

# RF-based Long-range Motion Recognition and Communication System

By

James Tian

Jason Zhang

Joe Luo

Final Report for ECE 445, Senior Design, Spring 2023

TA: Vishal Dayalan

May 2, 2023

Group 45

## Abstract

This report presents a duo-terminal system for long-range motion communication that can be used in various applications. The system reads motion data and sends it through RF communication to another terminal, which decodes and reproduces the motion in real-time with 3D software simulation or mechanical integration. The project successfully implemented RF transmission, PCB design, and mechanical integration, fulfilling the requirements set in the design document. The use of discrete accelerometer and gyroscope measurements with appropriate sampling rate ensures seamless recreation even in a wireless setting. The system offers a reliable, wireless, and interactive solution to long-range motion communication, which can enhance learning experiences in classrooms, simplify controlling robot arms, and improve workplace security, drone navigation, and smart home functions.

## Contents

1. Introduction.....	4
1.1 Section head.....	4
2 Design.....	5
2.1 Design procedure.....	5
2.2 Design detail.....	6
2.2.1 Power Supply subsystem.....	6
2.2.2 Microcontroller.....	7
2.2.3 RF module.....	8
3. Design Verification.....	9
3.1 [Motion Terminal Testing].....	10
3.1.1 [RF Module Testing].....	10
3.1.2 [IMU Testing].....	11
3.2 [Unity 3D Simulation Testing].....	12
3.3 [Mechanical Motor Testing].....	14
3.3.1 [Single Motor Test].....	14
4. Costs.....	15
4.1 Parts.....	15
4.2 Labor.....	15
5. Conclusion.....	16
5.1 Accomplishments.....	16
5.2 Ethical and safety considerations.....	16
5.3 Future work.....	16
Appendix A Requirement and Verification Table.....	18

# 1. Introduction

The science and engineering problem addressed in this report is the need for more intimate ways of communication in the digital age, particularly long-range motion communication. The purpose of the duo-terminal system proposed in this project is to provide a reliable, wireless, and interactive solution that can be used in various applications such as classrooms, controlling robot arms, workplace security, drone navigation, and smart home functions.

The report will cover the design, implementation, and evaluation of the system, including the RF transmission, PCB design, and mechanical integration. The upcoming chapters will delve into the details of the design and implementation process, outlining the challenges and solutions. The main conclusions of the project are that the system is capable of providing a seamless recreation of motions in a wireless setting, and it offers a practical solution to long-range motion communication in various applications.

## 1.1 Section head

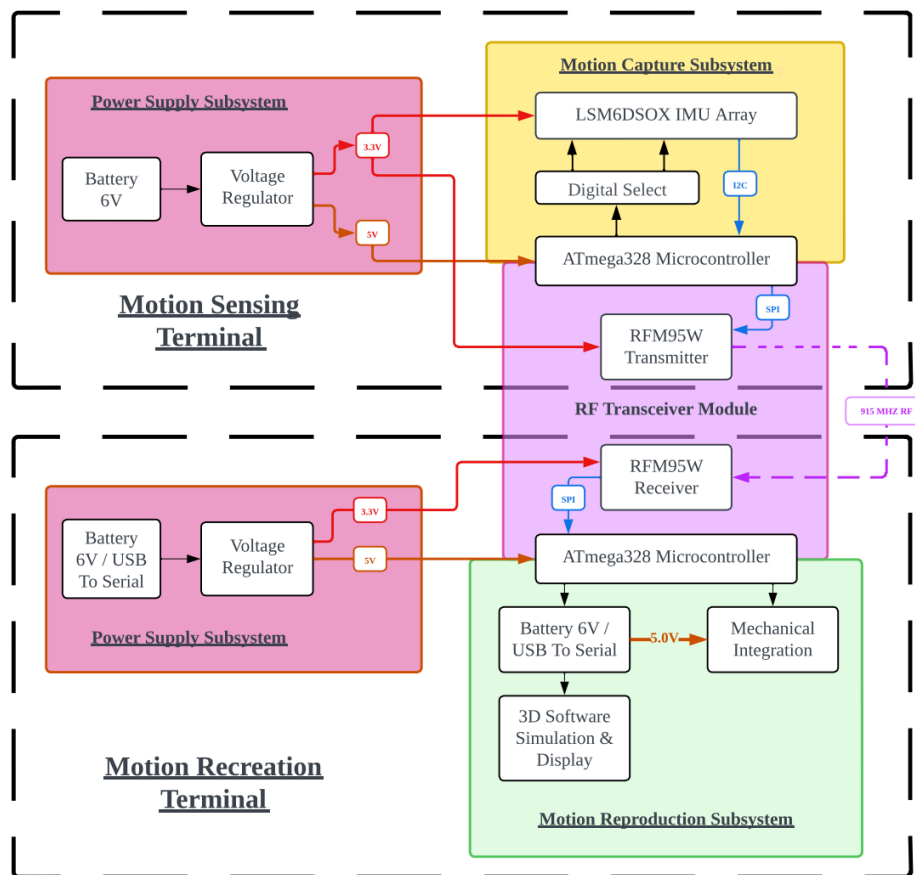


Figure 1 Block Diagram for all subsystems

## 2 Design

### 2.1 Design procedure

The origin block diagram is primarily divided into two remote terminals as indicated by the dashed box. They communicate via 900mhz RF waves. Each terminal consists of a power supply, a motion capture or reproduction system, and RF module. The transmitting end takes measurements from the MEMS sensor array, while the receiving end decodes the data and commands a software or hardware reconstruction of three-dimensional motion.

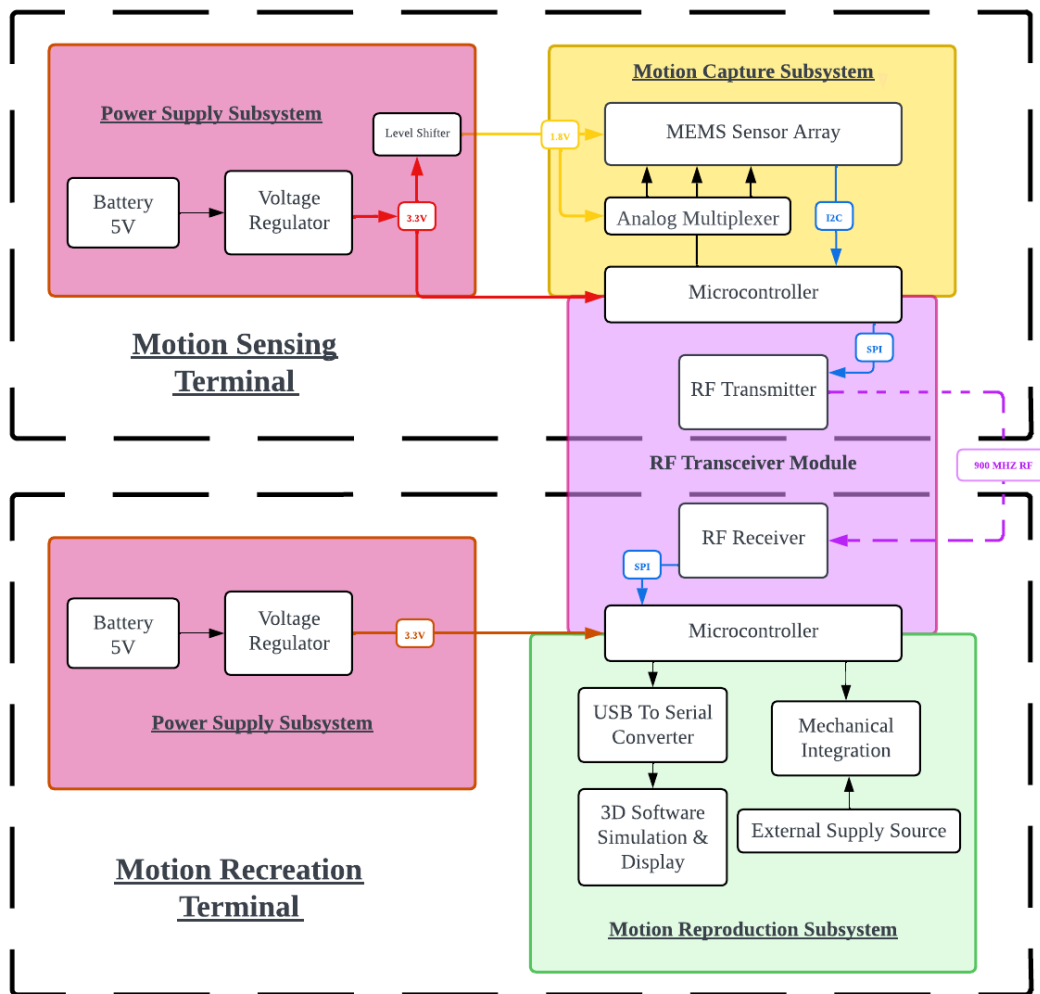


Figure 2 Original Block Diagram

In the new block diagram, the power supply subsystem is changed into two voltage regulators. One of which outputs 3.3V and the other outputs 5V. The USB to serial converter was replaced with a USB to

serial adaptor. The details of the design of each subsystem and the reason for all the changes are explained in the next section.

## 2.2 Design detail

### 2.2.1 Power Supply subsystem

The power supply subsystem originally consisted of one voltage regulator and some bypass capacitors. During the design process we end up changing the single 3.3V regulator to a 3.3V and a 5V regulator. The main reason for this change is that the microcontroller that we acquired from the supply shop operates at 5V instead of 3.3V. We could potentially replace it with a 3.3V version to save space but due to lack of time we don't want to risk waiting for weeks of shipping.

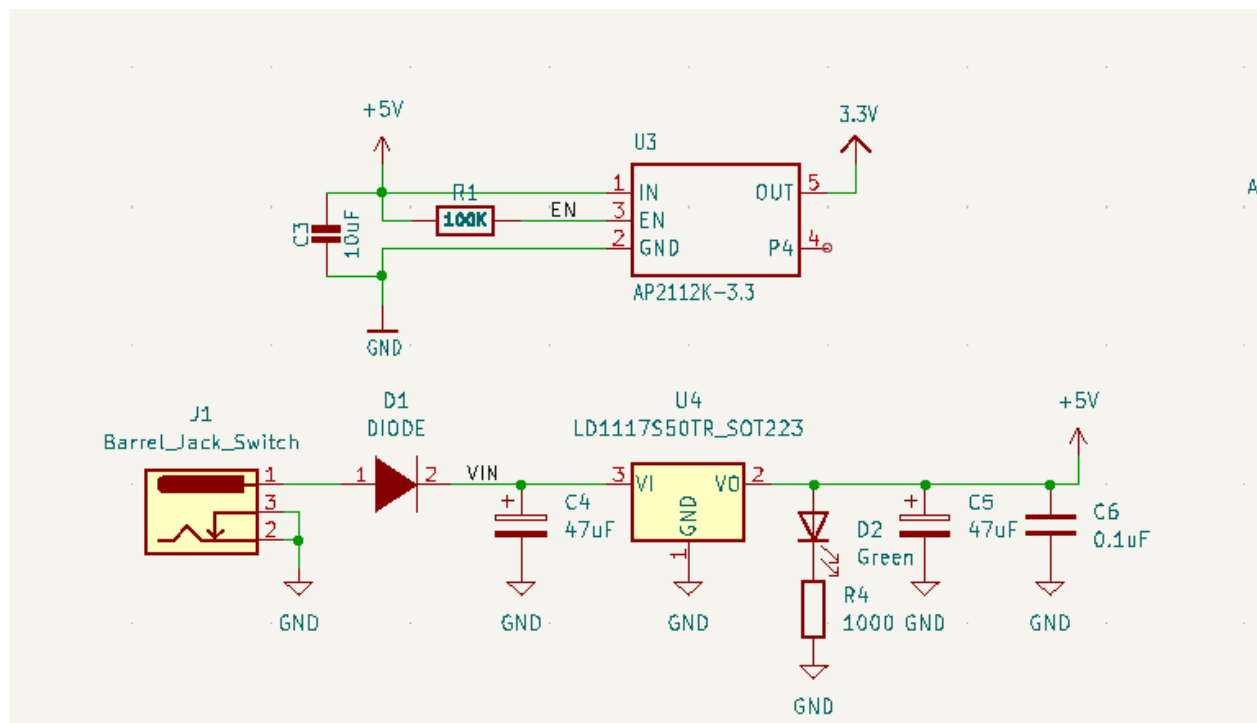


Figure 3 schematic for the power supply subsystem

In addition, to simplify the connection to the power source, a barrel jack is added so we can connect the system directly to the battery. We could also choose to connect the battery pack directly to the PCB but that would probably require us actually making a 3D-printed case for the circuit. We also added a green LED at the output of the 5V voltage regulator. This allows us to see if the system is currently powered and is a good indicator during testing. The diode is added so that we protect the circuit from inverting current, but it's not absolutely necessary if the user is careful. Figure 4 shows the 3D layout of the voltage regulator. We chose through hole resistors and LEDs thinking it would make soldering easier, but we didn't realize that there was actually an oven in the lab until the end of the semester. So potentially we could replace those parts with smaller SMD components so we can save size.

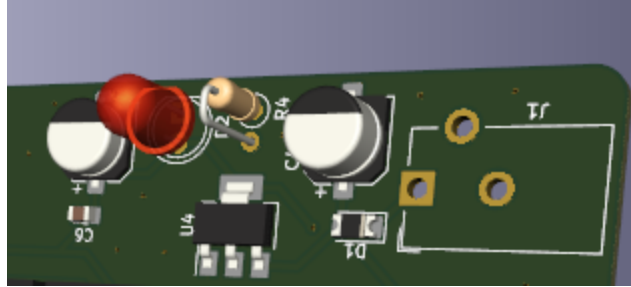


Figure 4 5V voltage regulator layout

The 3.3V power supply design is similar to the design in Adafruit RFM95W LoRa Radio Transceiver Breakout [2]. We decided not to change anything since we are not very familiar with RF modules and we do not want to break anything by changing that part. Also the size of the original design is extremely compact which allows us to keep the PCB as small as possible.

## 2.2.2 Microcontroller

We decided to use ATmega328p as the microcontroller of our system. The reason is that ATmega328p is the same chip that is used in the Arduino UNO which we are very familiar with. In order to integrate the chip to our own design, we have to first understand how the microcontroller works on Arduino UNO. In “Build an Arduino Uno R3 From Scratch” [3], the author explained that the standard Arduino uno is actually very “over built” for our design.

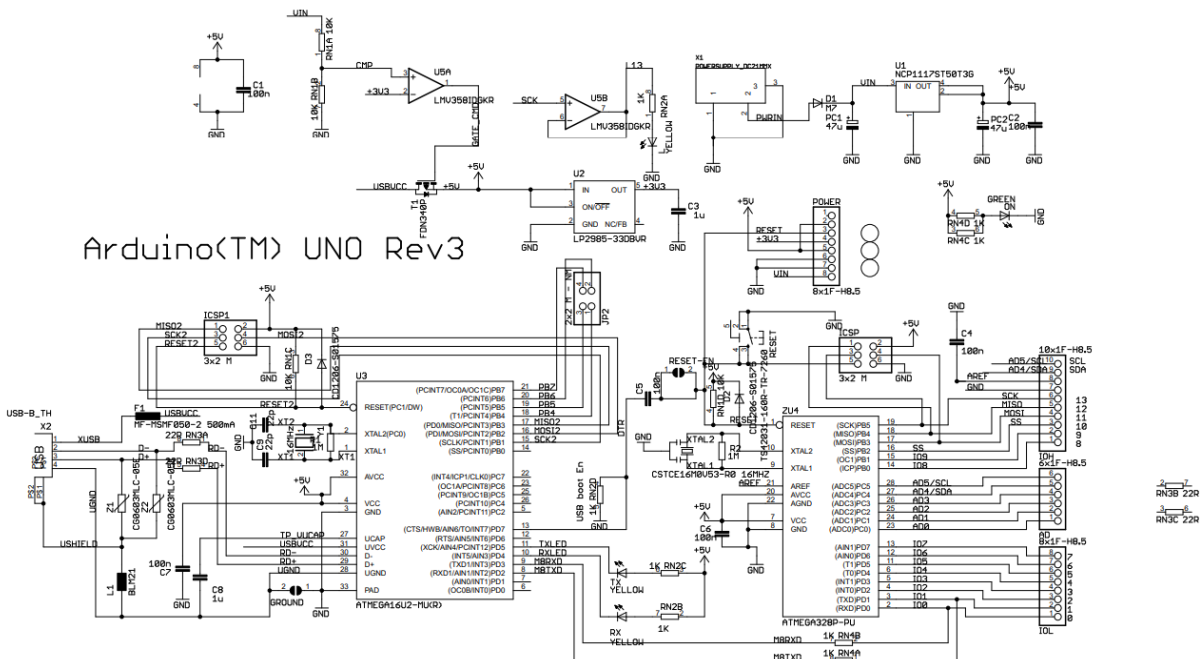


Figure 4 Arduino UNO rev3 [4]

As we can see in the schematic above, the Arduino UNO actually has two microcontrollers – ATmega16u2 and ATmega328p. ATmega16u2 and the entire left half of the schematic are dedicated to a USB to serial

subsystem. During the design we realized that we simply do not require that subsystem since we only need to program the motion capture system once so we don't need to build a giant system just to program the chip. In addition, we found out that in the lab there are usb to serial adaptors which provide the exact same function as the subsystem. The adaptor requires pin ground, +5V, RX, TX, and DTR, so we added a 1x6 connector so we could easily program the microcontroller without soldering it onto the board.

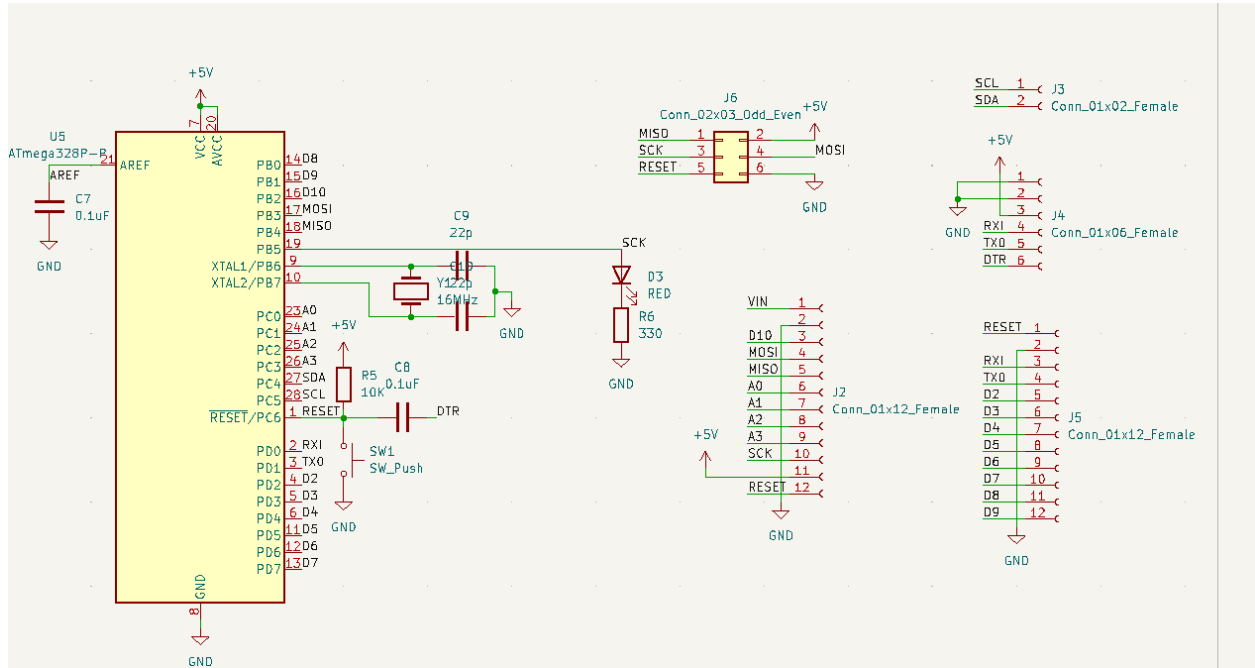


Figure 5 Microcontroller

We also added some essential components such as crystal oscillator, auto reset and some bypass capacitors. The 2x3 pin was there since the usb to serial adaptor is not actually capable of burning the bootloader of the microcontroller. We could potentially burn the bootloader on the breadboard since our microcontroller is the through hole version, but we decided to include the pin since it saves us time to gather extra components necessary for us to burn the bootloader on a breadboard. We also include some additional pins for all inputs and outputs of the microcontroller just in case we need them during testing. The potential improvement here is similar to the power supply, which is replacing all the through hole parts with surface mounts since they are generally smaller and could save us some space to maybe even include the usb to serial module.

### 2.2.3 RF module

Since none of our team members have any experience with RF modules. Our team decided to adapt the RFM95W LoRa Radio Transceiver Breakout [2] from Adafruit. The part that we were most worried about with the original design was the level shifter that takes 5V signal and output 3.3V signal. The pins of that level shifter are so small that it seems impossible to solder. Luckily we were able to consult the TA and realized the reflow oven should be able to solder the component if we order a stencil thin enough to avoid excessive soldering paste. Another issue that we encountered was that the level shifter that

Adafruit was not long on was on the market and we have to find a similar replacement which turns out to be an updated version of the original. We did try to modify the original design, however the engineers at adafruit did such a great job with the layout that it was as compact as it can be with the given component. Especially the antenna which was designed to be as close to the RFM95W chip as possible and is surrounded by ground vias.

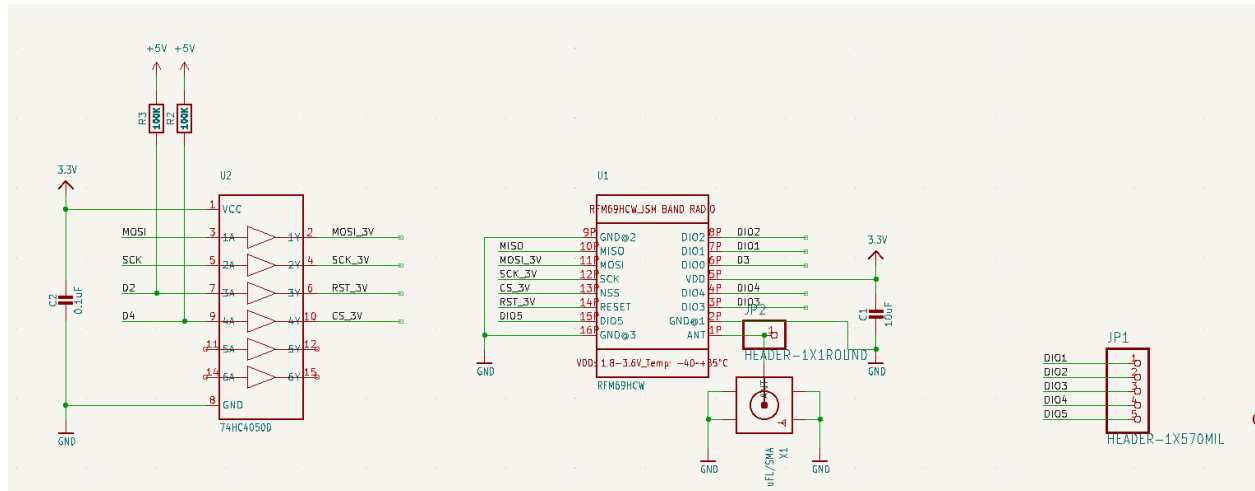


Figure 6 RF module schematic [2]

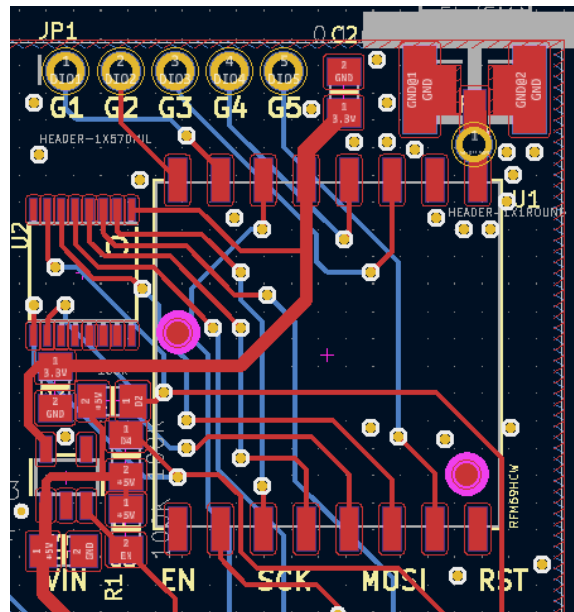


Figure 7 RF module layout [2]

### 3. Design Verification

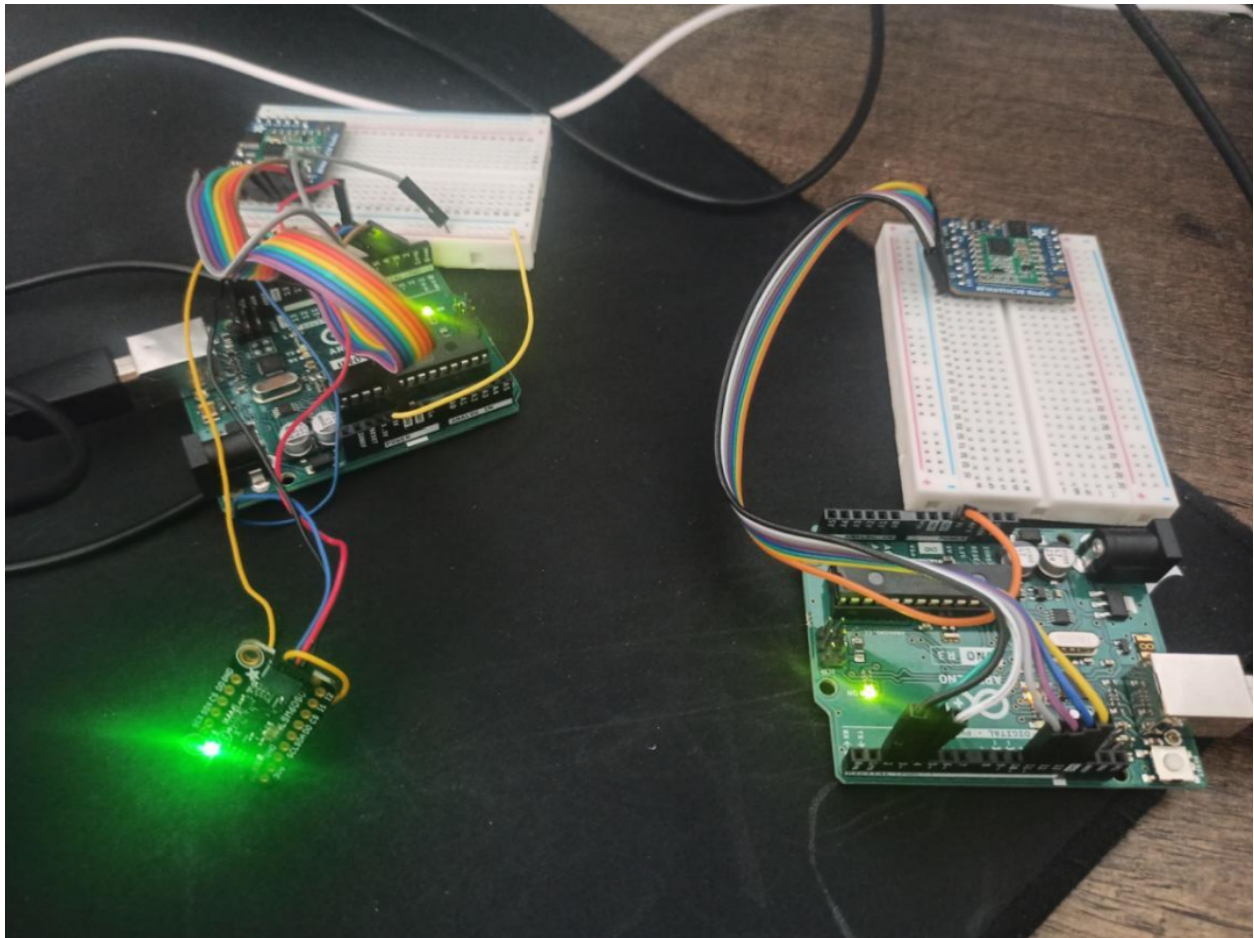
The unit test and qualitatively divided into four major sections – motion terminal testing, Unity 3D simulation testing, and mechanical motor testing. Out of all four of the described testings, three of them can be performed via serial or console monitor printouts.

In order to ensure the freedom of movement, we decide to separate the IMUs from primary PCBs. They are connected to the main circuits with STEMMAQT cables through the I2C communication protocol. In favor of its great capability and strong compatibility, we use LSM6DSOX 6-DOF IMU. As for the radio module, the final design uses RFM95, but for our experimental setup, we use RFM69 radio module that fixes at 900Mhz. Admittedly, the signal is choppy, and packet loss is a legitimate concern. We later included antennas to the PCB which effectively boost the performance of longer range communication. For the microcontroller, we use ATmega328P on Arduino Uno REV3 development board. Further similar testing is performed on the PCB board with a customUSB-to-Serial connector port.

### 3.1 [Motion Terminal Testing]

#### 3.1.1 [RF Module Testing]

Circuit assembled for RF module and IMU testing is shown below:



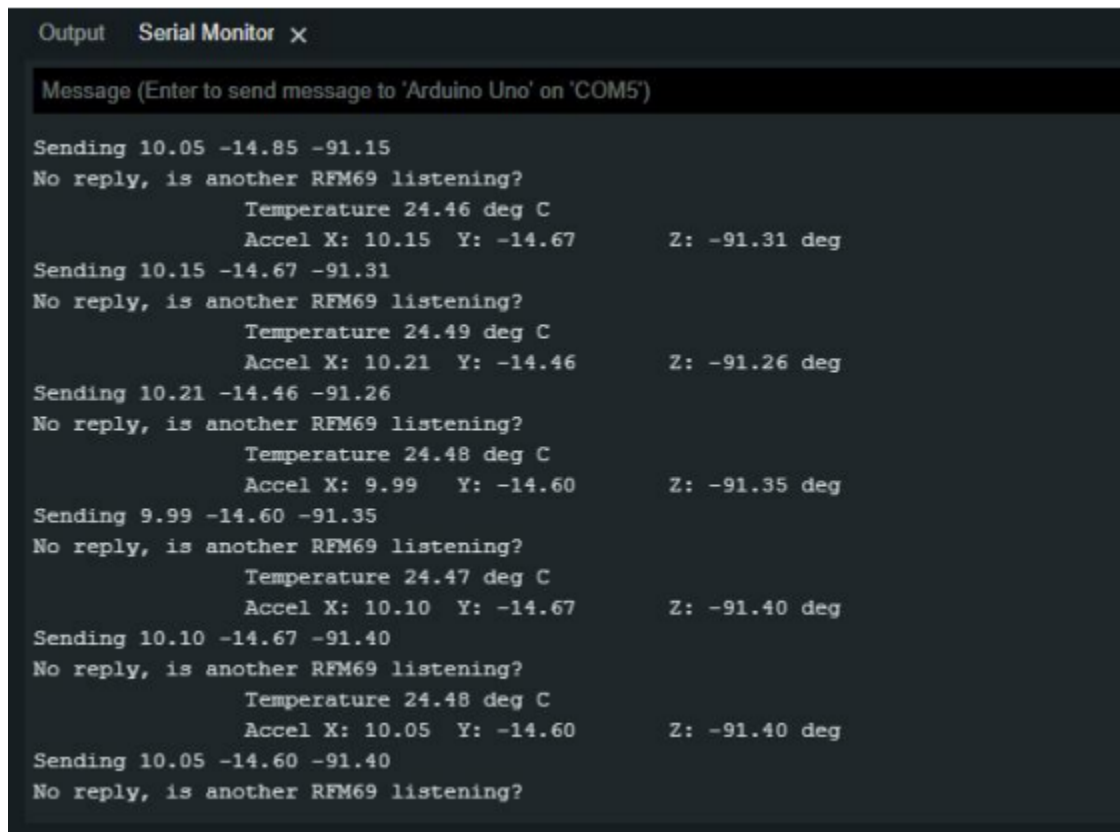
The objective of the device testing performed is to determine the feasibility of wirelessly transferring motion data. The ways pins on the two radio modules are connected with respect to the Arduino Uno processor are identical. The transmitter end has one IMU connected through STEMMA QT cables, while the receiver has none.

### 3.1.2 [IMU Testing]

Using the same setup in 3.1.1, we take the values of the measurements of 3-axis acceleration and rotation from the on-board accelerometer and gyroscope. Through accelerations, the pitch and roll of the aviation principle axes are calculated. Each value is confined to a range from -180 to 180 degrees by coding.

A screenshot of the serial monitor with the sent and received values are shown below:

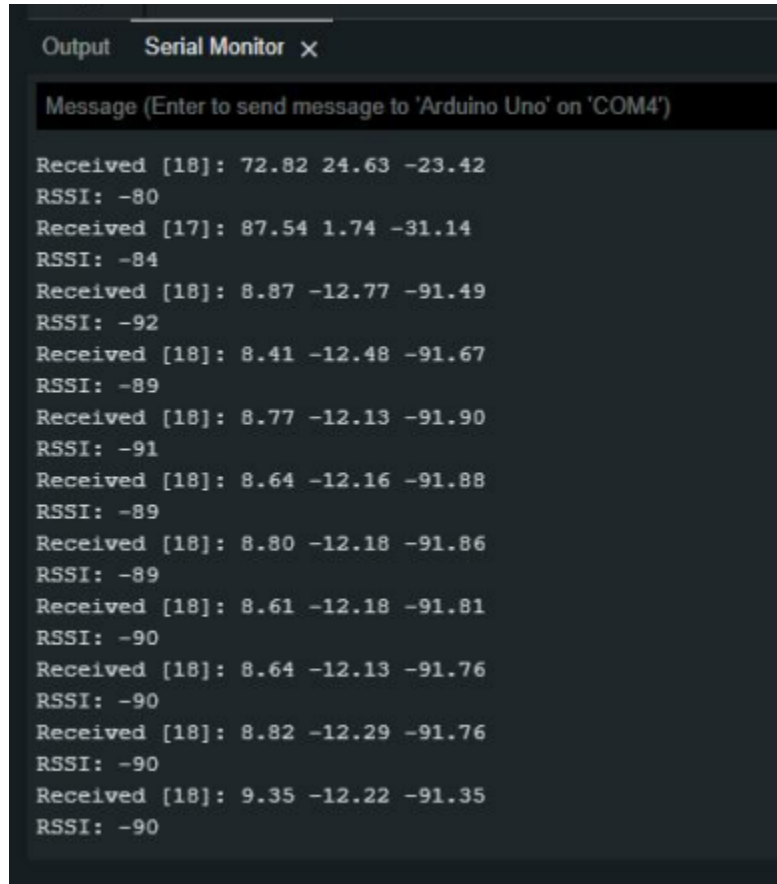
The TX terminal:



```
Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM5')

Sending 10.05 -14.85 -91.15
No reply, is another RFM69 listening?
    Temperature 24.46 deg C
    Accel X: 10.15 Y: -14.67      Z: -91.31 deg
Sending 10.15 -14.67 -91.31
No reply, is another RFM69 listening?
    Temperature 24.49 deg C
    Accel X: 10.21 Y: -14.46      Z: -91.26 deg
Sending 10.21 -14.46 -91.26
No reply, is another RFM69 listening?
    Temperature 24.48 deg C
    Accel X: 9.99 Y: -14.60       Z: -91.35 deg
Sending 9.99 -14.60 -91.35
No reply, is another RFM69 listening?
    Temperature 24.47 deg C
    Accel X: 10.10 Y: -14.67      Z: -91.40 deg
Sending 10.10 -14.67 -91.40
No reply, is another RFM69 listening?
    Temperature 24.48 deg C
    Accel X: 10.05 Y: -14.60      Z: -91.40 deg
Sending 10.05 -14.60 -91.40
No reply, is another RFM69 listening?
```

The RX Terminal:



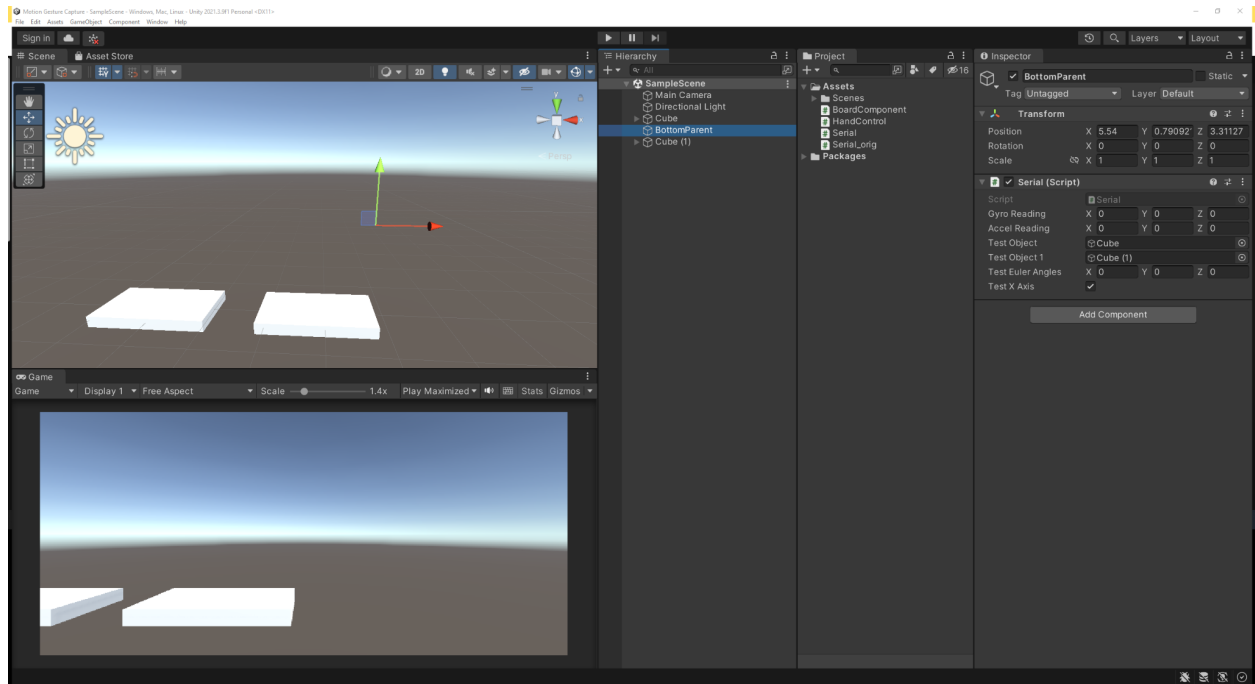
```
Output  Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM4')

Received [18]: 72.82 24.63 -23.42
RSSI: -80
Received [17]: 87.54 1.74 -31.14
RSSI: -84
Received [18]: 8.87 -12.77 -91.49
RSSI: -92
Received [18]: 8.41 -12.48 -91.67
RSSI: -89
Received [18]: 8.77 -12.13 -91.90
RSSI: -91
Received [18]: 8.64 -12.16 -91.88
RSSI: -89
Received [18]: 8.80 -12.18 -91.86
RSSI: -89
Received [18]: 8.61 -12.18 -91.81
RSSI: -90
Received [18]: 8.64 -12.13 -91.76
RSSI: -90
Received [18]: 8.82 -12.29 -91.76
RSSI: -90
Received [18]: 9.35 -12.22 -91.35
RSSI: -90
```

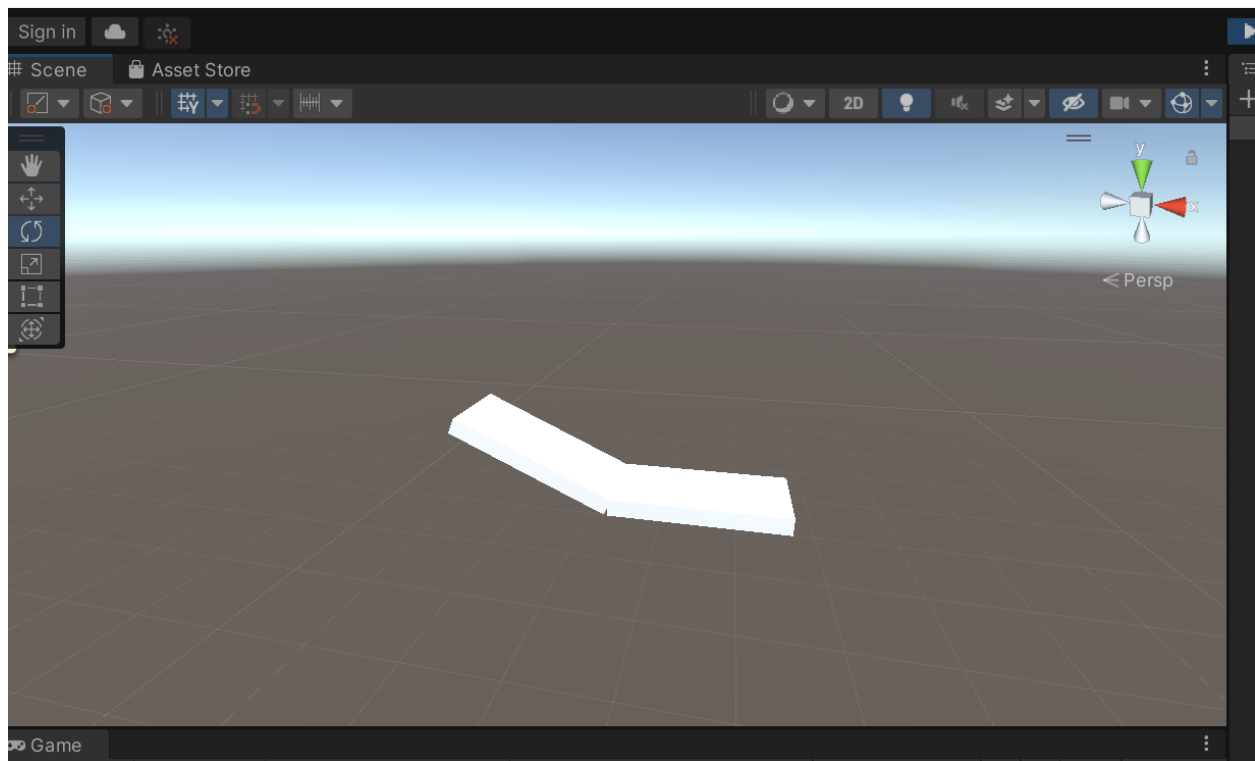
Replies are not needed from the RX terminal as our device only requires a one-way information transmission from the motion capture terminal to the motion reproduction terminal. As shown on the graphs below, the roll, pitch, and gyroscope measurements of the Z axis are successfully transmitted wirelessly.

### 3.2 [Unity 3D Simulation Testing]

The Unity 3D engine serves as the software end of the motion reproduction terminal. It takes input from the serial line and parse the IMU data for recreation of motion. The Unity setup is displayed below:



The two 3D models of boards are representatives of the two IMUs we are using in the final setup. When the simulation is started and data is sent through the serial port, The two boards will join together and bend and rotate just like the IMU, as shown below:



### 3.3 [Mechanical Motor Testing]

To ensure the proper functioning of the robot arm, rigorous testing of the motors is necessary. The motors must be tested to ensure that they spin freely and accurately in accordance with the program. Proper motor functionality is essential for achieving the desired level of precision in the movement of the robot arm. Testing should be performed throughout the development process to identify and address any potential issues as early as possible. It is also important to verify that the motors are compatible with the power supply and control system to prevent damage or malfunction. In addition, proper maintenance and regular inspection of the motors are necessary to ensure their continued functionality and avoid any potential safety hazards.

#### 3.3.1 [Single Motor Test]

Starting with testing a single motor, since we are using the dev board, an Arduino template code can be used, as shown below:

##### Sweep

Sweeps the shaft of a RC servo motor back and forth across 180 degrees.

```
1  #include <Servo.h>
2
3  Servo myservo; // create servo object to control a servo
4  // twelve servo objects can be created on most boards
5
6  int pos = 0;    // variable to store the servo position
7
8  void setup() {
9    myservo.attach(9); // attaches the servo on pin 9 to the
10 }
11
12 void loop() {
13   for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degr
14     // in steps of 1 degree
15     myservo.write(pos);                // tell servo to go to
16     delay(15);                          // waits 15ms for the
17   }
18   for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 de
19     myservo.write(pos);                // tell servo to go to
20     delay(15);                          // waits 15ms for the
21   }
22 }
```

COPY

The motor in the robot arm takes in one position variable and sets the arm to the corresponding position angle within a range of 0 to 180. The motor has three pins: a VCC, a ground, and a signal pin that takes in a PWM signal. To test the functionality of a single motor, an Arduino template code can be used with the dev board. The signal pin is connected to a digital output pin on the dev board and the position variable can be adjusted in the code to test different positions. Proper testing of the motors is crucial to ensure that the robot arm operates accurately and safely. It is important to thoroughly test each motor to ensure that it is functioning properly and in accordance with the program.

## 4. Costs

### 4.1 Parts

Item	Part Number	Quantity	Cost	Total
Microcontroller	ATmega328P	2	\$2.86	\$5.72
<a href="#">RF module</a>	RFM95W	2	\$9.5	\$19
<a href="#">5V Voltage Regulator</a>	LD1117	2	\$0.66	\$1.32
<a href="#">3.3V Voltage Regulator</a>	AP2112K	2	\$0.37	\$0.74
MEMS Sensor	LSM6DSOX	3	\$8.26	\$24.78
Small circuit components (Resistors, capacitors..)	N/A	28	\$21	\$21
USB to Serial adaptor		1	free	free
<a href="#">linear buffer</a>	CD74HC4050P WR	2	\$0.6	\$0.12
<a href="#">antenna</a>	TG.09.0113	2	\$8.15	\$16.3
Total				\$88.98

### 4.2 Labor

Name	Rate	Hours	Total (R*H*2.5)
Joe	\$33	\$210	\$17325
James	\$33	\$210	\$17325
Jason	\$33	\$210	\$17325
			\$51985

**Grand Total: \$52045.47**

## 5. Conclusion

### 5.1 Accomplishments

In the end, all of our high-level requirement lists are met. Not only are the minimum requirements fulfilled flawlessly, we also extend our goals further and complete the mechanical integration that was proposed in the design document. Motions along the roll and pitch axis are reflected within the Unity simulation, while motions along the pitch axis are reflected in the mechanical motor setup. Particularly, the mechanical integration is limited to around 100 degrees spin on both motor, but within the software simulation the motion is unconstrained. The IMUs are attached to a glove with a separate battery station to make it portable. The response time of the motion reproduction terminal is only around 0.2-0.5 second.

### 5.2 Ethical and safety considerations

Developing a project comes with responsibilities that go beyond technical expertise, and includes considering the ethical and safety issues that may arise during development or from the project's potential misuse. It is essential to adhere to the IEEE and ACM codes of ethics, which emphasize the importance of avoiding harm to others and upholding integrity to avoid ethical breaches. Compliance with relevant safety and regulatory standards is also necessary to minimize safety risks. While this project has been approved, it is essential to continue monitoring and evaluating its ethical and safety implications throughout the development process.

Despite the project's overall safety, there are some minor safety concerns that need to be addressed. To ensure the safety of users and developers, a set of lab safety instructions has been established. These guidelines include safe handling and use of small motors, moving robot arms, and small batteries. They also emphasize the importance of turning off power supplies, proper storage of batteries, and avoiding the presence of flammable substances. The safety instructions will help personnel work safely with the project's equipment and minimize any potential safety risks.

In conclusion, ethical and safety considerations are crucial for any project's success, and it is necessary to address these concerns in the project's early stages. By adhering to the IEEE and ACM codes of ethics [1] and relevant safety and regulatory standards, and establishing safety guidelines and training personnel, both developers and users can work safely with the project's equipment and minimize any potential safety risks.

### 5.3 Future work

Since the IMU chosen for this project only has 6 DOF, we are limited to the roll and pitch axes. However, it's possible to upgrade the IMU to 9 DOF with the added magnetometer. Advanced algorithms can then be employed to calculate the YAW axis and complete the principle axes of orientation in free space. With the added axis, we can also upgrade the mechanical system to use an omni-directional motor that is able to reproduce the motion of hands more faithfully than what we currently have, which is fixed to the pitch axis. If done, the device becomes a fully functional independent portable VR-controller-like machine that is able to not only record any motion from any direction, but also reproduce them.

## References

- [1] "IEEE code of Ethics," IEEE. [Online]. Available:  
<https://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: 09-Feb-2023].
- [2] L. Ada, "Adafruit RFM69HCW and RFM9X Lora Packet Radio breakouts," Adafruit Learning System. [Online]. Available:  
<https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/downloads>. [Accessed: 02-May-2023].
- [3] D. Hienzsch, "Build an arduino uno R3 from Scratch Part 1," Rheingold Heavy, 25-Aug-2021. [Online]. Available: <https://rheingoldheavy.com/build-an-arduino-uno-from-scratch-part-1/>. [Accessed: 02-May-2023].
- [4] "Arduino Uno Rev3-02-TH." [Online]. Available:  
[https://www.arduino.cc/en/uploads/Main/Arduino\\_Uno\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf). [Accessed: 02-May-2023].
- [5] Team, T. A. (n.d.). Servo Motor Basics with Arduino. Arduino Documentation. Retrieved May 3, 2023, from <https://docs.arduino.cc/learn/electronics/servo-motors>

## Appendix A Requirement and Verification Table

Requirements	Verifications	Verification status (Y or N)
<ul style="list-style-type: none"> <li>The RF Transmitter Subsystem must transmit measurements from the LSM6DSO32 to the Arduino or SPI/I2C-enabled microcontroller wirelessly through RFM69HCW transceiver with external Antenna connector.</li> </ul>	<ul style="list-style-type: none"> <li>Connect the RF Transmitter Subsystem to the Arduino or SPI/I2C-enabled microcontroller and LSM6DSO32. Verify that the measurements from the LSM6DSO32 are being sent to the Arduino or microcontroller via the SPI or I2C interface.</li> <li>Then, pack the measurements into a 16 to 32 bit code with at least 4 bits for position and 4 bits for rotation for each three-dimensional axis. Record the packed code as read from the microcontroller via serial debugging.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The RF Transmitter Subsystem must transmit data with minimal error or loss.</li> </ul>	<ul style="list-style-type: none"> <li>Ensure the RF Transmitter Subsystem is connected to the receiver subsystem. Record the packed code as read from the receiver subsystem via serial debugging.</li> <li>Move the RF Transmitter Subsystem to a location with greater distance or obstructions between it and the receiver subsystem. Record the packed code as read from the receiver subsystem via serial debugging. Confirm that there is minimal error or loss in the transmitted data.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The RF receiver subsystem must be able to recover information sent through the transmitter wirelessly.</li> </ul>	<ul style="list-style-type: none"> <li>Connect the output of the RF transmitter module to the input of the RF receiver module. Transmit 6 32-bit codes containing information about the three axes of gyro and accelerometer from the transmitter and confirm that the data is received at the output of the receiver. Use an oscilloscope to ensure that the signal is clean and free from interference.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The RF receiver subsystem must be able to connect to an SPI-enabled microcontroller to analyze and unpack the code received from the transmitter.</li> </ul>	<ul style="list-style-type: none"> <li>Connect the output of the RF receiver module to the input of the SPI-enabled microcontroller. Transmit a known code from the transmitter and confirm that the same code is received by the microcontroller. Use a logic analyzer to ensure that the code is correctly unpacked and extracted by the microcontroller.</li> </ul>	Y

Requirements	Verifications	Verification status (Y or N)
<ul style="list-style-type: none"> <li>The microcontroller must be able to prepare the data for respective motion recreation.</li> </ul>	<ul style="list-style-type: none"> <li>Program the microcontroller to analyze the received code and prepare the data for motion recreation. Verify that the prepared data accurately represents the motion information that was originally sent by the transmitter. Use a motion sensor to confirm that the recreated motion matches the original motion sent by the transmitter.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The motion sensing system must measure the linear acceleration and angular velocity in three dimensions</li> </ul>	<ul style="list-style-type: none"> <li>Secure IMUS on a hand and manually move the IMUs in different directions and rotations. Record the data and verify that the linear acceleration and angular velocity in all three dimensions are being measured by the IMUs.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The motion sensing system must use I2C communication protocol for data transmission</li> </ul>	<ul style="list-style-type: none"> <li>Connect the LSM6DSO32 IC to the microcontroller using I2C protocol. Verify that the data is being transmitted between the microcontroller and IMUs using an I2C sniffer or logic analyzer.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The software section must be able to receive data and animate a 3D model.</li> </ul>	<ul style="list-style-type: none"> <li>Connect the microcontroller at the receiver's end to the computer via serial port. Verify that the software section receives the data correctly and that it is able to decode the information.</li> <li>Using those inputs, verify if the 3D model simulates motions in the real world within a reasonable range.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The hardware component must be able to reproduce movements on a scale dependent on the degrees of rotation of MEMS sensors.</li> </ul>	<ul style="list-style-type: none"> <li>Vary the degrees of rotation of the MEMS sensors and record the resulting movements of the hardware component. Verify that the hardware component is able to reproduce movements on a scale dependent on the degrees of rotation of the MEMS sensors.</li> </ul>	Y
<ul style="list-style-type: none"> <li>The hardware component must be able to utilize pulse-controlled servo motors and 3D printed connection parts.</li> </ul>	<ul style="list-style-type: none"> <li>Connect the pulse-controlled continuous rotation servo motors to the hardware component. Verify that the hardware component is able to rotate freely in their own dimensions and reach limitations in a safe manner.</li> <li>Design and print 3D structures and connect them to the hardware component to connect the rest of the electronic parts.</li> </ul>	Y