System Design
Formulate

Team 25, MECHTRON 4TB6
Ahmed Nazir, nazira1
Stephen Oh, ohs9
Muhanad Sada, sadam
Tioluwalayomi Babayeju, babayejt

April 5, 2023

1 Revision History

Date	Version	Notes
2023/01/14	1.0	General Updates
2023/01/18	2.0	Final Version
2023/04/05	3.0	Revision 1

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
Formulate	Mechatronics Engineering Capstone Course
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
USB	Universal Serial Bus
TCP	Transmission Control Protocol
IP	Internet Protocol
SPDT	Single pole, double throw
TCP/IP	Transmission Control Protocol / Internet Protocol
SOC	Software On Chip
SPST	Single pole, single throw
TX	Transmit
RX	Recieve
USB	Universal Serial Bus
LED	Light-emitting diode
GB	GigaByte
SD	Secure Digital
PCB	Printed Circuit Board

Contents

List of Tables

List of Figures

3 Introduction

The system design document establishes the group's development considerations for the Formulate system. The motivations which drove each aspect of the design were referenced back to the System Requirements Specification, Hazard Analysis, and Development Plan documents.

4 Purpose

Documentation of the Formulate system's design serves to improve the maintainability, reusability, and understandability of the project. This is accomplished through the system design, software architecture, and software detailed design documents by detailing how the design addressed the requirements outlined in the documents from this document's introduction.

5 Scope

This document in particular focuses on the considerations for the user interface, mechanical, electrical, and communication protocol aspects of the system. All relevant design decisions relating to the requirements were detailed by each aspect of the system, and any visual components used to aid the design were included at the end of the document by each aspect of the system.

5.1   System Context Diagram

Hardware

Sends converted data

Raw data from sensors

Results Database

Data from application

Formulate Testing System

Sends Raw Data

Sensors on testing device

Data from database

Data Visualizer

Accepted data by user

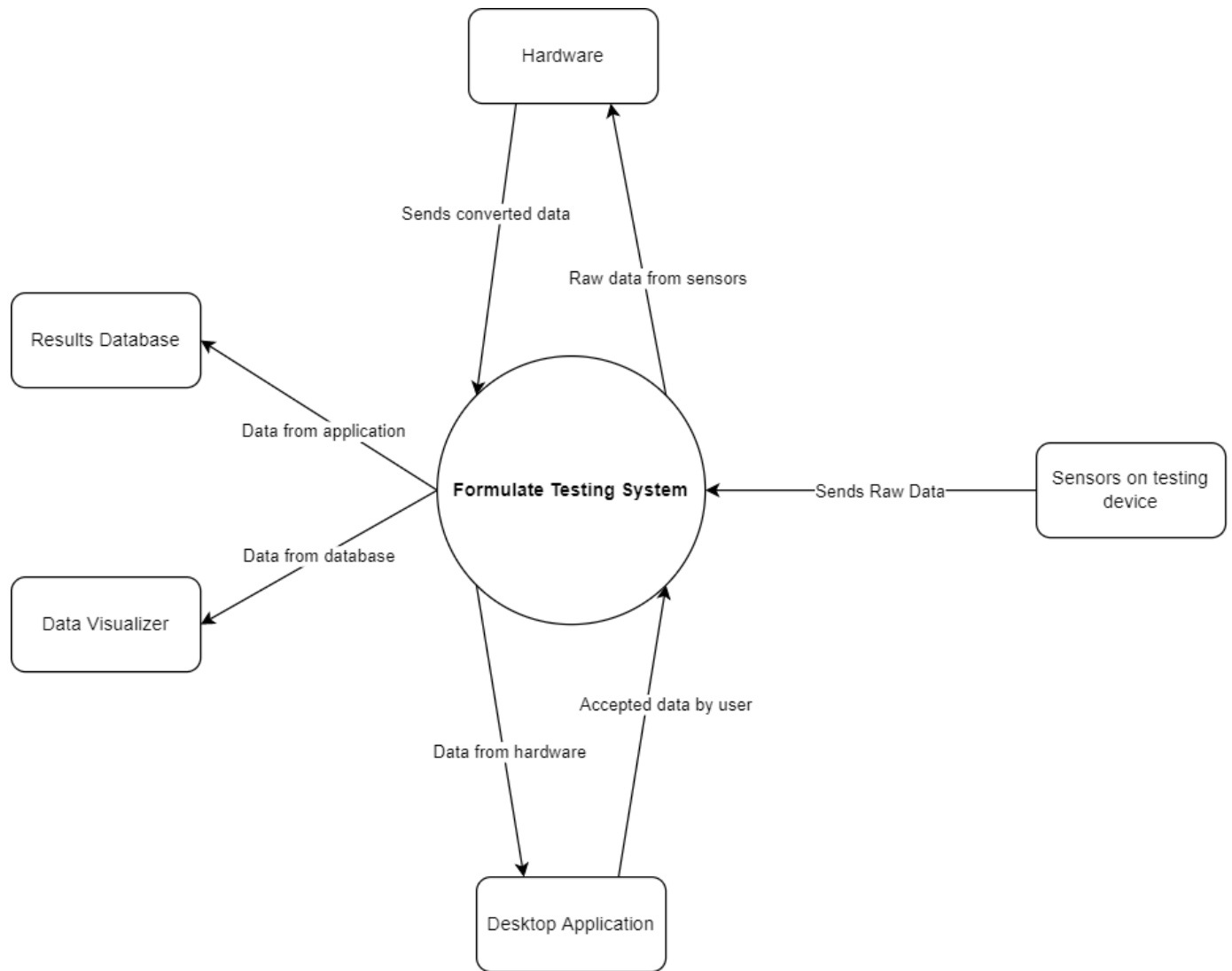Data from hardware

Desktop Application

Figure 1: Formulate Context Diagram

6 Project Overview

6.1 Normal Behaviour

During normal operation the device should work as follows, the user has an option to either connect wired or wirelessly to our device, regardless of which method they choose the results will be consistent between the two. To connect to the device the user will open our desktop application and either create an account or log in to an existing account. This will lead them to the home page where the user will select the connection method as wireless or wired.

Before the user can start any testing they will need to secure our device on the vehicle they are testing with. Our device mounts to the car via a DIN rail. The DIN rail is secured to a relatively flat plane of the car via M6 bolts, and the device has a snap on mechanism onto the DIN rail. After the device is secure the user will need to ensure that all the correct sensors are attached. To conduct any testing the user will first fill out the test parameters of the specific test they are conducting via a sensor configuration page. The user will then go to the test page in our application and start the test. During the test, our hardware will collect the sensor data and send it to the application. When the user stops the test they will be able to preview all the raw data collected from the test and can either decline the data or send it to the database. Before the user sends data to the database, they can add descriptive text to contextualize the test data and provide a picture of the test data. Whent the sensor test data is sent to the database, all the sensor data along with the test description and picture will be sent.

The user can view previous test data using the Power Bi dashboard to read all data in the database with visualized statistics from collected data. The Power Bi dashboard will allow the user to do comparisons between tests and quickly analyze if the tests conducted met the users expectations.
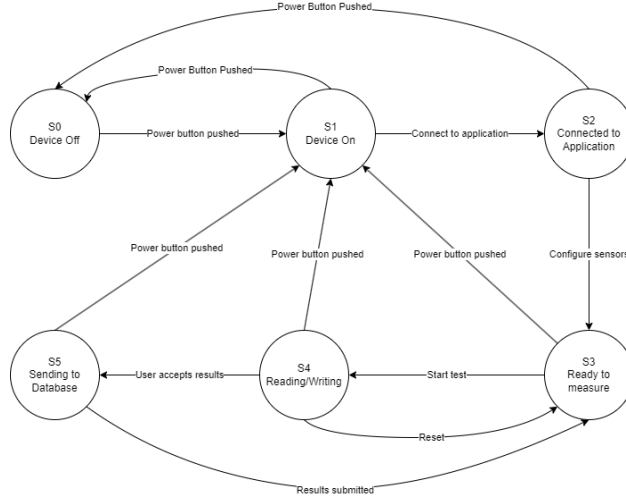
Figure 2: Finite State Machine

6.2 Undesired Event Handling

There might be cases where undesired events occur and to minimize any issues caused to the user we tried to account for them

6.2.1 Loss of Connection

If at any point the connection between the PC and the device fails, such as when the device moves out of wireless connectivity range, a fail-safe test data storage system is required to ensure that no testing data is lost. Whenever data is sent to the application, the data is also simultaneously sent to the local storage on the device. The onboard MicroSD Card will save the entire contents of a previous test that the user can recover in the event of failure.

6.2.2 Excessive Vibration/Shaking

Since our device is expected to be used in an environment with large vibrations, we decided to create a custom PCB instead of using normal jumper cables and a breadboard. This ensures that the electrical connections between the micro-controller, local memory module, and wi-fi module can maintain reliable electrical connections during tests.

6.2.3 Loss of Data Packets

When data from our device is sent to the PC, a situation where data is lost or an entire string of data was incorrectly sent can occur. To ensure we are only reading data from complete bytestrings, each bytestring will start with 'B' and ends with 'E' and our python program checks to make sure that the data we are saving has both those values.
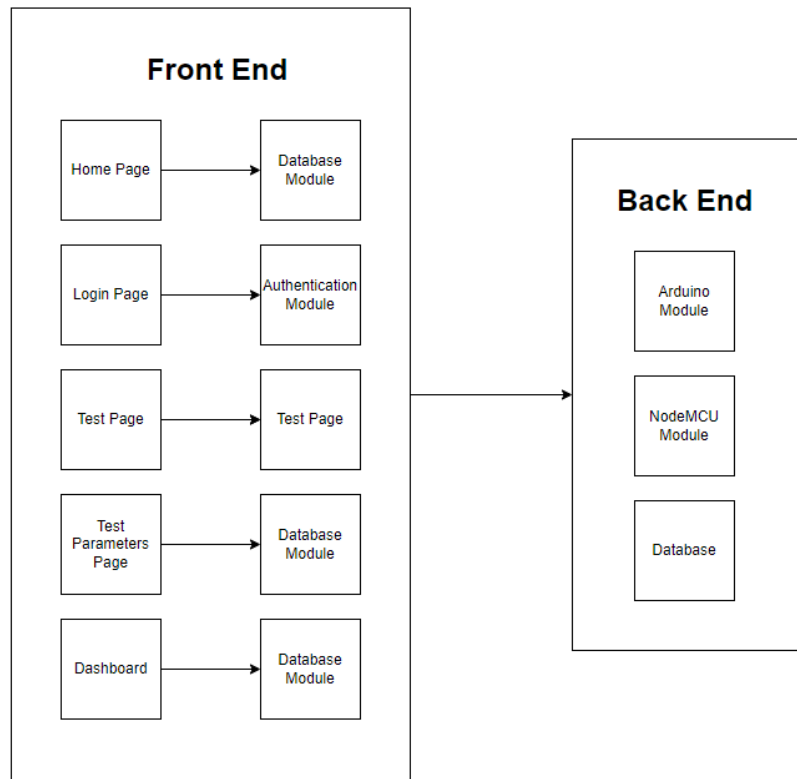
4

6.3 Component Diagram

Figure 3: Component Diagram

6.4 Connection Between Requirements and Design

The following requirements are from the SRS document.

FR1,FR5: To measure vibration we chose to use an accelerometer that outputs in units of G's. For temperature, we are using an LM35 temperature sensor which provides temperature in celsius and has an accuracy of +- 1 degree. To measure humidity we are using a DHT11 sensor which gives us the relative humidity in percent. The accelerometer also provides shock data which just needs some post-processing

FR2: The device transmits data using the UART protocol to the PC when it is connected via a wire.

FR3,FR4: The device will have the start and stop buttons to control the tests inside the desktop application so the user can conduct tests remotely.

FR6: The user can preview the data after a test once they stop the test. A table will populate with all the raw data from the test on the testing page of the desktop application

FR7: After the user previews the test they will be able to decline or submit the test to our database. The table of test values will be sent to our Azure database once the user approves.

FR8: The Power Bi dashboard will connect to the Azure database and will be able to read all the test data

FR9: The device will be clamped down to a DIN Rail

FR11: Our device is going to contain one 9V batteries to power the entire device. 9V batteries allows for the user to quickly swap old batteries for new ones when they lose charge.

FR12,FR15: The ESP8266 wi-fi module allows our device to wirelessly communicate with our desktop application through Wi-Fi. TCP will be used to send data back and forth.

FR14: The screw down style terminal blocks provides a rigid electrical connection between the sensor conductors and the device.

FR16: The device has an onboard programming mode switch allowing users to change the setting on the ESP8266 wi-fi module or the Arduino

FR17: The device will have a screw down style terminal block which will allow users to connect to 5 digital ports and 5 analog ports

7 System Variables

7.1 Monitored Variables

Monitored Variable	Type	Units	Description
m_vibration	Analog	V	A signal monitoring the vibration resistance of the motor
m_humidity	Analog	V	A signal monitoring the humidity of the motor's environment
m_temperature	Analog	V	A signal monitoring the temperature of the motor's environment
m_shock	Analog	V	A signal monitoring the shock resistance of the motor
m_conv_vibration	Digital	g	Converted vibration values that are in useful units
m_conv_humidity	Digital	%	Converted humidity values that are in useful units
m_conv_temperature	Digital	°C	Converted temperature values that are in useful units
m_conv_shock	Digital	g	Converted shock values that are in useful units
m_data_accepted	Digital	T/F	Determines if user has accepted the results and wants to send it to the database

Table 1: Monitored Variables

7.2 Controlled Variables

Controlled Variable	Type	Units	Description
c_green_light	Digital	1/0	Green LED light on testing device that indicates passed measurements
c_red_light	Digital	1/0	Red LED light on testing device that indicates failed measurements
c_sent_to_database	Digital	T/F	Determines if results displayed on the application are sent to the database

Table 2: Controlled Variables

7.3 Constants Variables

Constant	Units	Value	Description
k_temperature_range	°C	5-40	Acceptable ambient temperature values for a Formula Electric motor
k_humidity_range	%	5-85	Acceptable relative humidity values for a Formula electric motor
k_max_shock	g	100	Maximum shock resistance for a Formula Electric motor
k_max_vibration	g	20	Maximum vibration resistance for a Formula Electric motor

Table 3: Constants Variables

8 User Interfaces

8.1 Desktop Application

The user interface for the desktop application is designed through Qt designer, a software for designing and building GUIs through the Qt library. Qt designer generates UI files which can be converted to python scripts that build the static design and layout of the GUI. The desktop application is essentially multiple pages stacked on each other that change based on which buttons are clicked. The GUI is comprised of a left bar menu, top bar, and content pages being in the center, refer to figure 1 and 2 in the Appendix. Navigation through the application is done using the sidebar menu, where users can toggle the full menu and press on which page they want to go. The top bar will be used for extra functionality such as accessing user details, minimizing screen, etc. Users interact with the application using buttons to perform a variety of functions and form fields in which they can enter test/user information.

8.2 Hardware

The user will interface with the hardware as follows, they will mount a sensor to the top of our device and connect the sensor conductors to the device via a terminal block which is on either side of the device. The device will then be mounted to the Formula vehicle and the rest of the operations will take place on the Desktop Application.

8.3 Web Dashboard

The user interface for the dashboard will allow the user to visualize data received from the database through a dashboard using Power Bi. After a test is conducted the user will be able to view the data on the Power Bi website. It will prompt the user to view the data in a variety of graphs and tables which will allow the user to interpret the data in more manageable and understandable way. The design is made with user in mind allowing them to find and view the data in the dashboard in a variety of ways since different types of data will be stored in the database.

9 Design of Hardware

The hardware for our project will include a 3D-printed chassis that will house all electrical components. Our chassis was designed to meet the requirements outlined in our SRS document. The ability to easily mount the device on the Formula Electric vehicle was an important feature of the chassis. We are implementing a DIN rail connection solution to mount the device to any flat, planar surface on the car.

10   Design of Electrical Components

The electrical components were selected to address the functional requirements regarding robust sensor connection points, wireless functionality, and backup data collection capabilities. These capabilities were enabled using hardware modules selected to interface with the embedded device.

The Arduino Uno R3 (Uno) was the choice electrical component for the device's microcontroller. While other microcontrollers on the market were also capable of flexibly collecting data from a multitude of sensors, the Uno stood out as the optimal choice because of low monetary cost in hardware and the relatively high accessibility at large e-commerce platforms for the Formula Electric Team to purchase. In addition, the likelihood of the Formula Electric team using parts of the testing budget on the microcontroller was minimized as many Formula Electric members already posessed an Uno board.
To support testing application flexibility in cases where no direct connections to power were available, one 9 volt batterywith a battery holder was used to provide the Uno with adequate power for on vehicle testing sessions.

A single pole, single throw (SPDT) power switch was used to quickly connect and disconnect the single 9 volt battery with the circuit connecting all electrical components. The switch was oriented such that the common pin actuated to connect either the 9V battery to the circuit or the circuit directly to ground.

Although the Uno provided many functionalities, the standard Uno model did not natively support wireless communication capabilities. As a result, the group chose to integrate a hardware module capable of a Transmission Control Protocol / Internet Protocol (TCP/IP) stack through an ESP8266 Software On Chip (SOC). The electrical component containing the ESP8266 SOC that was selected was the Node MCU 1.0. The Node MCU 1.0 is a development board with the ESP8266 SOC already built onto the Node MCU's PCB in addition to an on-board voltage regulator for the development board's 3.3 volt input power requirement.

A single pole, single throw (SPST) switch packaged with four ports was also required to interface the Node MCU 1.0 module with the Uno during initial device comissioning. Specifically, three of the four ports were used to disconnect the 3.3 volt power, transmit (TX), and recieve (RX) signal between the wifi module and the microcontroller when flashing the wifi module with firmware via a micro Universal Serial Bus (USB) port. When the one time firmware flash is complete, the three switches could be actuated to reconnect the power and signal connections between the two components. Despite the connect/disconnect functionality requiring only three of the four ports, a four port SPST switch was selected due to the high accesibility on large electronics e-commerce platforms relative to three port SPST switches.

Two diagnostic light-emitting diode's (LED) were used to provide the user with feedback on the live transmission status of the wifi module. The first diagnostic signal conveyed when the microcontroller was powered on. The second diagnostic signal conveyed when the wifi module was connected via Wi-Fi to the desktop application.

Backup data storage to local memory in the event of wireless communication error due to the wifi module's failure or device operation outside the wifi router's range necessitated the use of a local memory storage electrical component. A 32 GigaByte (GB) micro Secure Digital (SD) card paired with a micro SD card adapter was used to provide the Uno with local memory storage to concurrently write test data to the SD card while also sending data over wifi to the desktop application. SanDisk, the microSD card manufacturer, was chosen primarily due to their cost effectiveness against other microSD manufacturers as measured by GB/dollar. Similarly, Geek Story was used as the microSD card adapter manufacturer as a result of their cost-effectiveness.

Robust sensor connection components between the sensor and the Uno's input ports were required for tests in physically demanding scenario's as loose or broken connections from high vibration or shock compromised the reliability of the sensor readings and thus the test data. As a result, Phoenix Contact's through hole, 10 port terminal blocks were used because the terminal block style connections provided a stronger connective interface between the sensor conductors and the Uno's ports.

Robust connections between all components of the circuit such as the electrical connections between the Uno, wifi module, micro SD card adapter, switches, and LED's also required a more robust solution relative to jumper wires. As a result, the group chose to design and manufacture a Printed Circuit Board (PCB) onto which all electrical components outlined could be soldered onto the board for a higher strength connection.

The required electrical connections between the micro SD adapters pinouts and the Uno's pinouts were first outlined to organize the connection layout in the table below.

Pin Name	Pin Description	Arduino Port	Arduino Port Description
VCC	5 Volt	5 Volt	Power output
GND	Ground	Ground	Ground
MISO	SPI output from microSD	12	Digital I/O
MOSI	SPI input to microSD	11	Digital I/O
SCK	Synchronize data transmission via Arduino clock	13	Digital I/O
CS	Select slave device on SPI bus	10	Digital I/O

Table 4: MicroSD Adapter to Uno Pinout

The required electrical connections between the wifi module's pinouts and the Uno's pinouts were also outlined to organize the connection layout in the table below.

Pin Name	Pin Description	Arduino Port	Arduino Port Description
3V3	3.3 Volt	3.3 Volt	Power output
GND	Ground	Ground	Ground
TX	Transmit	2	Digital I/O
RX	Recieve	3	Digital I/O

Table 5: Wi-Fi Module to Uno Pinout

A final list of the required electrical components was shown below.

Component	Manufacturer	Part Number	Description	Quantity
Microcontroller	Arduino	Uno R3	System micro-controller	1
Wifi Module	NodeMCU	1.0	System wifi	1
Micro SD Adapter	Geek Story	N/A	Local memory interface	1
Micro SD Card	Sandisk	SDSQUAR-032G-GN6MA	System local memory	1
Battery	Duracell	4330206640	System power	1
SPST Switch	E-Switch	100SPTITI1B4M2QE	System power switch	1
4 Port SPST Switch	TE	435640-2	Wifi signal control switch	1
Through Hole Terminal Block	Phoenix Contact	1715789	System sensor ports	2
LED	Kingbright	WP7113ID5V	LED Resistor	2
Custom PCB	JLCPCB	N/A	System PCB	1

Table 6: Component List

The electrical schematic for the overall circuit containing all components was designed in Kicad and was shown in Appendix C. The PCB layout was also designed in Kicad and was shown in Appendix C. The PCB layout was then fabricated by the manufacturer JLCPCB.

11   Design of Communication Protocols

Our project mainly uses two ways to communicate. The user will either directly plug the device into their computer or they will wirelessly communicate with the device over Wi-Fi. Each of the sensors that are connected to the Arduino uses different protocols to send data.

Device	Communication Protocol
MicroSD Card Module	SPI
Accelerometer (ADXL345)	I2C
Temperature Sensor (L535)	Single Bus
Humidity Sensor (DHT11)	Single Bus
Wi-Fi Module (ESP8266, NodeMCU 1.0)	UART

Table 7: Communication Protocols

To simplify the communication protocol, after the Arduino reads all the values from the sensors it formats them into bytestring to send to either the Wi-Fi module or the PC directly via USB. The bytestring takes the form

`(A<Sensor #><Sensor Val>,B<Sensor #><Sensor Val>,C<Sensor #><Sensor Val>)`

The order in which sensor values are sent is based on how the user configured the sensors in the sensor configuration page. Each bytestring starts with '(' and ends with ')', after the data is sent to the PC our python program parses through the received bytestring and only stores values from complete bytestrings, it checks for the '(' and the ')'.Each of the values in the bytestring is separated with a comma, our python program is then able to split the bytestring and read the correct information from it. The first value in each sensor batch is represented by a letter which corresponds to a specific sensor, the second value represents the number of which sensor is attached, the last value is the actual sensor value. The sensor number represents the number of sensors of that type which is attached, for example a user can have 2 accelerometers attached so they will have an accelerometer 1 and accelerometer 2. When the device is directly plugged into the PC it sends the data over USB via serial and data is sent every second. The other way of receiving data is over Wi-Fi, the Arduino takes the bytestring and sends it using the TX and RX pins to the NodeMCU which then relays the information to the PC via a TCP connection.
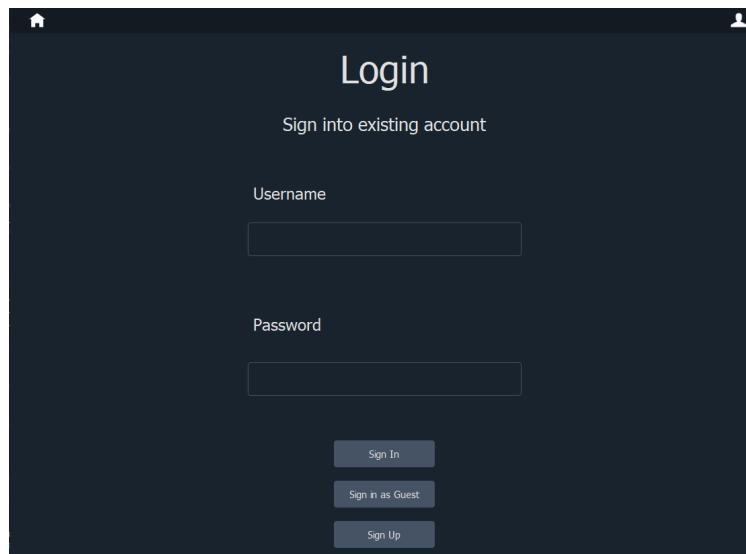
To connect the device to our PC wirelessly there are two main methods we have implemented. The first method is to make our device act as an access point, this means the PC will directly connect to the device and they will exchange information via a TCP connection. The second method would be to connect our device to a central hub and also connect our PC to the same hub, this would require an ethernet connection to be present to plug the hub in. Using both methods our ESP8266 acts as a relay device that passes the information from the Arduino to the PC via TCP.

12   Timeline

Objective	Deadline	Assigned Member
Device mounting mechanism design	2023/01/28	Ahmed
Sensor mounting mechanism design	2023/01/23	Tioluwalayomi
3D chassis print	2023/01/29	Stephen
Test and validate PCB	2023/01/22	Muhanad
Assemble circuit on PCB	2023/01/24	Ahmed
Modularize Arduino code for hardware interface	2023/01/22	Tioluwalayomi
Finalize dashboard design	2023/02/06	Tioluwalayomi
Finalize home page	2023/02/06	Muhanad
Finalize test parameter page	2023/02/06	Ahmed
Finalize test page	2023/02/06	Tioluwalayomi
Finalize login page	2023/02/06	Stephen

Table 8: Timeline

A   Interface

Figure 4: Log in page

Figure 5: Sign up page

Figure 6: Home page

Figure 7: Configure sensors page

Figure 8: Testing page

Figure 9: Submit test page
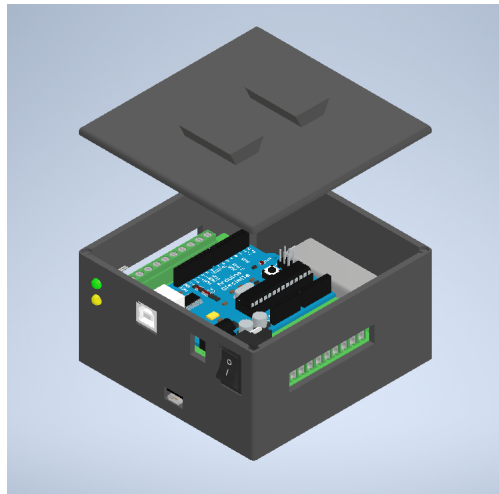
B   Mechanical Hardware

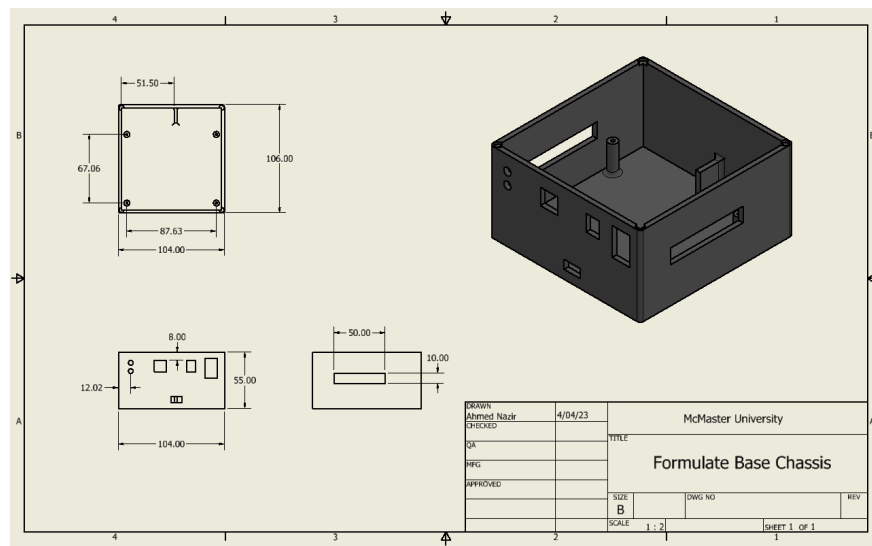Figure 10: Chassis 3D Render

Figure 11: Chassis Drawing

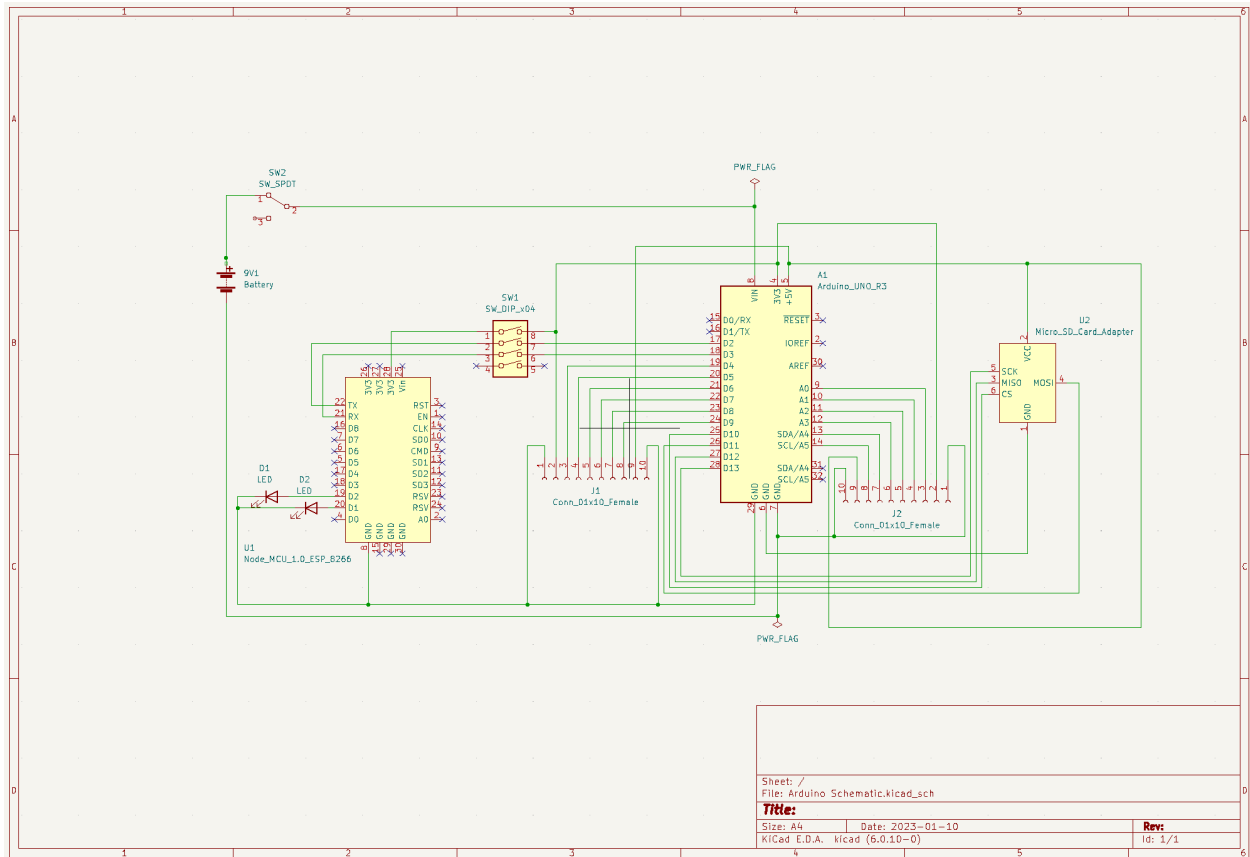# C Electrical Components

C.1 Electrical Schematic

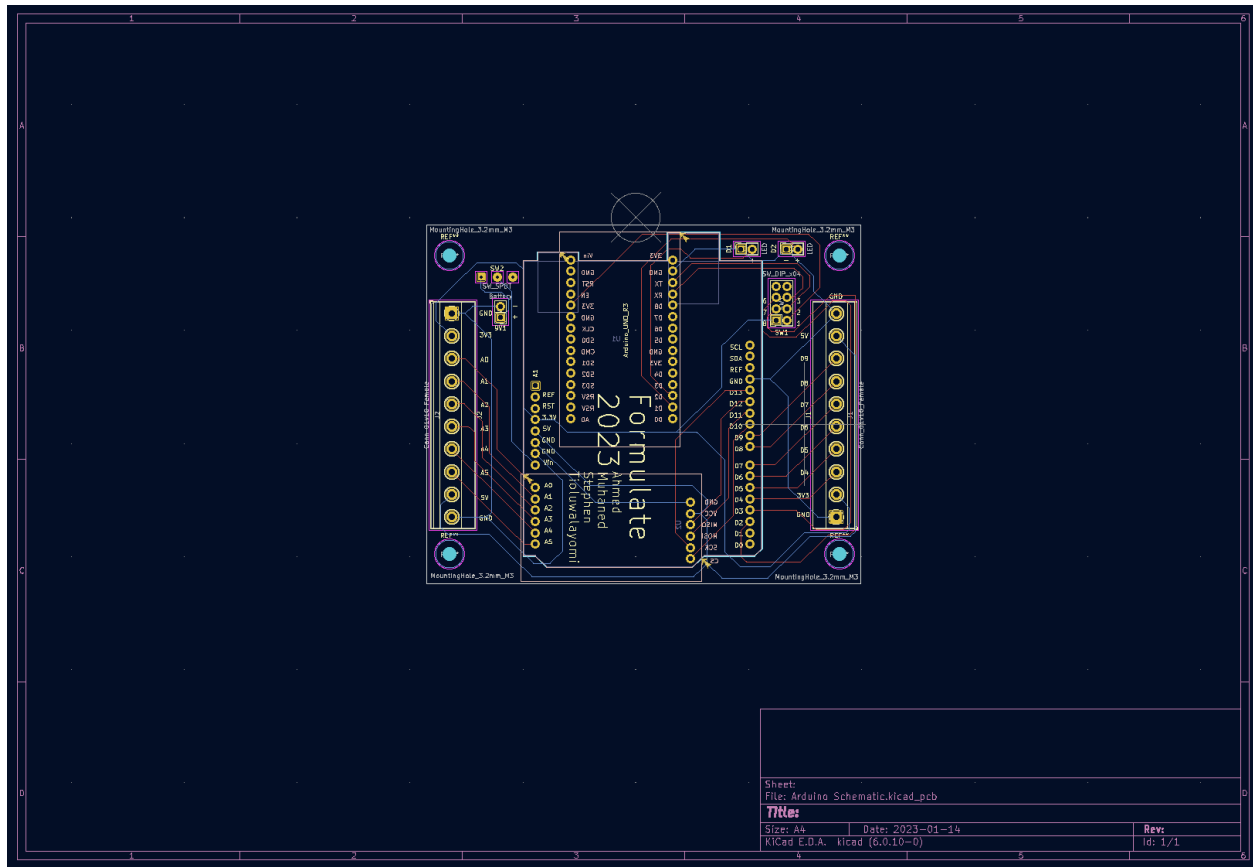Figure 12: Electrical Schematic

C.2 PCB Layout

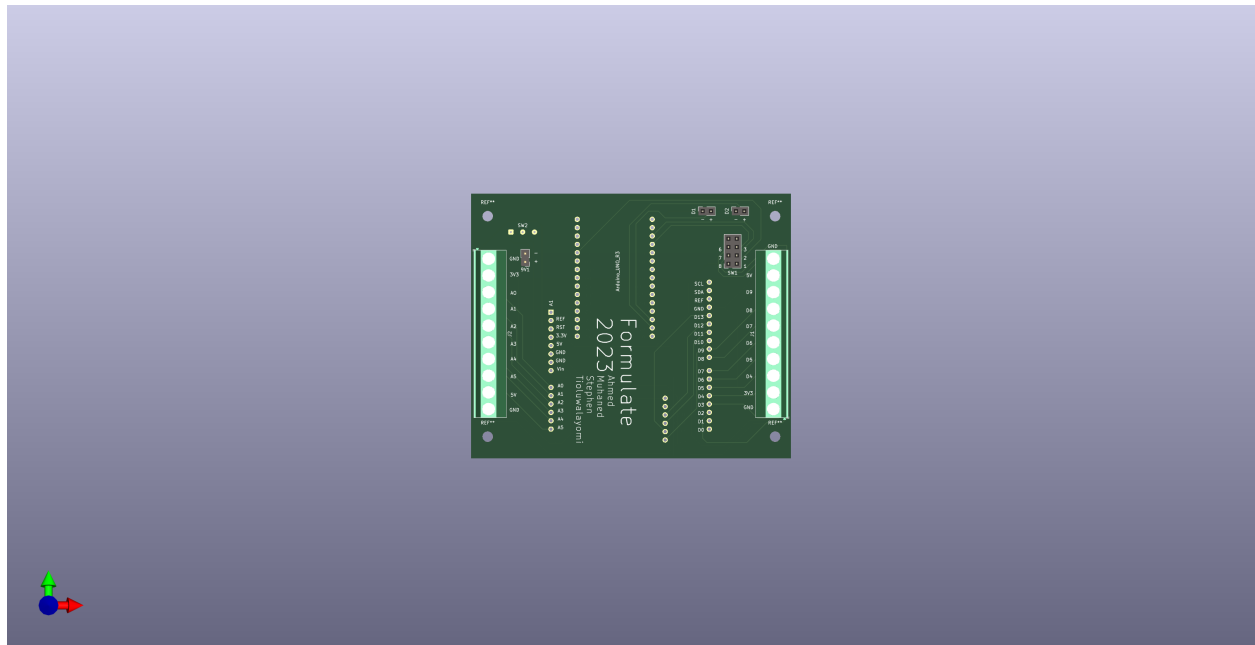Figure 13: PCB Layout in Kicad

C.3   PCB CAD

Figure 14: Top view of the PCB
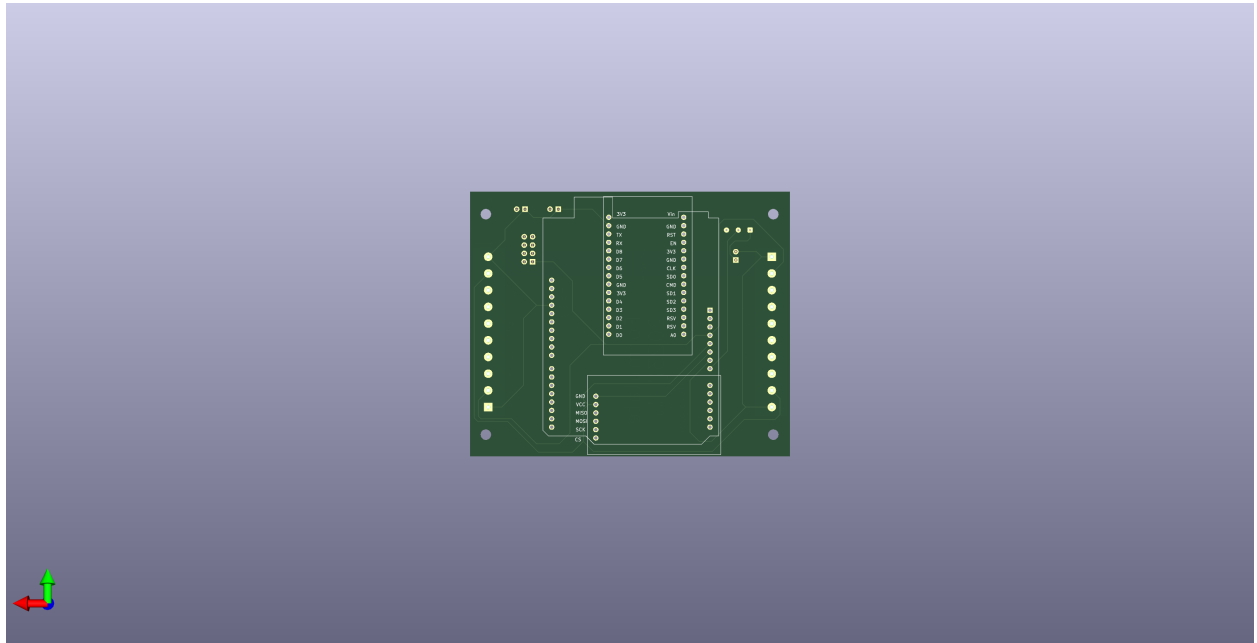
C.4 PCB CAD

Figure 15: Bottom view of the PCB

D   Reflection

D.1   Wireless Communication

1. Some of the limitations of our device would be the range that our device can connect wirelessly. We are currently using an ESP8266 NodeMCU as the wireless module for our device, it acts as an access point and and broadcasts a network which our PC connects to. The NodeMCU operates on a 2.4GHz wifi network which does have long range but if we had unlimited resources we could have used another wireless protocol like LoRaWAN. Using a LoRaWAN connection or something similar could extend the wireless range of our device by a significant amount.

2. An alternate method of connecting to our device would be to use a wireless hub. An external power router could have been used to connect our PC and device to, this would increase the range of the device and also simplify the connecting in our desktop application. The draw back of this solution would be that it requires an external router which would would need a direct ethernet connection. McMaster UTS blocks external router access on campus which would not make this solution feasible. Another method we considered was connecting our WiFi module to McMasters WiFi network, this would solve the range issue as campus wifi is broadly available around campus. There were two main issues with this solution, connecting to McMasters wifi would be complex as it requires a second layer of security to connect to it and also everytime the device would connect to wifi the IP address would change. The benefit of the currenlty implemented solution is that the IP address is static which means connecting to the device would not change.

D.2   Application GUI

1. Some of the limitations of our device would be the range that our device can connect wirelessly. We are currently using an ESP8266 NodeMCU as the wireless module for our device, it acts as an access point and broadcasts a network to which our PC connects. The NodeMCU operates on a 2.4GHz wifi network which does have a long-range but if we had unlimited resources we could have used another wireless protocol like LoRaWAN. Using a LoRaWAN connection or something similar could extend the wireless range of our device by a significant amount.

2. An alternate method of connecting to our device would be to use a wireless hub. An external power router could have been used to connect our PC and device, this would increase the range of the device and also simplify the connecting in our desktop application. The drawback of this solution would be that it requires an external router which would need a direct ethernet connection. McMaster UTS blocks external router access on campus which would not make this solution feasible. Another method we considered was connecting our WiFi module to McMasters WiFi network, this would solve the range issue as campus wifi is broadly available around campus. There were

two main issues with this solution, connecting to McMasters wifi would be complex as it requires a second layer of security to connect to it, and also every time the device would connect to wifi the IP address would change. The benefit of the currently implemented solution is that the IP address is static which means connecting to the device would not change.

D.3 PCB Layout

1. Minimal PCB optimization was made to the layout. Primarily due to the time limitation of a lengthy and costly design, manufacture, test cycle for each PCB iteration, it was not feasible to optimize important PCB characteristics such as the absolute minimal layout size, noise minimization, and maximum structural rigidity. As a result, the team focused on achieving the functional solution in the shortest amount of time which limited optimization considerations.

2. A breadboard circuit with jumper cables and pin header connections was a simpler alternative solution to a PCB design. The breadboard circuit had some benefits such as the inexpensive monetary cost to design, manufacture, and test, and the short time to complete a complete circuit iteration. With that said however, the breadboard circuit lacked the ability to create robust wire connections between components and the ability to design a circuit with a smaller physical footprint. As a result, a PCB layout which functionally replaced the breadboard circuit was chosen as the ability to design a physically smaller circuit with rigid connections through soldered points and terminal blocks was possible.

D.4 Dashboard

1. Minimal design optimization for the dashboard was made since the data that will be visualized is hard to make without the product being fully designed. We have used data that we believe will be similar to the data recieved by our product but there is no guarantee that the dashboard we desgined will be fully optimized for our product when it is fully function.

2. An alternate method would've been to design a website which would store all the data in a table for the user. The user would then graph this table in the format they please by copying and putting the data into a table that graphs the results. This method allowed for a more robust visualizing and storing of test data. This method although being more robust does come with flaws since it would require the user to already to take extra steps to visualize there data making rather inconvient for them. We ultimately came to the decision to use Power Bi to visualize the data since it will seemlessly visualize the data as its being recieved by the database. Even though the same flexibility and robustness is not there it was determined to be a better option because of how compabatible Power Bi and databases are.