfall times density of snow)	45 minutes
2	Determine if the coding is functional	Have the Arduino ide code on a laptop connected to what is needed to track visits, and simulate an entry and exit	If the code gives the proper and accurate number of entry/exits that were simulated.	10 minutes
3	Determine if laser sensor is working	Simulate entry and exiting from the bat box using a prop for a given number of times.	The Arduino will record the number of times the laser beam's sensor detects an object at the bat box's entry. Verify that the sensor has accurately recorded the number of entries.	5 minutes
4	Determine if SD card stores data	Simulate visits and plug card into computer and check stored data, also simulate a power outage to see how SD card reacts, also test to ensure that stored data stays the same after you remove the SD card and re insert it into the bat box	Record if SD card stores data correctly, that it can properly operate before and after a power outage, and if SD card keeps data even after it gets removed from the box.	45 minutes
5	Determine if RTC Module gives real time and dates	Simulate visits and plug card into computer and check if the detection happens at real date and time	Record the detection date time and verify if it matches the real date and time	15 minutes
6	Determine if battery aliments Arduino Uno	Simulate visits to bat box, and observe if the code is updating the simulated visits properly, if so than that shows the battery is powering the Arduino uno	If the coding setup is running properly, and the Arduino uno is providing the correct input for the code, we can assume the battery is properly powering the Arduino Uno.	15 minutes
7	Determine if the push button is working	Simulate visits to the bat box, and then press push button	If the number of visits, entrance and exits, give the real value of bats after the push button is pressed	5 minutes

7 Stopping Criteria

Table 4: Stopping Criteria Table

Test	Stopping Criterion
1	The test can be stopped once the snow (10-15 cm to simulate a typical heavy snowfall) has been put on the roof and the temperature, as well as any structural deformations, have been recorded
2	The test can be stopped once a bat entry and exit has been simulated and the code has been observed to be either functioning properly or not
3	The test can be stopped once the given number of entries and exits has been simulated and the laser sensor has been observed to either accurately recorded the number or entries and exits or not
4	The first component of the test can be stopped once the given number of entries and exits have been recorded and the SD card has been observed to be either storing data correctly or not. The second component can be stopped once a power outage has been simulated and the SD card has been tested for functionality. The third component can be stopped once the SD card has been removed from the bat box, reinserted, then tested for functionality
5	The test can be stopped once the real date and time are shown in the detection files.
6	The test can be stopped once an entry and exit has been simulated and the code/battery have been observed to either be powered and functioning properly or not
7	The test can be stopped once the push button has been pressed and the results (ie whether the number of visits, in total) have been observed.

8 Conclusion

The goal of this deliverable was to improve our last prototype based on our team's first and second prototype bat box and tracking system. Through analysis, peer review, and feedback/comments from clients, we were able to review and update, finally, our target specifications, bill of materials, prototyping test plan, and detailed design. With our test plan and stopping criteria well defined, and our overall designs and plans improved through client and peer feedback, we have built a more effective and efficient bat box enclosure.

9 Link Sections

Hardware:

Bat Box Structure CAD:

https://cad.onshape.com/documents/cf85593d9314c67186a69158/w/261c4abd28dc68f89aa972f0/e/e67d684057078f74b29aa4f6

1/4" MDF Sheet (18in × 24in):

https://makerstore.ca/shop/ols/products/mdf/v/M003-1-4-18-NCH

8 oz. Black Waterproofing Stain:

https://www.homedepot.com/p/BEHR-PREMIUM-8-oz-SC-102-Slate-Solid-Color-Waterproofing-Exterior-Wood-Stain-and-Sealer-Sample-501316/203728594

11mm Aperture Wire Mesh:

Link

Electrical Components:

Arduino Uno:

https://makerstore.ca/shop/ols/products/arduino-uno-r3-clone/v/MC001-A

Breadboard:

https://makerstore.ca/shop/ols/products/breadboard/v/C005-HLF

IR Sensor Emitter/Receiver:

DAOKI 5-Pack IR Infrared Obstacle Avoidance Sensor Module for Arduino Smart Car Robot 3 Wire : Amazon.ca: Tools & Home Improvement

MicroSD Card Adapter:

CANADUINO® 3 x Micro-SD Memory Card Adapter for Arduino with 3.3V-5V Converter : Amazon.ca: Electronics

MicroSD Card:

VERBATIM 16GB Premium microSDHC Memory Card with Adapter, UHS-I V10 U1 Class 10, Black (44082) : Amazon.ca: Electronics

DS3231 RTC Module:

CANADUINO® DS3231 RTC Module, 32kB Memory, I2C Interface, Battery Backup : Amazon.ca: Electronics

Power Supply Adapter:

https://www.amazon.ca/dp/B082VVSLGR?ref=cm_sw_r_cso_wa_apan_dp_AC713SVMR8KPGXTE0ZN0&ref_=cm_sw_r_cso_wa_apan_dp_AC713SVMR8KPGXTE0ZN0&social_share=cm_sw_r_cso_wa_apan_dp_AC713SVMR8KPGXTE0ZN0&starsLeft=1&skipTwisterOG=1&th=1

Push Button Switch:

https://makerstore.ca/shop/ols/products/on-off-power-button-pushbutton-toggle-switch

4 Jumper Wires (Male-Male):

https://makerstore.ca/shop/ols/products/jumper-cables-pack-of-10/v/C004-20-MM

16 Jumper Wires (Male-Female):

https://makerstore.ca/shop/ols/products/jumper-cables-pack-of-10/v/C004-20-MF

1/4" - 25ft Protective Sleeve:

https://www.amazon.ca/dp/B071JH14WZ?ref=cm_sw_r_cso_wa_apan_dp_CRCCENJABDHPYTZSM4SR&ref_=cm_sw_r_cso_wa_apan_dp_CRCCENJABDHPYTZSM4SR&social_share=cm_sw_r_cso_wa_apan_dp_CRCCENJABDHPYTZSM4SR&peakEvent=1&starsLeft=1&skipTwisterOG=1

Junction Box:

https://www.amazon.ca/LeMotech-Project-Electrical-Junction-Electronics/dp/B0BQYQ1W7X?crid=1DD9KWGZI9Y7M&dib=eyJ2IjoiMSJ9.PVnnAH2RpagAubtbKxGHlApGrxRNtBiyfyA8SnUJTK_dxRy72h2W6NF_6TPf2IZXciJiX42IH5yprYN0kMirJvpOApjU-gpHoaPhgwy_9WQTUBBilgW1pkz2Ba4NwtqvUEea6opvNYg4DugNZa-E7WQgzdL1TLAZNC8S4g97tLDCZURA2FPCvPg8Slbj2nbeEE1_kgCmldeigu_JSCMZleZKR1vN4vZI8gLmLntQzJ-FC7yFBuCC1iDmRvnJr5UefVxJvV5SI2EO97qKQsPR2tmhsipVOkyMWBztuynld4Q.uEqNnpbxPqEIfU19bVLRytUuySQfs36Q5yRmTcMua-8&dib_tag=se&keywords=small%2Bjunction%2Bbox&qid=1731867587&sprefix=small%2Bjunction%2Bbox%2Caps%2C86&sr=8-5&th=1

Libraries:

SPI: https://docs.arduino.cc/learn/communication/spi/

SD: https://docs.arduino.cc/libraries/sd/

Wire: https://docs.arduino.cc/language-reference/en/functions/communication/Wire/

RTClib: https://reference.arduino.cc/reference/en/libraries/rtclib/

10 References

1. IR Sensor Emitter/Receiver Image: https://hackster.imgix.net/uploads/attachments/1307911/_9cw77Dc0bn.blob?auto=compress%2Cformat&w=900&h=675&fit=min
2. Breadboard Image: https://www.shutterstock.com/image-vector/half-breadboard-vector-illustration-providing-600nw-2306585395.jpg
3. Arduino Uno Image: https://www.pngitem.com/pimgs/m/14-146612_arduino-uno-vector-png-transparent-png.png
4. MicroSD Card Adapter Image: https://store.nerokas.co.ke/image/cache/catalog/MicroSD%20module-500x500.JPG
5. MicroSD Card Image: https://www.bhphotovideo.com/images/images2000x2000/sandisk_sdsqunc_064g_an6ia_sandisk_1170203.jpg
6. DS3231 RTC Module Image: https://m.media-amazon.com/images/I/61jIFJRna4L._AC_SL1333_.jpg
7. Power Supply Adapter Image: https://m.media-amazon.com/images/S/aplus-media-library-service-media/e7062853-0fb1-43a7-a0b8-0aa87483312c.__CR0,0,1600,1600_PT0_SX300_V1___.jpg
8. Push Button Image: https://th.bing.com/th/id/OIP.LL_TiS5naraLfQ7KDedDngHaF9?pid=ImgDet&w=171&h=137&c=7&dpr=2.2