March 12, 2023

Dr. Michael Hegedus
Simon Fraser University
School of Engineering Science
Burnaby, British Columbia
V5A 1S6

Re: ENSC 405W Design Specification for Croma Tech's S-ROM

Dear Dr. Hegedus,

The attached document defines the design specifications for S-ROM, a wearable device with a data presentation application used for comprehensively evaluating the range of motion (RoM) of a shoulder.

The design specification document will outline the system, hardware, mechanical, firmware, and software specifications for S-ROM. Specifications will be included for the Proof-of-Concept prototype and engineering prototype. A design alternatives appendix will be included along with our most recent test plan for ensuring the design meets the requirements specifications.

Our team – Ansley Ang, David Bechert, Diego Flores, Luka Cuk, Ritesh Nandakumar, and I – are senior Simon Fraser University engineering students. With our diverse backgrounds in physics, electronics, systems, and computer engineering, we are confident in our ability to complete this project.

Thank you for taking the time to review our design specification. For inquiries, please contact our Chief Communications Officer, Ansley Ang, by phone (778-987-4926) or by email (ansleya@sfu.ca).

Sincerely,

[illegible]

Elias Bircher
Chief Executive Officer
Croma Tech

Enclosed: Croma Tech Design Specifications document for S-ROM

Design Specification for S-ROM

March 12th, 2023

CROMA TECH

Elias Bircher	Chief Executive Officer
David Bechert	Chief Technology Officer
Ansley Ang	Chief Communications Officer
Luka Cuk	Chief Operations Officer
Diego Flores	Chief Financial Officer
Ritesh Nandakumar	Chief Product Officer

Abstract

The design specifications document will describe the design of the S-ROM device from the system, electrical, software, and hardware perspectives. S-ROM is a wearable device that evaluates the shoulder range of motion for a variety of shoulder injury patients attending physiotherapy. The device can be used in-clinic to provide the physiotherapist with an immediate quantitative assessment of the patients range of motion and or can be used at home by the patient to amplify the dataset available to the physiotherapist when creating custom recovery plans. The design descriptions will include relevant justification based on our extensive conversations with licensed physiotherapists, our design requirements, external research, and analytical and empirical usability testing.

CROMA TECH

Table of Contents

CROMA TECH

List of Figures

List of Tables

Glossary

Term	Definition
Ack	Acknowledge
BOM	Bill of Materials
Calibration	A configuration procedure that ensures an instrument provides results within an acceptable range of values
Directions For Use	“Full information as to the procedures recommended for achieving the optimum performance of the device, and includes cautions, warnings, contra-indications and possible adverse effects” [1]
DMP	Digital Motion Processor
EKF	Extended Kalman Filter
FW	Firmware
Goniometer	An instrument that measures the range of motion at a joint using angles
HW	Hardware, in relation to the electronic components of the design
I2C	Inter-integrated Circuit Protocol
LDO	Low Dropout Regulator
MCU	Microcontroller Unit
Nak	Not Acknowledged
Overcompensation	Tightening and straining of non-injured muscles caused by changes in normal joint mechanics
PC	Personal Computer
PoC	Proof of Concept
PT	Physiotherapist
RoM	Range of Motion
S-ROM	Shoulder - Range of Motion (Our Product Name)
SW	Software
Universal Serial Bus (USB)	Protocols that outline communication between a host controller, such as a PC, and a peripheral device

Version History

Version	Date	Comment
1.0	03-17-23	Initial Release

Approvals

This document has been reviewed and approved by Croma Tech and contains the most recent design specifications for S-ROM as of 03-17-23.

1) Introduction

Physiotherapists (PT) play a key role in the long-term health and quality of life for shoulder injury patients. During a typical appointment, a universal goniometer is used to measure rotation angles for various RoM exercises. The goniometer has poor repeatability and accuracy, and only provides measurements when the patient is in the clinic – about once or twice a week on average [2] [3]. To improve RoM assessments and subsequently help PTs create a more tailored recovery plan, Croma Tech is developing the S-ROM, a wearable device that accurately measures shoulder RoM and quantifies progress through at-home and in-clinic measurements.

Our device will resemble commercially available braces with adjustable Velcro straps. Once the device is equipped, it will connect to a desktop application that allows the user to select an exercise, begin the exercise, and view results of the measurement afterwards. Results will include plots of the angle over time, as well as indicators for overcompensation and incorrect exercise detection. The system overview is shown in Figure 1. The overcompensation detection feature is expected to be our most significant challenge as it may require fundamental alterations to the mechanical design to incorporate. Additionally, because a key feature of our device is to allow patients to use the device at home, it is important to ensure that the device is easy to put on and remove. It will be challenging to create a device that can repeatedly be put on by various users while still providing the necessary accuracy.

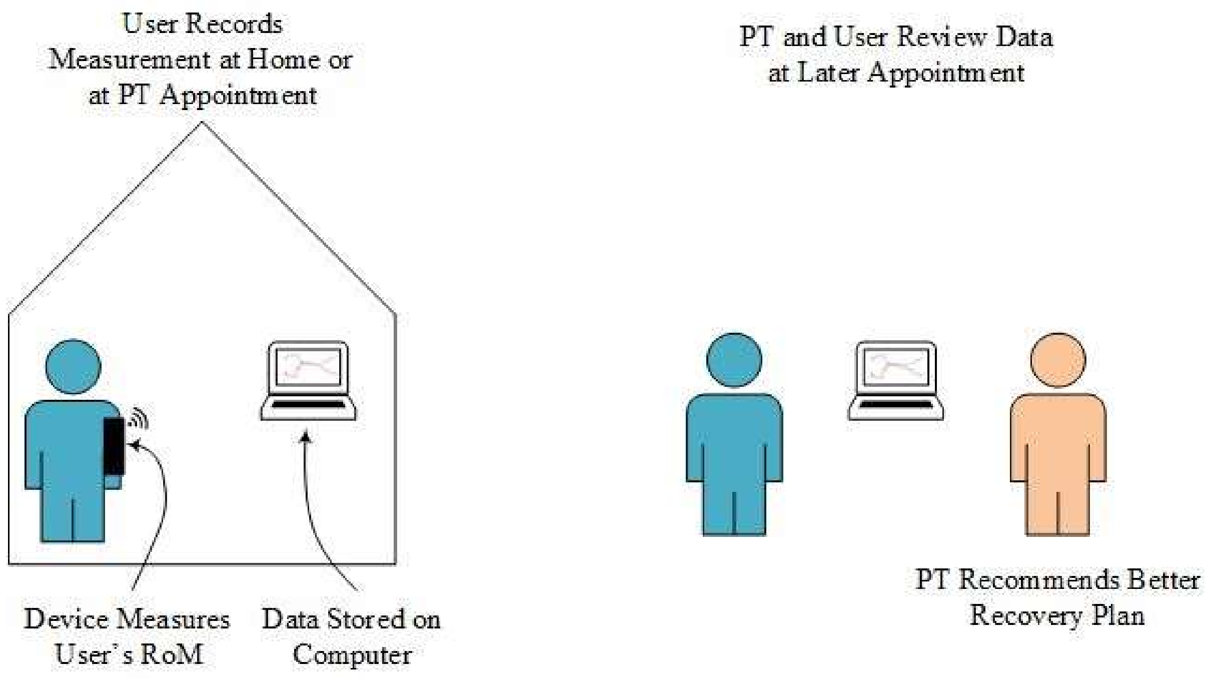

User Records Measurement at Home or at PT Appointment
PT and User Review Data at Later Appointment
Device Measures User's RoM
Data Stored on Computer
PT Recommends Better Recovery Plan

Figure 1: S-ROM Use Case Diagram

2) Design Specifications

The following sections describe the design of the S-ROM. All design specifications are based on requirements as prescribed by the "Requirements Specification" document [4]. By designing our system based on our requirements we can guarantee that our product will be functional for our end user.

2.1) Design Specifications Format

Our specifications will follow the following format:

D<section #>.<subsection #>.<specification #>.<project stage>

The following table, Table 1: Project Stage Definition, outlines the meanings of the different project stages in requirements.

Table 1: Project Stage Definition

Project Stage	Meaning
A	Proof of Concept (PoC)
B	Engineering Prototype
C	Production Version

2.2) PoC Bill of Materials

The following table, Table 2, provides the Bill of Materials (BOM) for the PoC prototype of S-ROM. Our aim is to provide a low-cost measurement device allowing physiotherapists to purchase multiple S-ROM devices and lend them out to multiple patients at a time. By researching low-cost components and ensuring that we are designing with cost in mind we can ensure our product falls within our requirement Req 2.6.1.B.

Table 2: PoC Bill of Materials

Item	Unit Cost (CAD)	Quantity	Extended Cost ($ CAD)
Arduino Mega 2560	$66.27	1	$66.27
ICM-20948 IMU Evaluation Board	$26.65	2	$53.30
Qwiic 4-pin JST Cables	$1.31	2	$2.33
Elbow Brace	$21.99	1	$21.99
Hook and Loop Straps	$2.33	2	$4.66
Electronics Plastic Enclosure	$0.80	3	$2.40
	Total Cost		$150.95

2.3) System Overview

At a high level our system has three distinct aspects: the user, the GUI and the HW. These elements must communicate information to each other for us to collect measurement data for shoulder abduction, external rotation, and flexion exercises (R2.1.1.A - R2.1.3.A). A high-level diagram of our system to be implemented for the PoC prototype is shown in Figure 2, where the distinct elements of the system are outlined. To characterize these three aspects and their relations, our system can be broken down into hardware, mechanical, firmware and software components. Each of these components will be described in further detail through our design specifications. The S-ROM hardware system is shown in Figure 3.

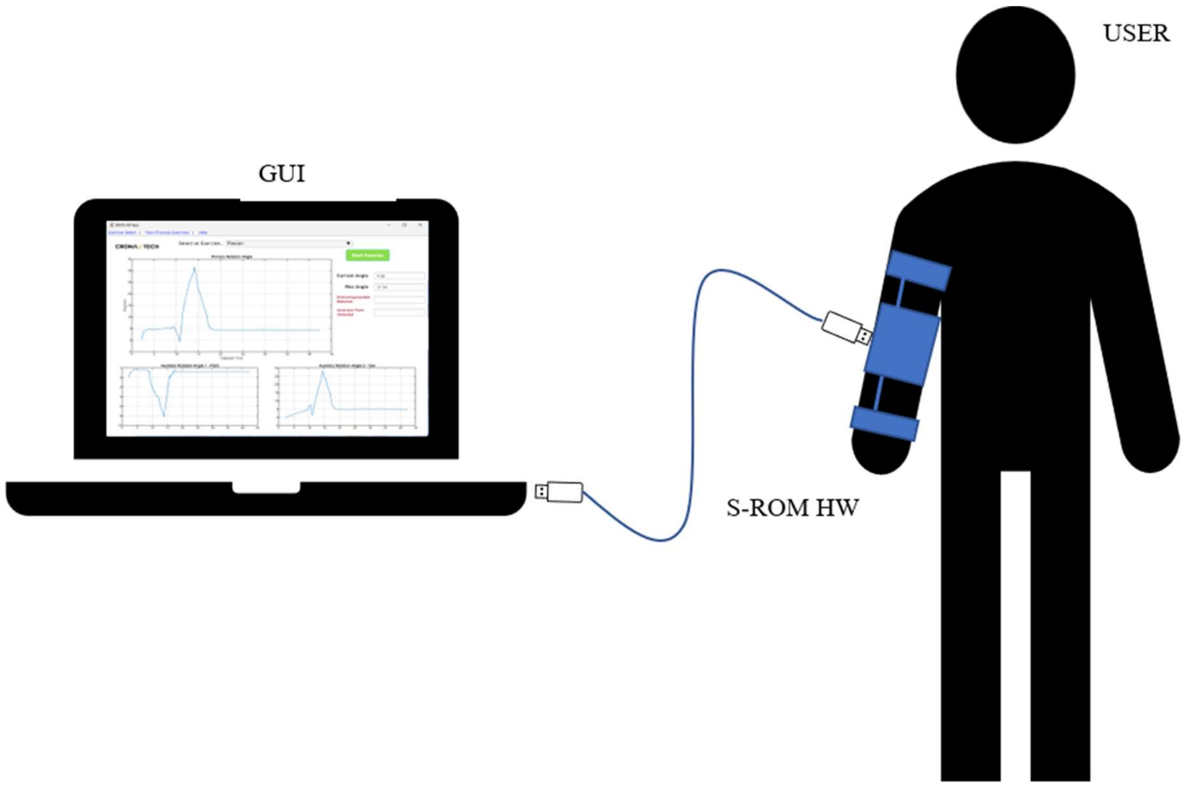


Figure 2: High Level System Diagram

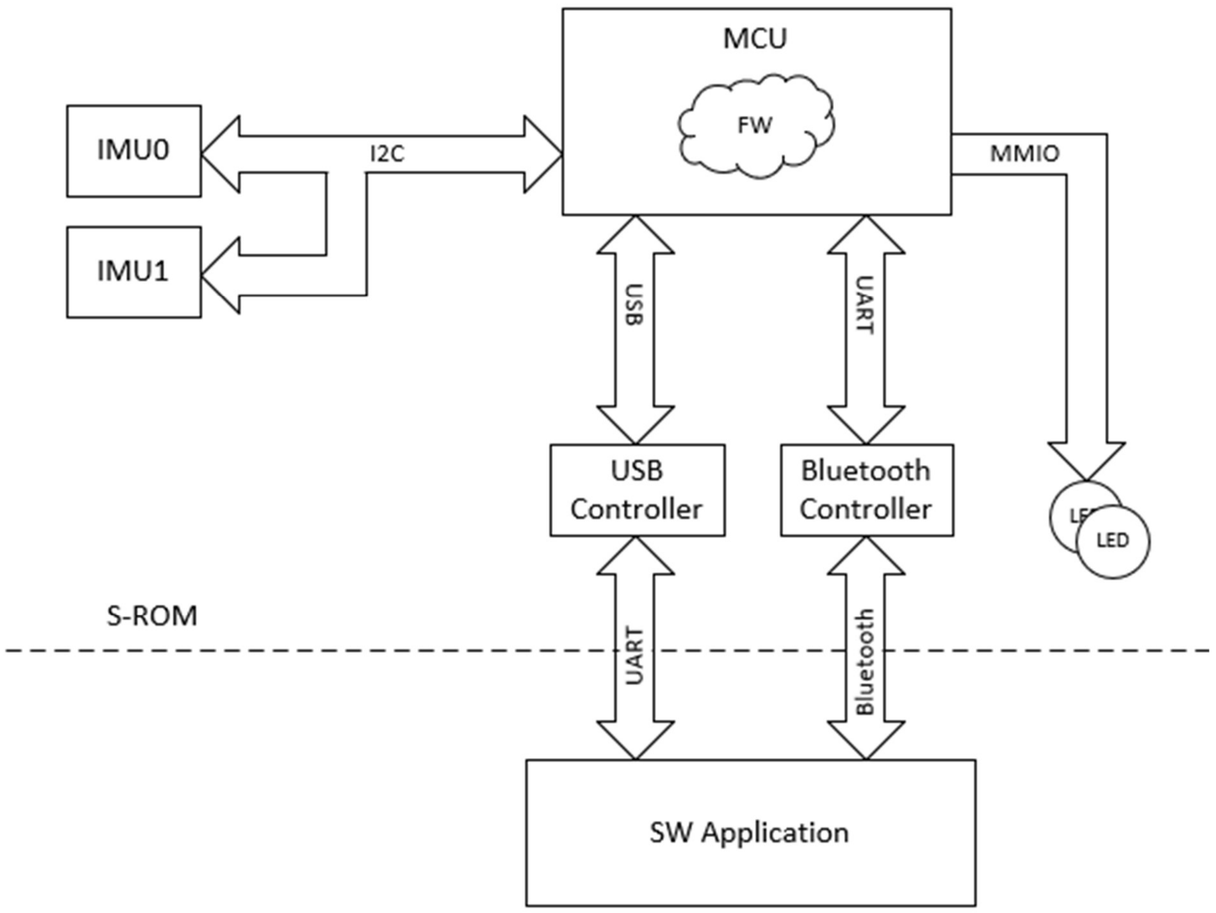


Figure 3: S-ROM System Hardware

The wearable device is designed to allow flexibility for different users. It consists of three adjustable portions: the forearm strap, elbow brace, and shoulder strap, shown in Figure 4. The two straps are adjustable through a looped fastener system and house the 9-axis IMUs within each of the black plastic cases. The IMU positioned at the wrist facilitates accurate tracking of the rotation angles for each exercise, while the IMU at the shoulder tracks overcompensation and incorrect exercise performance and acts a redundant measurement sensor. The middle elbow brace is composed of a compression fabric to ensure the device is comfortable to wear yet is not affected by loose positioning. The plastic enclosures at the elbow contain the main controller unit as well as the power unit. For the PoC prototype, the device will be controlled by an Arduino Mega 2560 powered directly using a wired 5V USB connection. The engineering prototype will provide users with a wireless option to enable convenient operation of the device.

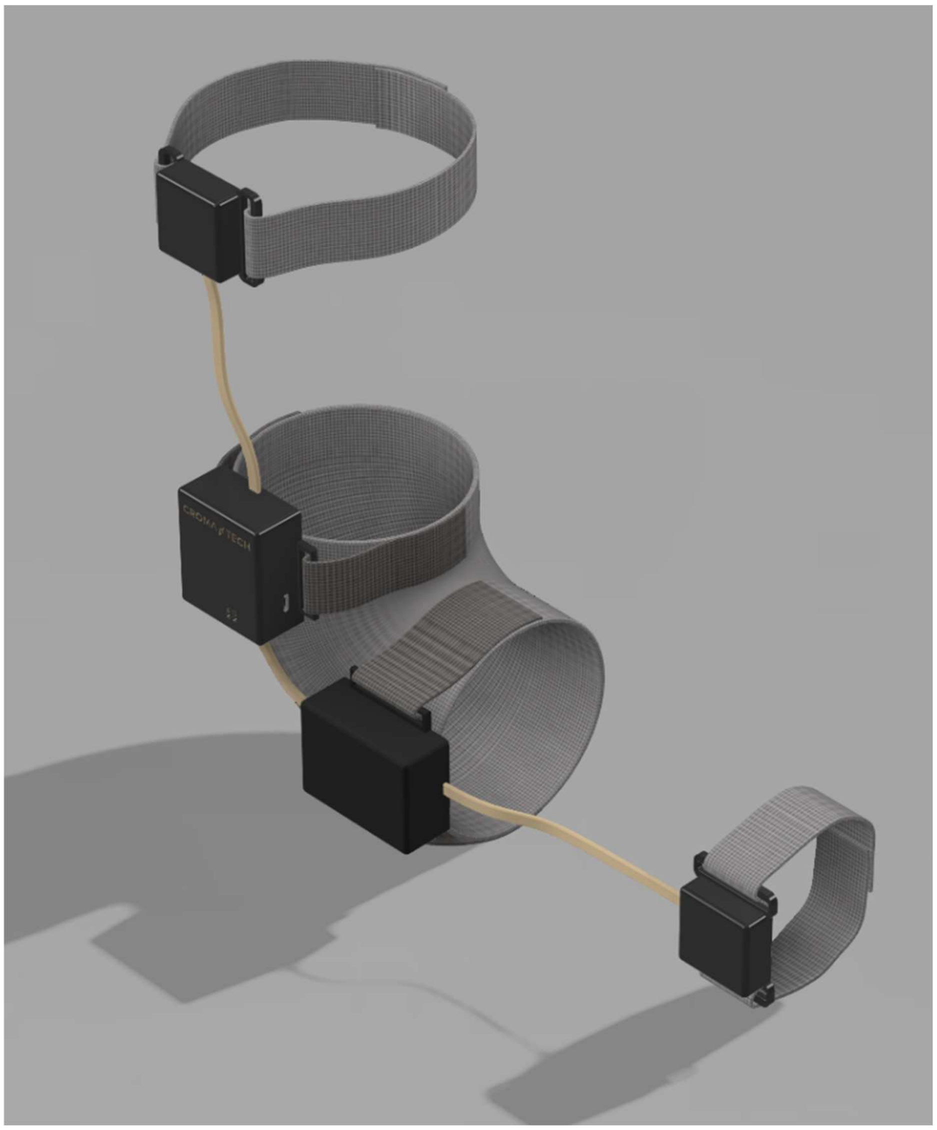


Figure 4: S-ROM Mechanical Sleeve Design

The users of S-ROM are not required to configure the wearable device beyond putting it on. The remainder of the effort is concentrated in the software application provided with the product. For the PoC prototype, this will be done using the MATLAB App Designer tool. For the engineering prototype and future releases, the GUI will be developed using a more sophisticated language. Figure 5 shows an initial rendering of the engineering prototype GUI. The GUI will allow users to select an exercise to make a measurement, start and restart the measurement, and view previous measurement data to assess trends in the patient's recovery.

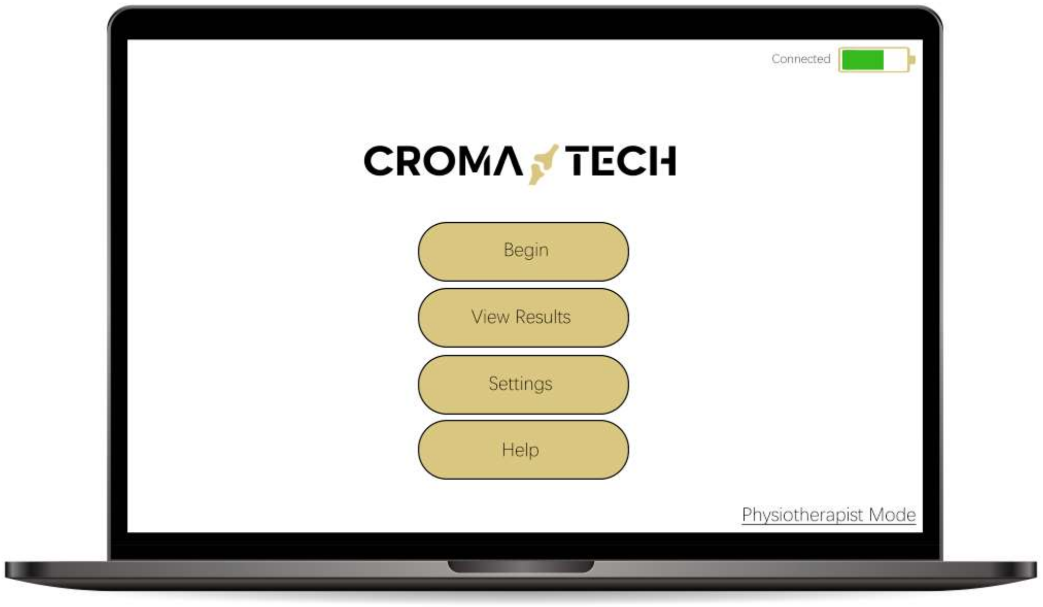


Figure 5: S-ROM GUI Main Menu

The following synchronization diagram, Figure 6: System ULM Synchronization Diagram, shows how the HW and GUI interact when a user wants to connect the device or start an exercise. This diagram assumes the user has turned on the GUI and S-ROM HW.

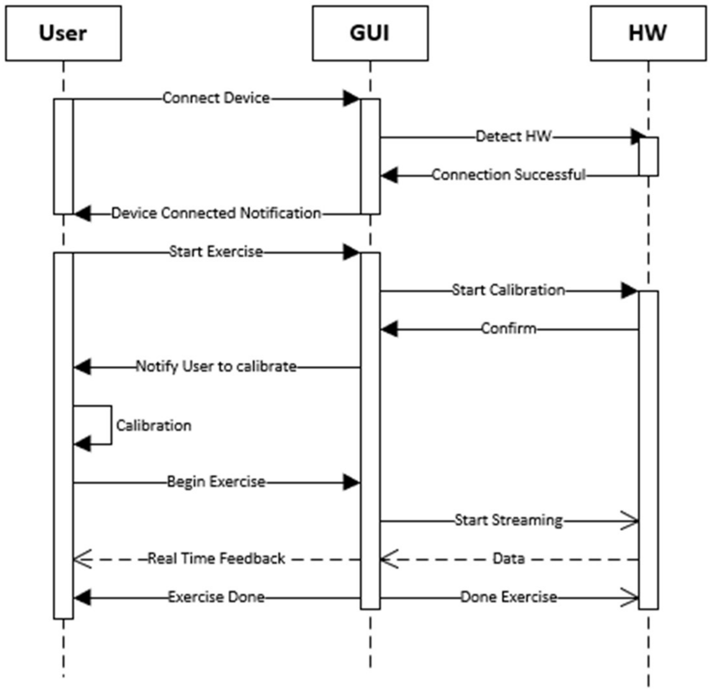


Figure 6: System ULM Synchronization Diagram

2.4) Hardware Specifications

The hardware specifications, outlined in Table 3, correspond to the electronic design of our wearable device. Our PoC prototype will feature two 9-axis IMUs with their breakout boards, an Arduino Mega 2560, and a USB connection to interface with a computer. The IMUs output sensor position values in quaternions to the Arduino over I2C lines; these positions are computed by an onboard DMP using the sensor's accelerometer (m/s^2), gyroscope (dps), and magnetometer (µT) readings. After some FW processing, the Arduino will then communicate with the PC through USB. The power will be supplied by the USB connection (5V, 500mA) to power the Arduino, while the breakout boards step down the voltage to 1.8V to power the IMUs.

For the engineering prototype, there will be additional features added to the design to meet our requirements. Figure 7 displays a high-level block diagram of our important components. The prototype will feature a main control board that can interface with the PC application over Bluetooth or serial port, control the power and status LEDs to indicate the state of the device, and turn the device on/off and pair to host computer using a power button (Req 2.3.14B). IMU sensors will be placed on their own peripheral boards on the bicep and wrist locations the patient's arm.

Table 3: Hardware Design Specifications

Design Spec ID	Description	Related Req ID(s)
D2.4.1.A	The HW will use 9-axis IMUs to capture the shoulder RoM.	Req 2.1.1.A – Req 2.1.5.A Req 2.1.9.A
D2.4.2.A	USB will power the device when the cable is connected	Req 2.2.2.A
D2.4.3.A	No individual component exceeds 30 degrees in operation	Req 2.5.1.A
D2.4.4.A	The state of the device will be indicated by green-red LEDs	Req 2.3.14.B
D2.4.5.B	The HW button will power on/off the device and pair to the host computer over Bluetooth LE.	Req 2.2.4.B Req 2.3.14.B
D2.4.6.B	A battery unit supplies power to the device when operating wirelessly.	Req 2.2.4.B Req 2.2.5.B
D2.4.7.B	The battery will be rechargeable through a USB connection	Req 2.4.6.C
D2.4.8.B	The battery will be rated 3.7V at 1000 mAh	Req 2.2.4.B
D2.4.9.B	The sensors will be assembled on a separate PCB than the other components.	Req 2.6.1.B

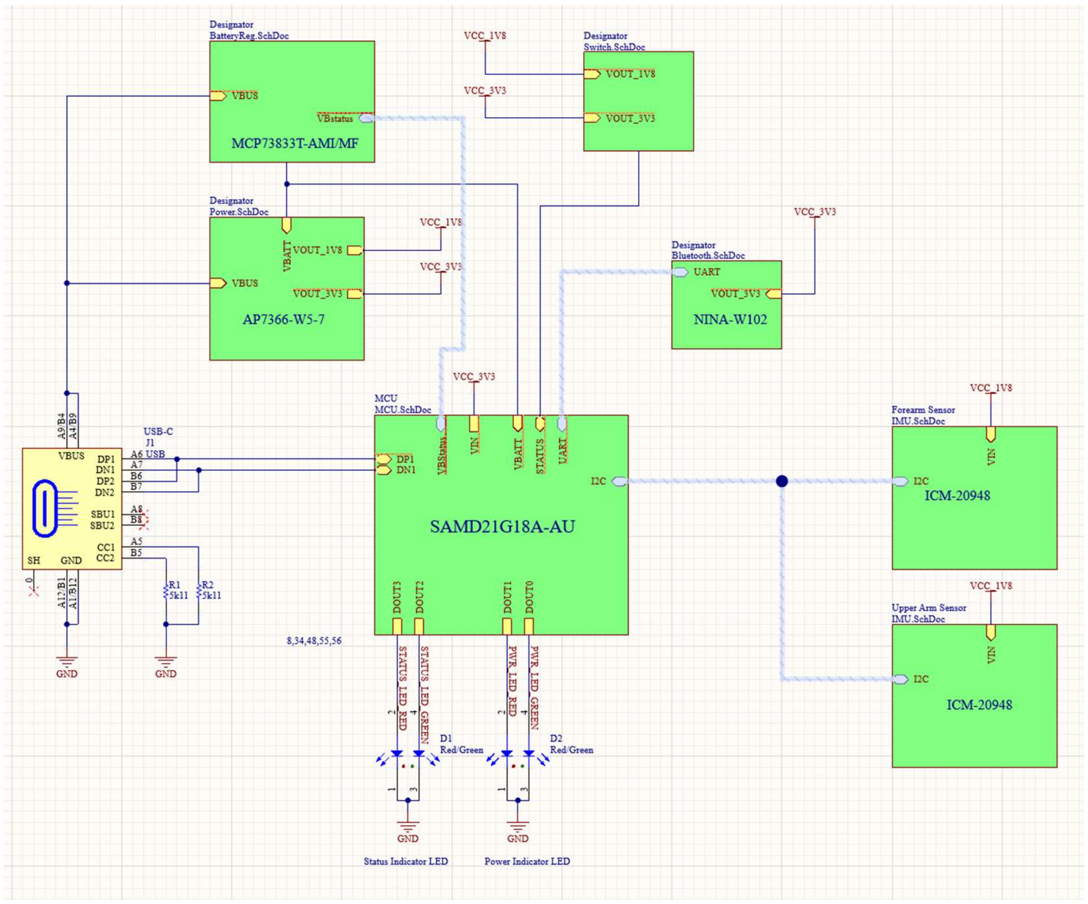


Figure 7: Circuit Schematic for the Engineering Prototype

2.4.1) MCU Specifications

For our PoC, our device is controlled by an Arduino Mega 2560 which uses the ATMEGA2560 MCU shown in Figure 8. Choosing a product from the Arduino family allows us to focus solely on firmware development since the Arduino IDE provides the necessary overhead for our product. The Mega has 256KB of flash memory and supports an additional I2C bus over similar models – both critical features for our prototype design. Currently, our FW uses approximately 35KB of storage space for the DMP libraries which would be too large for an Arduino Uno (32KB flash storage). Furthermore, since the ICM-20948 IMUs only support two unique I2C addresses, having an additional I2C bus is a proactive design approach if more sensors are required.

For our engineering prototype, we will use an ATSAMD21G18A-AU MCU from Microchip Technologies, also shown in Figure 8. There are a few reasons why we plan to move away from the ATMEGA2560. First, the ATMEGA2560 is currently out of stock, whereas we were able to source the ATSAM MCUs at $6 each which is cheaper than some lower-end alternatives. This MCU is also compatible with the Arduino bootloader, meaning we can transfer over a significant amount of our current FW. Also, it operates at 3.3V and features multiple low power modes, which reduces our battery power needs. A list of the ATSAMD21G18A-AU specifications are outlined in Table 4.

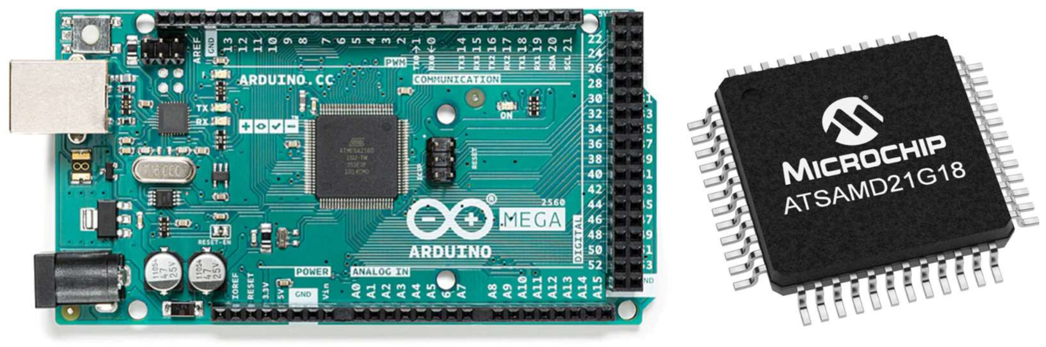


Figure 8: Arduino Mega 2560 (left) vs. ATSAMD21G18A-AU chip (right) [5] [6].

Table 4: ATSAMD21G18A-AU Specifications

Feature	Specification
CPU	32-bit ARM Cortex M0+
Max Speed	48 MHz
Flash Memory	256KB
Communication Interfaces	USB, USART, SPI, I2C
I/O Pins	38
Nominal Operating Voltage	3.3V

This MCU includes a 32-bit ARM Cortex M0+ CPU running at up to 48MHz along with 256KB of flash memory. There are 38 configurable I/O pins, with the ones being used defined in Table 5. Additionally, a USB interface is included onboard, allowing us to simplify our design by omitting a USB to UART converter. The USB interface ensures firmware can be loaded onto the device and data can be streamed back to the host computer. Also, the MCU has six serial ports that can be configured to operate as either a USART, I2C or SPI interface. While these specifications are more than what we need for our system, we want to leave room for scalability to ensure smooth integration if we decide to include any additional sensors in the future (additional IMUs, EMG circuit, etc.).

Table 5: ATSAMD21G18A-AU Pin Definitions

Pin number	Function	Description
VDDIN, VDDIO, VDDCORE, VDDANA	Power Pins	Power pins to turn on MCU core and other peripherals
PA25	USB_P	USB positive
PA24	USB_N	USB negative
PA08	I2C_SDA	I2C Data pin
PA09	I2C_SCL	I2C Clock pin
PA22	UART_TX	UART transmit pin

Pin number	Function	Description
PA23	UART_RX	UART receive pin
PA30	PWR LED RED	Red Power pin LED
PA31	PWR_LED_GREEN	Green Power pin LED
PA10	STATUS_LED_RED	Red status pin LED
PA11	STATUS_LED_GREEN	Green status pin LED
PA12	VBAT	Detects battery charge
PA13	STATUS	Detects Pairing status and turns device on and off
PA14	VBsatus (STAT1)	Charging IC status indicator 1
PA15	VBstatus (STAT2)	Charging IC status indicator 2
PA16	VBstatus (PG)	"Power good" signal

Furthermore, this MCU will communicate with the two IMUs over one of the I2C buses. For communication with the host computer, the device will either use a USB connection or Bluetooth connection. The USB connection will have priority over the Bluetooth connection in efforts to deplete the batteries as little as possible.

2.4.2) Hardware Indicators

The status and power LEDs, shown in Figure 9, will be controlled by the MCU I/O pins based on the states of the system outlined in Table 6. The red, green, and yellow colours will be made using bicolour red/green LEDs. If both green and red LEDs are turned on simultaneously, we can achieve a yellow-coloured light by light theory.

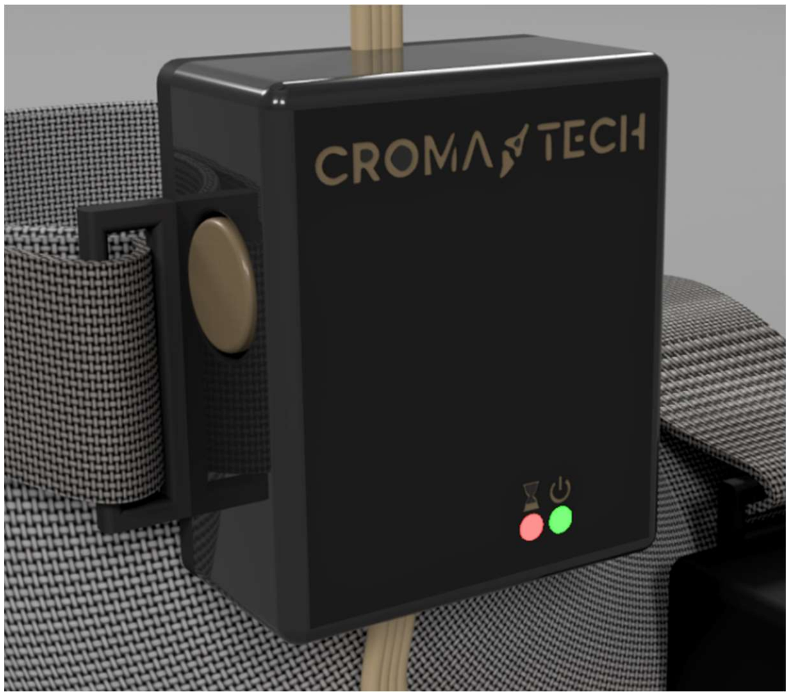


Figure 9: S-ROM LED Indicators

Table 6: Device LED Descriptions

Purpose	Symbol	LED Colour
Power/Pairing	⏻	Green: Device On, High Battery Level Yellow: Device On, Pairing Mode Red: Device On, Critical Battery Level **Blinking Indicates the Device is Disconnected from the Computer*
Status	⌛	Green: Device Ready to Record Measurements, will Blink when Recording Starts Yellow: Calibration Mode Red: Device Not Ready for Measurements

2.4.3) IMU Specifications

The sensors used for angular measurements are the ICM-20948 9-axis inertial measurement units from TDK InvenSense, shown in Figure 10. The unit itself contains a 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, and a Digital Motion Processor. This DMP requires memory from the MCU to store the sensor fusion algorithms; however, most of the computations are done on-chip, removing much of the load from the MCU and allowing for better power performance. The 9-axis IMU's were chosen because they allow us to perform the measurement without any external setups (i.e., for a motion detection system) and are relatively low cost at $15 per chip [7]. More details regarding the sensor selection can be found in Appendix A.

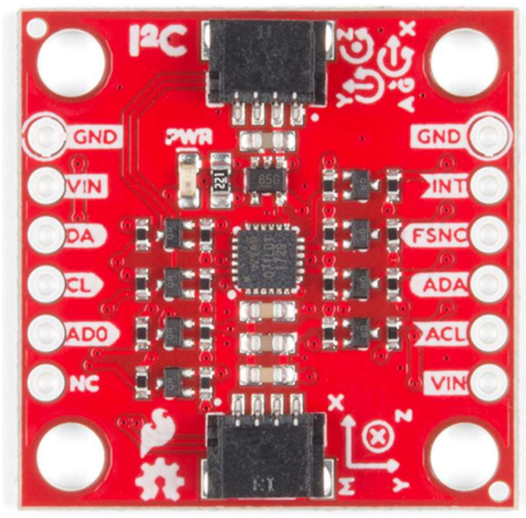


Figure 10: 9-axis IMU Breakout Board - ICM-20948 [7]

An additional advantage of the IMU is the capability of interfacing over both I2C and SPI. This allows us to scale up our design in the future in terms of additional sensors on the same I2C bus or switching to SPI and reaping the benefits of increased communication rates.

While the ICM-20948 datasheet does not supply specifications for the accuracy of the sensor, our preliminary analytical testing has confirmed that its performance will be sufficient in meeting requirement Req 2.1.4.A. The test rig shown in Figure 11 is what we used for accuracy testing, which allowed us to mount the sensor and move it in two axes. This allowed us to perform exercises on the sensor "arm" consistently across trials with less ambiguity in where to measure.

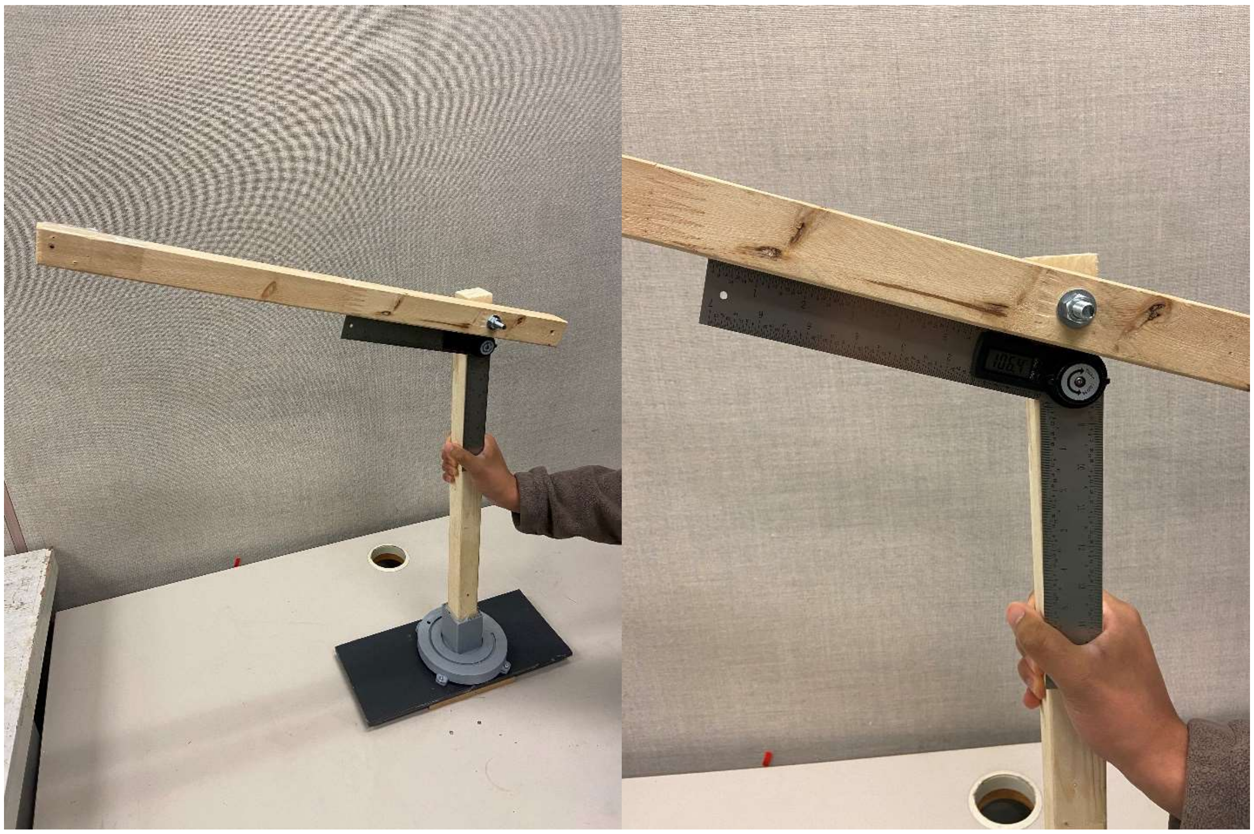


Figure 11: Rig for IMU Angle Testing

2.4.4) Bluetooth Specification

For our PoC prototype, we will not implement a Bluetooth connectivity feature as it is not a crucial part of our product. However, SROM's optional Bluetooth connectivity feature for the engineering prototype will be handled by the NINA-W102 wireless module from u-blox shown in Figure 12. This module integrates an MCU, multi-radio module and integrated antenna all in one. The USB connection will take priority over the Bluetooth connection, meaning if the USB cable is plugged in and the devices is connected to the host computer via Bluetooth, the device will automatically disconnect Bluetooth and begin communication with the host computer over USB. This simply ensures that the most robust communication type is being used and limits our power consumption. The Bluetooth module has a standard 4-wire UART interface used to communicate with the external MCU.

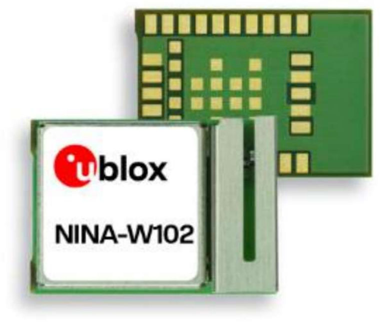


Figure 12: NINA-W102 Wireless Module [8]

One concern we had with Bluetooth was if it had the bandwidth for our data. This module's datasheet specifies a maximum transmission rate of 1Mbps for Bluetooth LE. Based on the calculations for the sensors, this is our required throughput between the device and the GUI:

$$Required\ Throughput = \frac{n_{sensors} * (2 * 3 * b_{sensor})}{t_{sample}}$$

$$Required\ Throughput = \frac{2 * (2 * 3 * 32)}{10ms} = 38.4kbps$$

Where 3*b_{sensor} is the number of output bits for the sensor's quaternion outputs for each axis, $n_{sensors}$ is the number of sensors, and we have twice the number of bits for packet header overhead. Assuming we sample data at t_{sample} = 10ms for two sensors with 32-bit outputs and 32-bit messages, our required throughput should be 38,4kbps, which is well within Bluetooth LE's protocol. Furthermore, even if we had five sensors and wanted the sensors to output an additional 32 bits of data, the throughput would only be 192kbps. Although further testing will be required for confirmation, our current Bluetooth solution should be well suited for future design additions.

2.4.5) Power and Battery Specifications

The S-ROM device will be powered through the USB connection for the PoC prototype, but we will include a rechargeable battery for the engineering prototype to support wireless operation. The battery will automatically begin charging when the USB cable is connected to the device. The battery included in the S-ROM device will be a 3.7V rechargeable LiPo battery rated for 1000 mAh, as shown in Figure 13. With this battery, the optimal battery life can be calculated as follows:

$$t_{battery} = \frac{V_{battery} * Battery\ Capacity}{P_{device,max}}$$

$$t_{battery} = \frac{3.7V * 1000mAh}{0.56W} = 6.65\ hours$$

A detailed breakdown of the device power consumption is shown in Table 7 with the most significant components and their worst-case scenarios; the numbers were either found in their datasheets or through physical measurements. These calculations demonstrate that our power requirements are well within the specifications of Req 2.5.1.A. The expected battery life of 6.65 hours is also much higher than the 1.5 hours from requirement Req 2.2.5.B. The current power/battery life margin, combined with the fact that most of the power dissipated is from the BLE module, confirms we can add other functionality to the device for future iterations and remain within the target requirements. Furthermore, the total current consumption is much lower than a USB port supplying 5V at 500mA, implying that a cabled connection is viable for both the PoC and engineering prototype.

Figure 13: Lithium Polymer Battery - 3.7V, 1000mAh [9]

Table 7: Estimated Power Consumption and Battery Life of the S-ROM

Component	Voltage Dropped (V)	Current Consumed (mA)	Power Dissipated (W)
MCU	3.3	3.78 (CPU, I2C, UART)	0.012
LDO (3V3)	0.4	134 (MCU & Bluetooth Module)	0.054
LDO (1V8)	1.9	16.5 (IMUs)	0.031
IMUs	1.8	16.5 (2 IMUs outputting quaternions)	0.030
BLE Unit	3.3	130	0.429
		Total Power Dissipated	0.560

2.4.5.1) Battery Management System

To charge our battery, we will use a MCP73833T-AMI/MF LiPo battery charging IC from Microchip Technologies [10]. This chip has three output pins that indicate the charging state and outputs the present voltage charge on the battery. These parameters will be fed to the MCU to eventually provide a battery percentage indicator for requirement Req 2.4.2.B.

2.4.5.2)Power Supply Regulation

The main S-ROM PCB requires a 3.3V rail to supply power to the MCU and Bluetooth module. On the other hand, a 1.8V rail will also be needed to power the IMUs. These are not issues for the PoC prototype, since the IMU breakout boards have an on-chip voltage converter and the MCU requires a 5V input provided by the USB connection directly. For the engineering prototype, we will use multiple AP7366-W5-7 LDO voltage regulators that can be adjusted at the output and provide up to 600mA of current [11]. We chose to use LDO regulators over switching regulators since they are easier to implement, small physical footprint, and are usually cheaper.

2.5) Mechanical Specifications

The mechanical design specifications describe the physical design of the S-ROM sleeve in terms of materials, dimensions, and other factors. Table 8: Mechanical Design Specifications, outlines the mechanical design specs of the S-ROM device. Figure 14: Mechanical Design Overview, shows a skeleton diagram of the mechanical design components of the S-ROM sleeve.

Table 8: Mechanical Design Specifications

Design Spec ID	Description	Related Req ID(s)
D2.5.1.A	The S-ROM sleeve consists of a compressive elbow brace.	Req 2.1.10.B Req 2.1.13.B Req 2.5.2.A
D2.5.2.A	The S-ROM sleeve consists of adjustable Velcro straps at the upper bicep and wrist locations.	Req 2.1.10.B Req 2.1.13.B Req 2.5.2.A
D2.5.3.A	The electronics will be housed in PLA plastic enclosures at the outer end of the sleeve (away from body).	Req 2.2.8.C Req 2.4.1.A Req 2.5.3.B - Req 2.5.6.B Req 2.5.8.B Req 2.5.9.B
D2.5.4.A	The device can be folded to fit within a space of 16x14x4cm	Req 2.1.12.B
D2.5.5A	The device will weigh 250g at first use, excluding packaging	Req 2.2.9.C
D2.5.6.B	The main HW unit features a push button.	Req 2.3.14.B
D2.5.7.B	Status LEDs are visible to users	Req 2.3.12.B
D2.5.8.B	Connecting wires are protected with heat shrink	Req 2.5.3.B
D2.5.9.B	The design will use cost effective parts.	Req 2.6.1.B

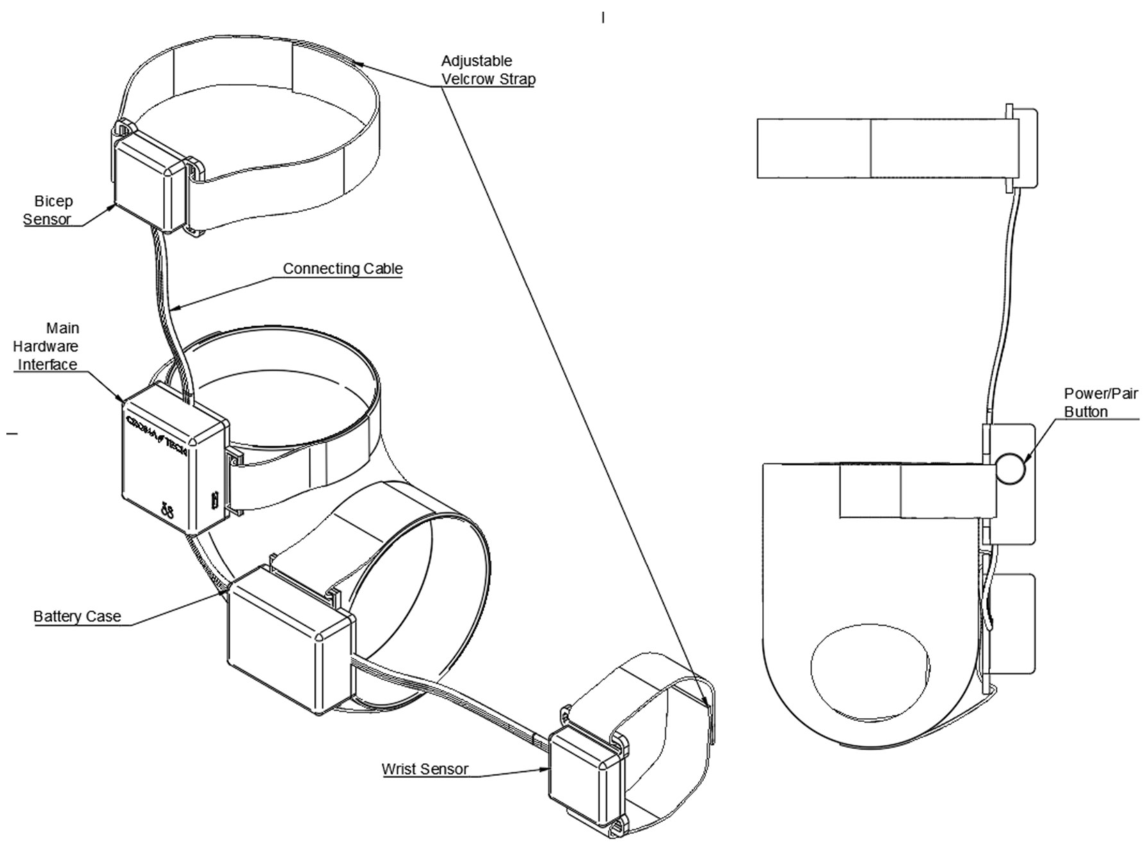


Figure 14: Mechanical Design Overview

The mechanical design of S-ROM is critical to ensuring the device is comfortable and easy to put on the injured arm. In addition, the design must be adjustable such that it allows users with a wide range of arm circumferences to use the device without blood flow restriction (too tight) or significantly disorienting the sensors (too loose). To meet both requirements, we will use two adjustable Velcro straps at the wrist and bicep locations and a compressive nylon/spandex blend sleeve at the elbow joint, as shown in Figure 15: S-ROM Velcro And Compressive Sleeve Design. The Velcro straps have arm circumference ranges of 15cm - 46cm, whereas the compressive elbow brace is for users with arm circumferences of 30cm - 40cm but will come with additional sizing options.

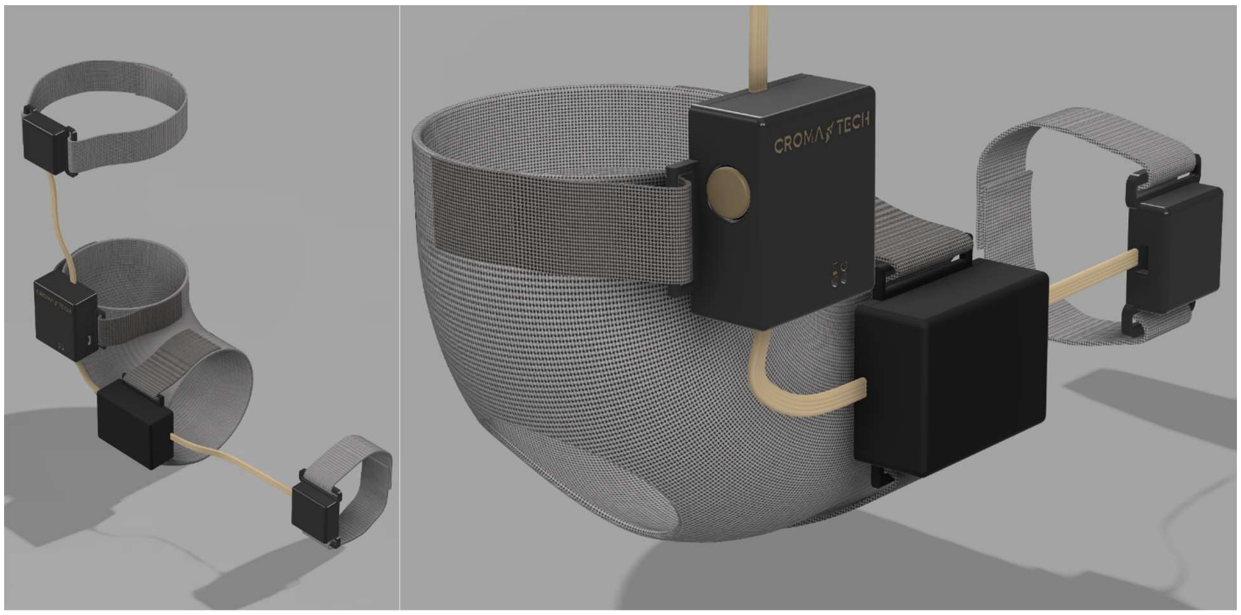


Figure 15: S-ROM Velcro And Compressive Sleeve Design

It is essential that the user can equip our device with only one functioning arm to allow patients to continue monitoring recovery progress at home through exercise measurements. The use of fastened sensors with adjustable tightness at the elbow and biceps ensure the user only needs one arm to hold the openings, slide the injured arm through, then tighten the straps as necessary with the healthy arm. Furthermore, to accommodate users with injuries in either shoulder, the S-ROM device is fully rotatable. An additional compressive opening on the elbow brace allows it to be flexed in either direction while the rest of the mechanical design is orientation independent. The nylon Velcro straps and nylon/spandex materials for the elbow brace are safe for human skin contact as they are common synthetic fabrics used in clothing.

The black plastic enclosures housing the electronics are designed to be modular and replaceable as the Velcro straps can slide out of the fasteners. This will extend the use life of our device and allow for simple repairs, as well as allow the fabric components to be sanitized. PLA plastic will be used due to its abundancy, low cost (30$/kg), and low weight (1.24 g/cm^3). A close-up of the IMU enclosure can be seen in Figure 16, where we see that the edges are rounded and not sharp. This, combined with the intended placement of the enclosures on the outside of the arm, ensure that these surfaces are never facing the user's body during any of the 3 exercises outlined for the PoC prototype demonstration. The plastic enclosures also ensure the device electronics are shielded from human sweat and splashes of water to help render our device IPX4 compliant.

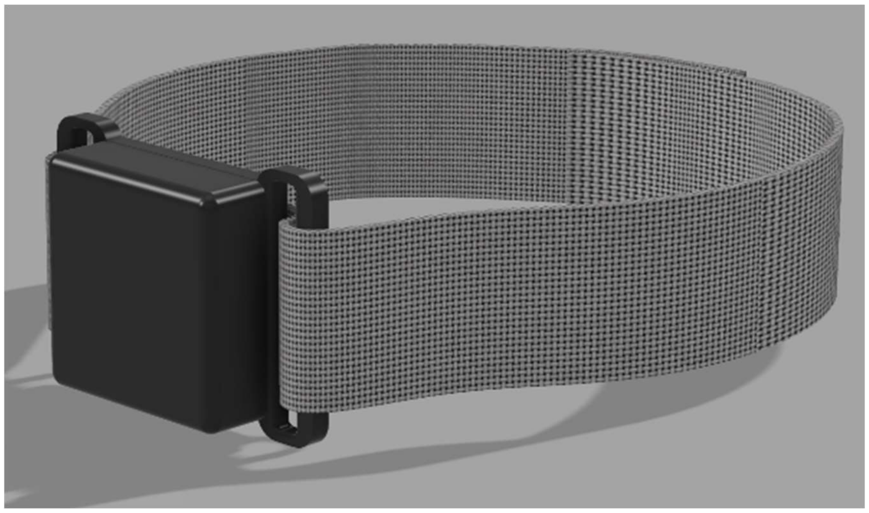


Figure 16: 9-axis IMU Enclosure

The main hardware enclosure, shown in Figure 17, houses the MCU and related UI circuitry including power-on (connected to the button shown on the left edge), Bluetooth, status LEDs, and the USB connector. For the PoC prototype, this will house the Arduino Mega 2650, requiring the following dimensions: 10.8 x 6 x 1.85cm.

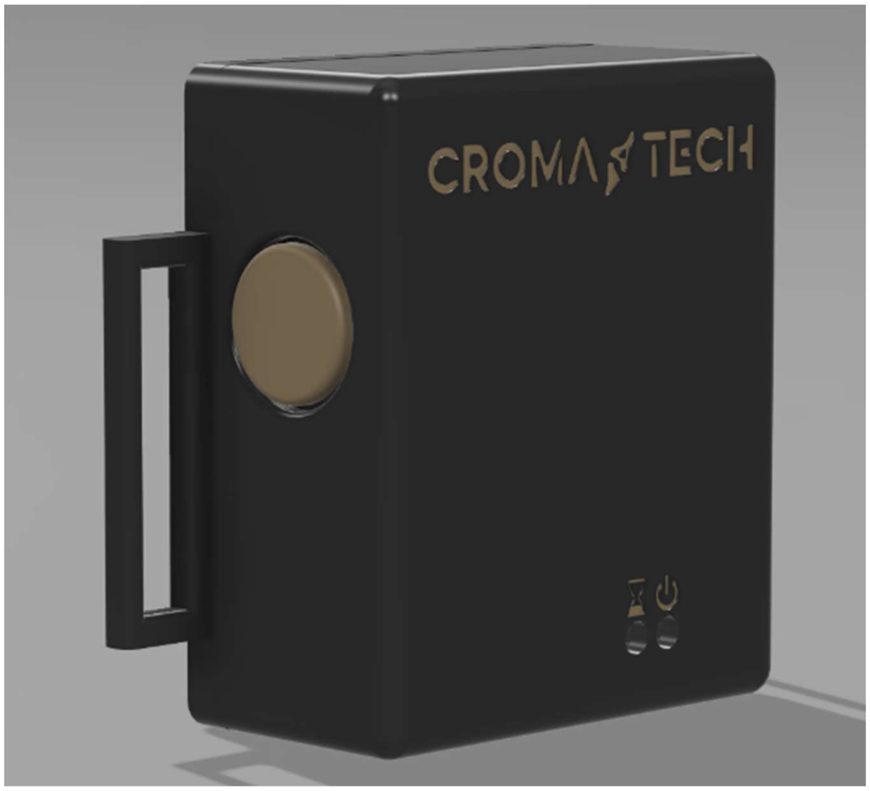


Figure 17: Main HW Enclosure

The battery enclosure houses the Lithium Polymer battery used for powering the device when used in wireless mode (i.e., via Bluetooth, therefore will not be included for the PoC prototype). It can be seen in Figure 18: Battery Unit Enclosure, along with the heat shrink providing protection for all connecting wires across the device [12].

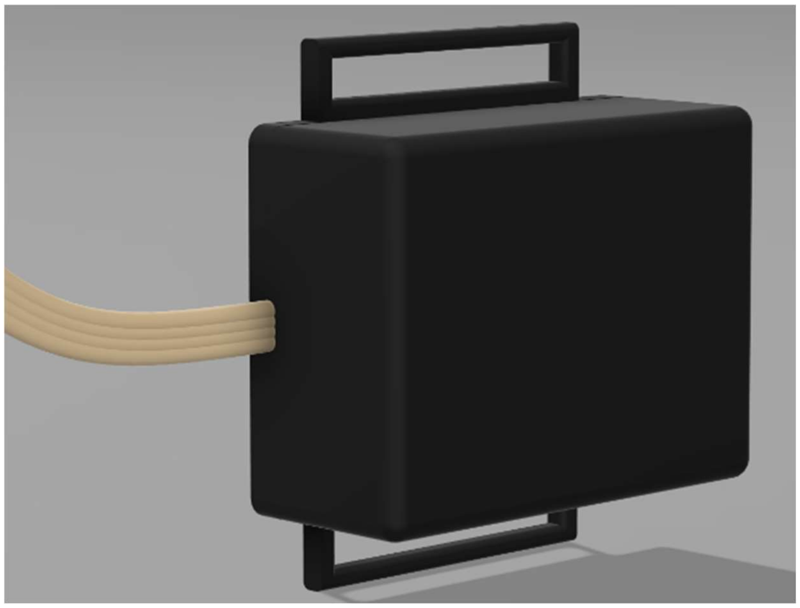


Figure 18: Battery Unit Enclosure and Heat Shrink Connecting Wires

In an additional attempt to lessen the burden on patients using the device at home, we wanted to design the S-ROM for easy transport. This is guided by requirements R2.1.12.B and R2.2.9.C which state that the device must be storable in a 30cm^3 space when not in use and weigh less than 500 grams, respectively. The current dimensions of the device shown in Figure 19: Profile of the S-ROM Device and Figure 20: Front and Top Views of the S-ROM , highlight the dimensions of the device in a slightly bent state. Using these dimensions, we can find that the maximum dimensions of the device when fully extended are 48 x 14 x 3cm. However, the elbow brace design allows the user to fold the device and decrease the dimensions to 16 x 14 x 6cm. The overall mass of the device (when in use, i.e., not including packaging) can be calculated by summing the individual masses of its components, shown in Table 9.

Table 9: Total Mass Calculation for PoC and Engineering Prototype Calculations

Item	Weight (g)
Velcro Straps (2x)	33
Elbow Brace	64
Arduino Mega	62
9-axis IMUs (2x)	14
Plastic Housings	32
Wire	20
Battery Pack (not included in PoC)	20
TOTAL (PoC)	245

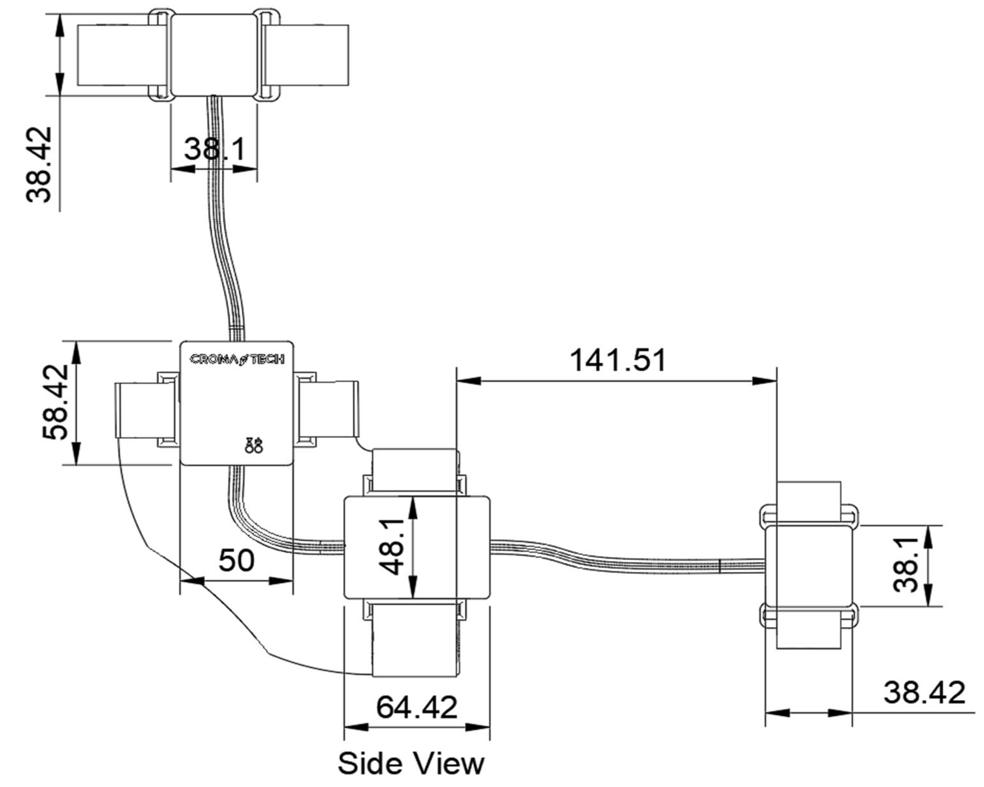


Figure 19: Profile of the S-ROM Device (units: mm)

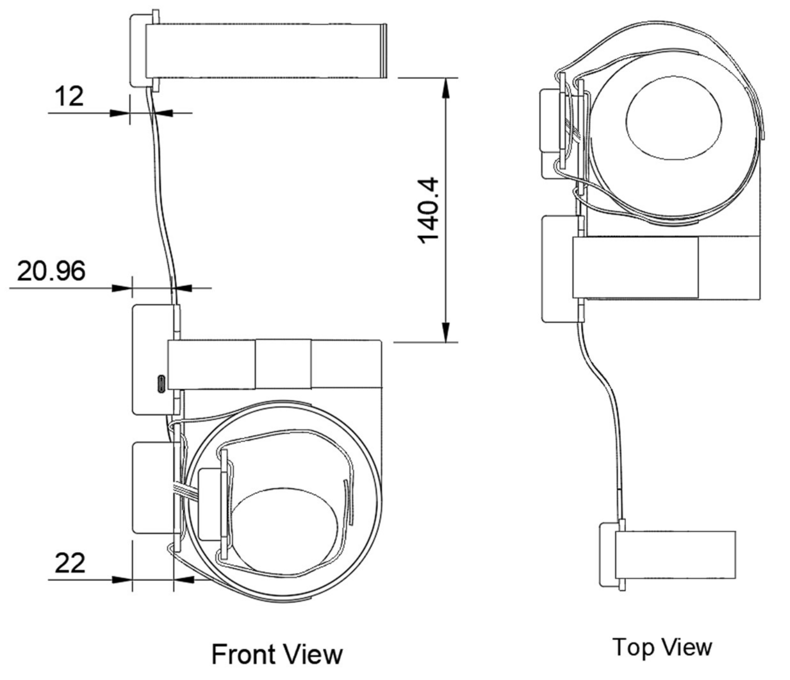


Figure 20: Front and Top Views of the S-ROM Device (units: mm)

2.6) Firmware Specifications

The FW is the code that runs on the MCU of the S-ROM. It oversees configuring the components at boot, handling error conditions, facilitating communication with the GUI SW, and setting the LED states. The following table, Table 10: FW Design Specifications, outlines the FW design specifications.

Table 10: FW Design Specifications

Design Spec ID	Description	Related Req ID(s)
D2.6.1A	FW will manage booting the device upon power on	Req 2.1.11.B
D2.6.2.A	FW may use the USB to communicate with GUI	Req 2.2.1.A
D2.6.3.A	FW responds to messages sent from GUI	Req 2.2.6.B Req 2.3.13.B Req 2.3.15.B
D2.6.4.A	FW will be aware of the different exercise types and be able to modify calibration accordingly	Req 2.2.7.B Req 2.3.7.B
D2.6.5A	FW will send data to GUI from sensors including corrected values from calibration	Req 2.3.1.A Req 2.3.3.A
D2.6.6.A	FW will calibrate device when messaged to	Req 2.2.7.B
D2.6.7.B	FW will set LED values to display status of device	Req 2.3.12.B
D2.6.8.B	FW will send error messages to GUI; it will do so in the response from a GUI message	Req 2.3.13.B
D2.6.9.B	FW can use the Bluetooth port to communicate with GUI	Req 2.2.4.B
D2.6.10.B	FW will prioritize USB over Bluetooth when both are connected	Req 2.2.1.A
D2.6.11.B	FW will communicate battery percentage to the GUI	Req 2.3.15.B
D2.6.12.B	FW will manage sensor configuration	Req 2.1.11.B

2.6.1) Sensor Configuration

The sensors used for angle measurements are the ICM-20948 9-axis IMUs, which feature a DMP that must be configured by FW. This DMP uses sensor fusion for the outputs of the embedded accelerometer, gyroscope, and magnetometer to estimate the relative angles of the sensor body. The algorithm used to combine these sensor outputs allows us to obtain a result that keeps drift to a minimum.

Using SparkFun's library for the DMP, parameters such as the sensor ranges, the sampling mode and rate, as well as digital low-pass filter bandwidths can be configured [13]. For the PoC prototype, these settings are implemented as static values; however, for the engineering prototype, they will be altered based on the calibration procedure. Furthermore, the output precision can also be defined. Rotation values are outputted as quaternions, since it helps prevent a common phenomenon known as gimbal lock [14]. This occurs when two axes align, causing the system to lose one degree of freedom when the math is performing with Euler angles. Quaternions are an alternative method of representing rotations which uses four values rather than three, allowing the system to avoid gimble lock. This is why Quaternions are used over Euler angles.

2.6.2) FW States

The FW will keep track of the current state of the S-ROM HW. The following figure, Figure 21: FW State Diagram, summarizes the FW states as well as the messages received to move between them. For more information on the messaging see section GUI Communication outlines the messaging between FW and the SW GUI.

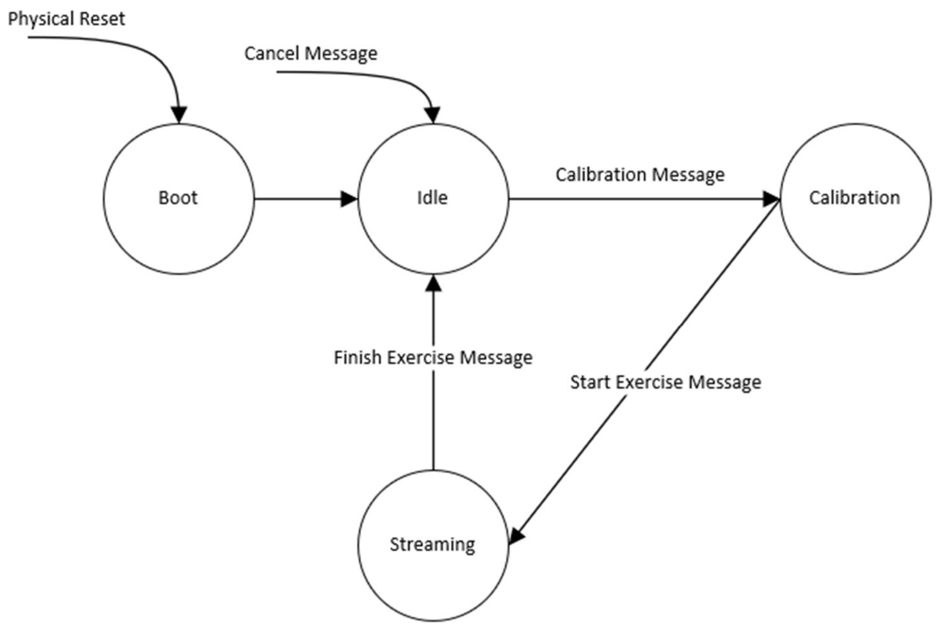


Figure 21: FW State Diagram

2.6.2.1) Boot

The FW handles booting the S-ROM. During initialization, the FW will configure the IMUs which have an integrated digital motion processor (DMP) that must be programmed each time the device is turned on. This programming sequence will be required for each of the IMUs. Additionally, the FW will configure both the serial port (USB-C) and the Bluetooth controller at startup. When a device is plugged into the serial port it will take precedence over a Bluetooth connection to maintain the battery life. Once the device has been booted up the FW will go into an idle state; From the idle state, the FW waits to receive a message from the GUI to start calibrating for an exercise. The following figure, Figure 22: FW Boot Sequence, shows the boot sequence beginning with the user turning on the S-ROM.

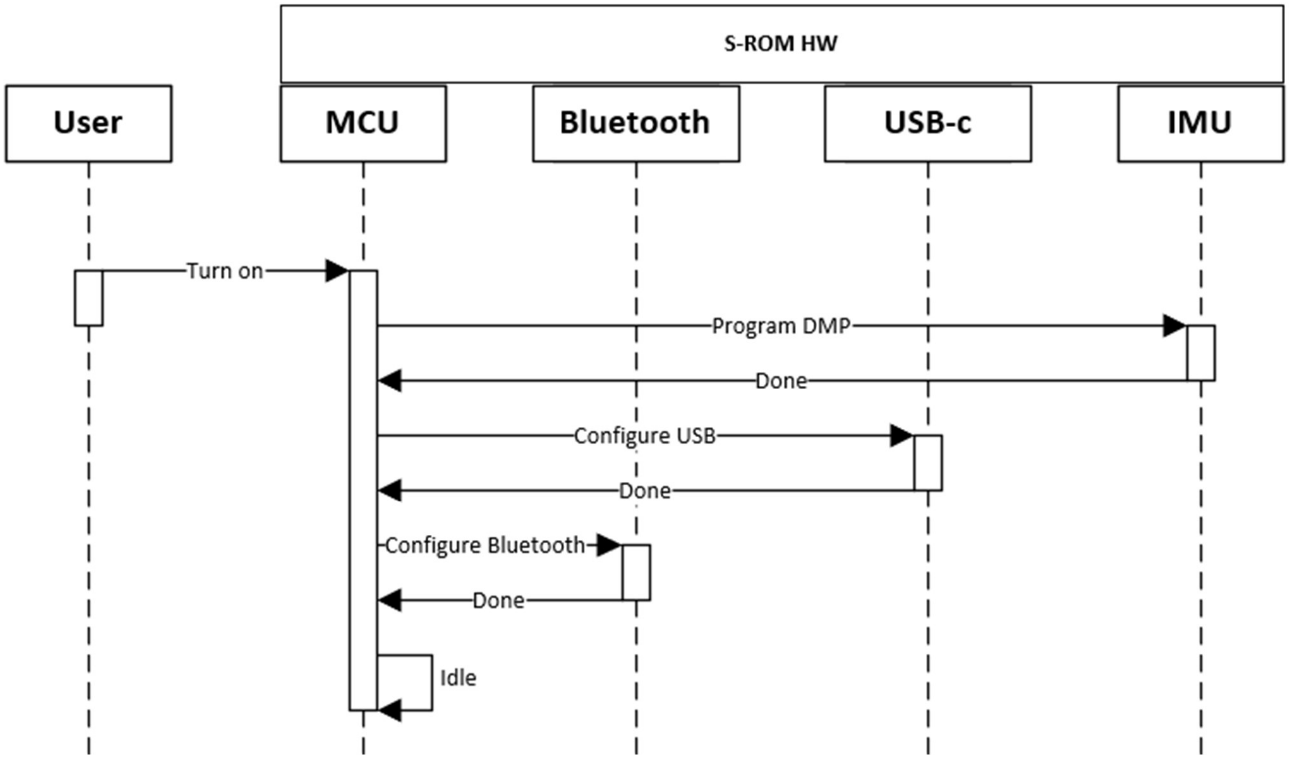


Figure 22: FW Boot Sequence

When the boot is in progress, messages will not be responded to until the boot is finished; this is because during the boot sequence, the connection to the GUI is not established yet, meaning it would be unreliable to attempt to reply to messages. However, if boot is unsuccessful, messages will respond to notify the GUI that the S-ROM is in a live-lock state.

2.6.2.2) Idle

When the FW is in the idle state it will only respond to messages from the SW GUI. The idle state can be entered from any state when the GUI sends the cancel message. For more information on messaging, see section GUI Communication.

2.6.2.3) Calibration

The calibration state is where the FW will collect data from the IMUs and store the necessary information for calibration. The calibration state will know the type of preceding exercise so that different calibration settings can be used for each exercise type. For example, the external rotation exercise requires the user places their hand on their stomach for calibration, whereas the flexion and abduction exercises require the user to place their arm along the side of their body – this initial position will affect the calibration procedure.

2.6.2.4) Streaming

Once calibration is complete, the state will change to streaming. While in the streaming state, the IMU will send information regarding the current positions of the IMUs with corrections from the calibration settings recorded while in the calibration state.

2.6.3) GUI Communication

GUI Communication will be implemented using polling. Polling involves the FW checking for requests from the GUI such as reset exercise or begin calibration. The FW will respond to all messages from the GUI eventually, where responses can either be an Acknowledge (Ack) or a Not-Acknowledge (Nak). When the FW responds with a Nak, the GUI message is unexpected and will be considered an error. When responding to the GUI, the FW sends a status message as well. This message will contain information about why the message was Nak or if there are any fatal HW errors such as an unreachable IMU. The full list of status messages that may be returned is in Table 13: Status Code. When receiving a message from the GUI the message will tell the S-ROM what action to take. The following table, Table 11: GUI to FW Messages, shows the possible messages to be received from the GUI.

Table 11: GUI to FW Messages Overview

Data [7:4]	Message (Code) [3:0]
Reserved	Heartbeat (0x0)
Exercise ID	Calibration (0x1)
Reserved	Begin Exercise (0x2)
Reserved	Finish Exercise (0x3)
Reserved	Cancel (0x4)

Table 12: FW to GUI Packet Format, shows the format of data sent the GUI for different packet types. The data count bits are to ensure that the data being received has not lost a packet which may happen when Bluetooth communication is used. "Device ID" is the ID of the IMU the data is from, and "Axis" is the ID of the axis data (x, y, or z).

Table 12: FW to GUI Packet Format

N	Bits	Ack	Nak	Data Low	Data High
8	7:0	Battery Health		Lower half word	Upper half word
8	15:8	Status			
2	17:16	Current State		Data Count	
1	18	Bluetooth			
1	19	Reserved			
2	21:20			Reserved	
2	23:22			Axis	
4	27:24			Device ID	
4	31:28	0x1	0x2	0x3	0x4

The FW GUI packet is restricted to 32 bits because our engineering prototype MCU has a word length of 32 bits; this choice will make the code simpler to write. The consequence of this decision is that each 32 bits of sensor data will have to be sent over 2 packets. The following table, Table 13: Status Code, shows the possible error messages responded by the FW in the status field of the Ack or Nak.

Table 13: Status Code

Status	Code	Description
Okay (no Error)	0x0	All is well
Unexpected message	0x1	The message received is not expected in the current state
Unknown message	0x2	The message received did not match a known message type
Unreachable Device	0x3	One of the sensors can not be initialized

2.6.4) FW Pseudo Code

The following figure, Figure 23: FW Pseudo Code, is FW pseudo code for the top-level functions.

```
int main(){
    message_t message;
    // Boot Initialization
    setup_imu(imu0);
    setup_imu(imu1);
    setup_serial_port();
    setup_bluetooth();

    while(1){
        message = rx_message_blocking();
        if(message == heartbeat || message == cancel){
            respond_ack();
        } else if (message == calibration) {
            respond_ack();
            perform_calibration();
        } else {
            respond_nak();
        }
    }
}

void perform_calibration(){
    message_t message;

    // Calibration Code

    message = rx_message_non_blocking();
    if(message){
        handle_message_calibration(message);
    }
}

void start_streaming(exercise_type){
    message_t message;
    data_t data0, data1;
    data0 = get_data(imu0);
    data1 = get_data(imu1);

    send_to_gui(data0, data1);

    message = rx_message_non_blocking();
    if(message){
        handle_message_streaming(message);
    }
}
```

Figure 23: FW Pseudo Code

2.7) Software Specifications

The following section will describe the software specifications for S-ROM. Most of the computational effort will be absorbed by the MCU, leaving the desktop application responsible for the GUI processing, minor data manipulation, and communication with the MCU. Note that requirement Req 2.5.7.B, relating our software to Canada's Medical Device Regulations, will be covered through the software test plan outlined in Appendix B. The following table, Table 14 Software Design Specifications, shows the software design specifications.

Table 14 Software Design Specifications

Design Spec ID	Description	Related Req ID(s)
D2.7.1.A	The SW application will be developed using MATLAB App Designer	Req 2.5.7.B
D2.7.2.A	The SW application reads real-time data from the MCU via 115.2kHz serial communication	Req 2.3.1.A Req 2.2.1.A Req 2.1.11.B
D2.7.3.A	The SW application plots real-time angular displacement data from the MCU on 3 plots with the primary angle being on the most prominent plot	Req 2.3.3.A Req 2.2.6.B
D2.7.4.A	The SW application includes a drop-down menu of 3 exercises for measurement	Req 2.1.1.A Req 2.1.2.A Req 2.1.3.A Req 2.3.7.A
D2.7.5.A	The SW application displays the current angle of the primary angle displacement	Req 2.1.6.A
D2.7.6.A	The SW application displays the maximum angle achieved in the primary angle displacement	Req 2.3.2.A
D2.7.7.A	The SW application detects and displays non-uniform motion, overcompensation, and incorrect exercise form via roll, pitch, and yaw angles, as well as raw angular velocity and acceleration data	Req 2.1.5.A Req 2.1.7.A Req 2.1.8.A Req 2.3.4.A Req 2.3.5.A
D2.7.8.A	The SW application has a button available for resetting the measurement after it begins	Req 2.3.13.B
D2.7.9.A	The SW application includes a tab for viewing previous exercise data (3 plots, max angle, overcompensation, incorrect form) organized by timestamp	Req 2.1.6.A Req 2.2.3.A Req 2.3.9.B Req 2.3.11.B
D2.7.10B	The SW application reads real-time data from the MCU via Bluetooth	Req 2.2.4.B
D2.7.11.B	The SW application displays a figure and text specifying how the user should position their arm for the calibration procedure.	Req 2.1.11.B Req 2.2.7.B Req 2.3.12.B

Design Spec ID	Description	Related Req ID(s)
D2.7.12.B	An optional display showing a stick-figure replicating the user's arm motion with the primary angle perpendicular to the viewing plane will be available	Req 2.3.6.B
D2.7.13.B	A skippable video demonstration of each exercise is played prior to the beginning of the measurement	Req 2.3.8.B
D2.7.14.B	An "Export Data" menu option is provided to export data to CSV format of a specific date range	Req 2.3.10.B
D2.7.15.B	A battery percentage indicator is included in the top right corner of the GUI.	Req 2.3.15.B
D2.7.16.C	Measurement data is uploaded to the cloud via a user account.	Req 2.3.16.C Req 2.2.10.C Req 2.2.12.C

2.7.1) GUI Specifications

For the PoC prototype, the GUI will be developed using the MATLAB App Designer tool. Using this tool allows us to meet all our complex data presentation requirements without being bogged down by the intricacies of custom GUI development. The entire Croma Tech team is familiar with MATLAB through prior coursework, so this will also help mitigate any time sensitive issues by allowing multiple team members to assist in development. A flowchart showing the GUI navigation options is shown in Figure 24.

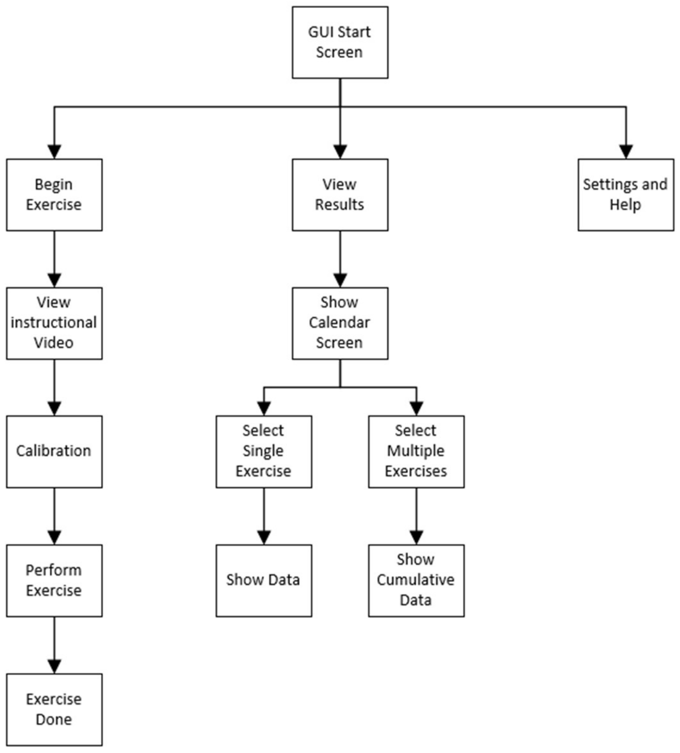
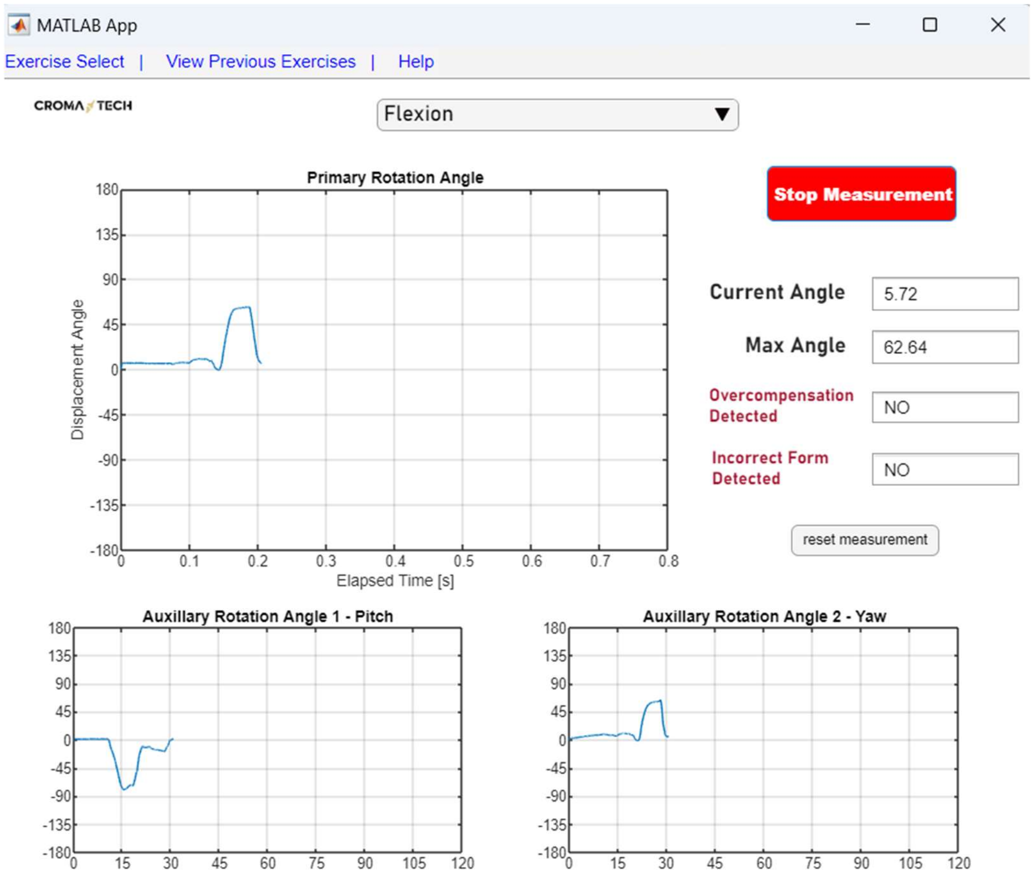


Figure 24: S-ROM GUI Navigation Flowchart

The main recording screen on the PoC prototype version of the software is shown in Figure 25. Here, the user can select an exercise through the top drop-down menu. Then, once the exercise begins after pressing the “Start Measurement” button, the user will be able to view real time plots with the primary angle on the large axis, and the additional angles for overcompensation and/or incorrect form detection on the smaller axes below. In addition, there four fields next to the primary angle plot displaying the current angle, max primary angle of the current measurement, overcompensation detected, and incorrect form detected. The user will also be able to reset a measurement if the software glitches or something disrupted their exercise via the reset button on the screen.



Current Angle 5.72

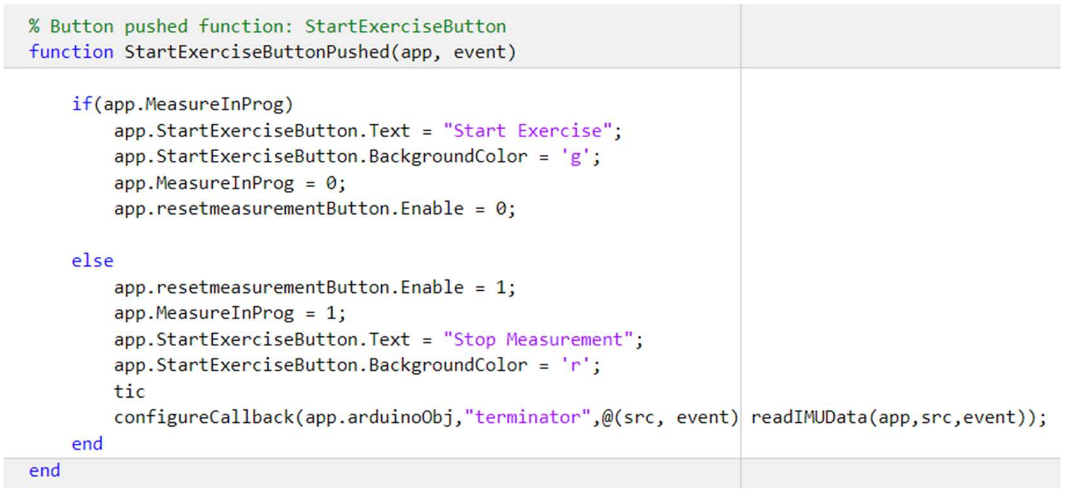
Max Angle 62.64

Overcompensation Detected NO

Incorrect Form Detected NO

Figure 25 - PoC Prototype Exercise Measurement Screen

The MATLAB application code takes advantage of the auto-generated application component properties and event callbacks. A sample of the code is shown in Figure 26, where we handle the case that the “Start Measurement” button is pressed in the GUI. It should be noted that this code is only an initial draft version and the intention is simply to show the structure of our code in the MATLAB App Designer tool. The custom global application variable, “MeasureInProg” can be accessed via the application window variable “app” and allows us to identify weather an exercise is in progress or not. If this callback is executed while the measurement is in progress, it means the user wishes to end the measurement. In this case, we reconfigure the button to signal to the user that they can start another exercise and disable the reset button to confirm the exercise is no longer being recorded. The other case is where the button is pressed outside of a measurement, indicating their desire to start an exercise. This is handled by reconfiguring the button to a "Stop Measurement" button and begin executing the callback function that checks for and handles serial data. In the final version of the software for the PoC, this will be changed to begin executing the calibration initiation function.



```
% Button pushed function: StartExerciseButton
function StartExerciseButtonPushed(app, event)

    if(app.MeasureInProg)
        app.StartExerciseButton.Text = "Start Exercise";
        app.StartExerciseButton.BackgroundColor = 'g';
        app.MeasureInProg = 0;
        app.resetmeasurementButton.Enable = 0;

    else
        app.resetmeasurementButton.Enable = 1;
        app.MeasureInProg = 1;
        app.StartExerciseButton.Text = "Stop Measurement";
        app.StartExerciseButton.BackgroundColor = 'r';
        tic
        configureCallback(app.arduinoObj,"terminator",@(src, event) readIMUData(app,src,event));
    end
end
```

Figure 26 - Initial Code for Response to Start Exercise button pushed.

The other main menu page that will be available for the PoC prototype will be the "View Results" page. Here, users will be able to select to view a previous measurement by date and type for the purpose of progress analysis. The previous exercises will be organized using a tree structure. Once the user selects their desired past result, 3 graphs will be populated with the data from that measurement as well as fields for the max angle, overcompensation, and exercise correctness. This page is currently under development for the PoC prototype, but the page will be similar to the anticipated design of the page for the engineering prototype, shown in Figure 27.

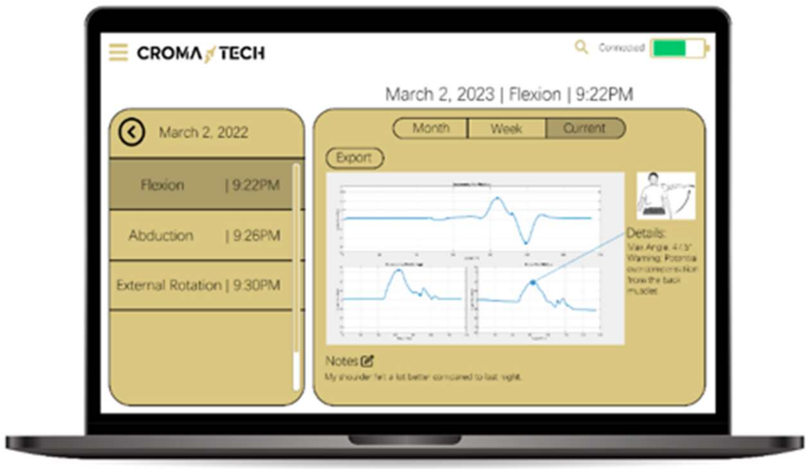

CROMA TECH
Connected
March 2, 2023 | Flexion | 9:22PM
March 2, 2022
Flexion | 9:22PM
Abduction | 9:26PM
External Rotation | 9:30PM
Month
Week
Current
Export
Details:
Max Angle: 47.5°
Warning: Potential overcompensation from the back muscles
Notes
My shoulder felt a lot better compared to last night.

Figure 27 - View Previous Exercise Page for Engineering Prototype Design

The engineering prototype will be developed using Python to provide even more flexibility in meeting our requirements specifications that the MATLAB App Designer tool cannot. This includes displaying a skippable video demonstration of how to perform the selected exercise, appropriate calibration messages, and a video representation of the patient's arm during the exercise. In addition, we will export the data above to a csv file as a bridge between our software and the physiotherapist's regular patient tracking software.

2.7.2) Data Manipulation

Data manipulation is an integral part of our system and we have defined it as any computational work that gets done outside of our MCU. The data manipulation can be split into two aspects: data computation and file storage.

There are a few pieces of data that we want to present: this includes non-uniform motion (Req 2.1.5.A), overcompensation (Req 2.1.7.A) as well as improper technique during the exercise (Req 2.1.8.A). During the exercise, we will be analyzing the data sent by our MCU in real time to check for these three different aspects and if any of it is detected.

Presently we are not certain as to how we will be detecting non uniform motion; however, the most likely technique will be to take the derivative of the angular position to see the angular velocity. Through the angular velocity, we hope to assess to a certain degree if the user is moving their arm smoothly or not. Depending on testing and the real-life response, we may or may not need to do some smoothing on the angular velocity to provide a clear picture of the response.

Overcompensation will be detected in a similar manner, where testing will be critical. There are two primary methods of overcompensation, which consists of the shoulder raising prematurely and the lateral trunk flexion. Trunk lateral flexion is a rotation of the upper body, with Figure 28 and Figure 29 outlining the anatomy of the shoulder and a visual representation of both types of overcompensation [15]. Figure 28: (a) Schematic anatomy of complex shoulder; (b) The compensatory movement in the task execution

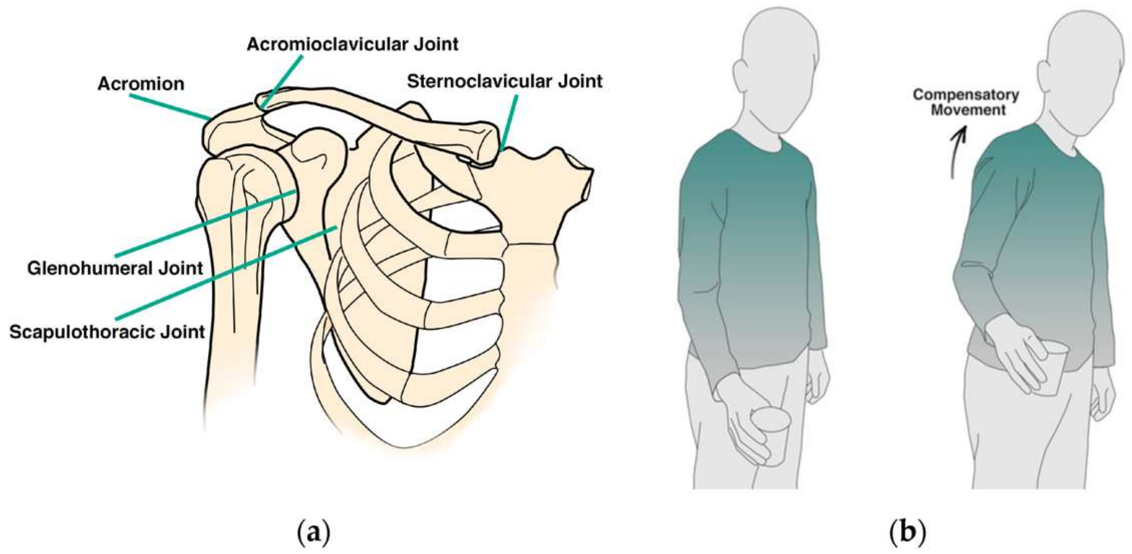


Figure 28: (a) Schematic anatomy of complex shoulder; (b) The compensatory movement in the task execution [15]

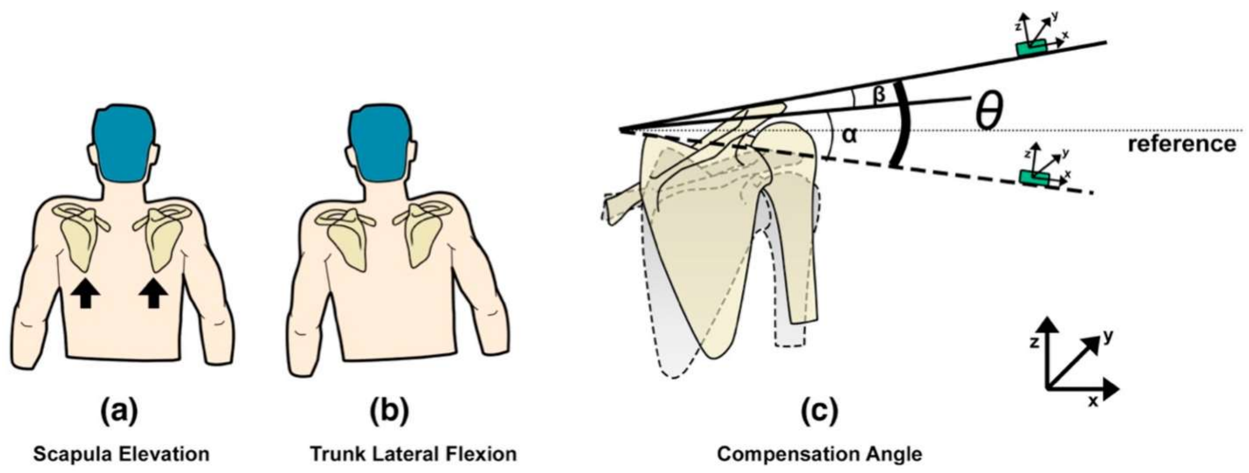


Figure 29:(a) Shoulder Raising Prematurely; (b) Trunk Lateral Flexion; (c)The overall compensation due to shoulder raising as well as trunk flexion [15]

Data storage saving is critical for us to be able to view previous exercises later. We will have to save files in a format that is easily accessible and can provide necessary information such as time of exercise, type of exercise, as well as if there was overcompensation detected during the exercise. These attributes would most likely be presented as a header of sorts in our file that we can read into our GUI.

2.7.3) Communication with the S-ROM wearable device

The SW oversees communicating the user inputs to the S-ROM. When the user selects options on the GUI that require the use of the S-ROM (exercises) the SW will send data to the S-ROM to notify the S-ROM to handle the user's request. For a more in-depth description of the data sent / received to / from the S-ROM see the corresponding FW specification section: GUI Communication. When communicating with the S-ROM the SW must ensure there are timeouts on requests and if a response is not received within that timeout, it must resend the request eventually the request should fail, and the SW should notify the user that it cannot communicate with the S-ROM.

3) Conclusion

Croma Tech is confident that our design of S-ROM will enable PTs to develop tailored injury recovery plans and improve the overall wellbeing of their patients. S-ROM will employ a straightforward mechanical sleeve design with adjustable Velcro straps enabled by a compact, yet effective hardware design centered around 9-axis IMUs. It will also be controlled by a sleek user interface that allows patients to record measurement for various exercises prescribed by their PT without any extensive RoM knowledge required; however, it will also provide PTs with configurable tools to analyze up to months' worth of RoM measurement data. There was no feedback provided on the Requirements Specification document that required changes to any aspect of S-ROM.

4) References

[1] "Medical Devices Regulations," Consolidated federal laws of Canada, [Online]. Available: https://laws-lois.justice.gc.ca/eng/regulations/sor-98-282/FullText.html. [Accessed 1 February 2023].

[2] L. Newlands, "How Often Are Physio Appointments," Strive Physiotherapy & Performance, [Online]. Available: https://strivept.ca/how-often-are-physio-appointments. [Accessed 11 February 2023].

[3] K. Hayes, "Reliability of five methods for assessing shoulder range of motion," *PubMed,* vol. 47, no. 4, pp. 289-294, 2001.

[4] D. Flores, E. Bircher, R. Nandakumar, D. Becher, A. Ang and L. Cuk, "Requriement Specification For S-ROM," Burnaby, 2023.

[5] Arduino, "Arduino Mega 2560 Rev3," 29 September 2020. [Online]. Available: https://docs.arduino.cc/hardware/mega-2560. [Accessed 1 March 2023].

[6] Microchip Technologies, "ATSAMD21G18," September 2021. [Online]. Available: https://www.microchip.com/en-us/product/ATsamd21g18#. [Accessed 4 March 2023].

[7] InvenSense, "ICM-20948 Datasheet," 9 February 2021. [Online]. Available: https://invensense.tdk.com/download-pdf/icm-20948-datasheet/. [Accessed 1 March 2023].

[8] u-blox, "NINA-W10 series (open CPU)," 19 December 2022. [Online]. Available: https://www.u-blox.com/en/product/nina-w10-series-open-cpu. [Accessed 4 March 2023].

[9] Lee's Electronic Components Ltd, "BATTERY, RECHARGEABLE, LI-POLY, 3.7V, 1000mAh," [Online]. Available: https://leeselectronic.com/en/product/8837-8837BATTERYRECHARGEABLELIPOLY37V.html. [Accessed 10 March 2023].

[10] Microchip Technologies, "MCP73833," May 2009. [Online]. Available: https://www.microchip.com/en-us/product/MCP73833. [Accessed 4 March 2023].

[11] Diodes Incorporated, "600mA, LOW QUIESCENT CURRENT FAST TRANSIENT LOW DROPOUT LINEAR REGULATOR," January 2023. [Online]. Available: https://www.diodes.com/assets/Datasheets/products_inactive_data/AP7366.pdf. [Accessed 4 March 2023].

[12] D.-K. Electronics, "RNF-100-1/16-BK-STK," Digi-Key Electrionics, August 2011. [Online]. Available: https://www.digikey.ca/en/products/detail/te-connectivity-raychem-cable-protection/RNF-100-1-16-BK-STK/573410. [Accessed 2 20 2023].

[13] SparkFun, "SparkFun_ICM-20948_ArduinoLibrary," 17 April 2019. [Online]. Available: https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary. [Accessed 15 February 2023].

[14] V. Koch, "IEEE Okanagan," 27 January 2016. [Online]. Available: http://www.okanagan.ieee.ca/wp-content/uploads/2016/02/GimbalLockPresentationJan2016.pdf. [Accessed 27 January 2023].

[15] Q. Wang, W. Chen, L. De Baets and A. Timmermans, "Motor Control Training for the Shoulder with Smart Garments," *MDPI,* vol. 7, no. 17, 2017.

[16] H. M. Devices, "HALO," Halo Medical Devices, 2018. [Online]. Available: https://halomedicaldevices.com/. [Accessed 20 February 2023].

[17] J. Owoyemi, "Kalman Filter: Predict, Measure, Update, Repeat.," Medium, 2017.

Appendix A. Design Alternatives

To ensure that the S-ROM design provides the best solution to meet our requirements, we have discussed and debated multiple design alternatives for various aspects of the system. Design decisions are supported by research, user interviews, and, ultimately, their agreement with our requirements specifications.

A.1) Angle Measurement Sensors

One of the primary goals for our system is that we can repeatedly collect data for three exercises within six degrees of a measurement made by a goniometer. This is outlined in requirements 2.1.1.A-2.1.4.A and 2.1.9.B. Based on these requirements there are two main options to carry out the measurements: an optical system (camera), or a sensor based system (IMU). Between these options we selected a sensor-based system with IMUs for three reasons. Firstly, an IMU application requires less computational capabilities that our end user may or may not have. Additionally, the GUI might be migrated to app instead of desktop app at which point camera might be more difficult. The computer vision algorithms that are needed are computationally quite intensive. Secondly, it is very likely that we would need either multiple cameras, a more expensive detector (something like a Kinect) or our patient to move around during the exercises to determine the angles for exercises where it is impossible to ascertain the angle of RoM from a two-dimensional perspective. Thirdly, using IMUs allow us a certain level of flexibility and scalability. We can track more complex and compound exercises. As well we can make more physical measurements on the user in the future, such as EMG.

A.2) 6-axis vs 9-axis IMU Selection

Our initial testing was performed using a 6-axis IMU. While it was able to provide reasonable roll and pitch measurements, the absence of a magnetometer meant it could not supply yaw measurements. The yaw measurements are crucial to detecting if users are performing the prescribed exercises correctly by only moving in one plane of motion. Theoretically, the issue could be solved by using two perpendicular IMUs, but to maintain a low-complexity mechanical design and ease the burden on our MCU, we opted to use one 9-axis IMU instead.

A.3) Sensor Fusion Algorithms

Unfortunately, the details of the DMP sensor fusion algorithm not open source. However, one algorithm that we were developing when considering the 6-axis IMUs was the Extended Kalman Filter (EKF), which helped develop us understand the underlying principles behind sensor fusion. For our application, we cannot find the rotational angles of the sensor directly, so we can only infer the rotation based on sensor values. The EKF computes a new rotation estimate based on the previous rotation and a model of our expected behaviour using a primary sensor [13]. This is compared to the other sensor model readings and the final rotation is weighed based on the covariance of each prediction. In other words, the filter determines which sensor is most reliable at a given time and allows it to contribute more to the final rotation prediction to decrease drift in the output. The DMP implements a sophisticated sensor fusion algorithm from our testing.

A.4) Wearable Device Materials

Critical factors in the decision making for material selection was comfort and time constraints. One design option was a full sleeve with our sensors and circuitry embedded in the fabric. We ultimately decided against this design as it could cause discomfort for those with larger arm circumferences and may impact the integrity of the measurement for those with smaller arm circumferences. Taking inspiration from our competitor, the HALO Digital Goniometer, another design option was to constrict all the electronic and mechanical components into one enclosure that can be place at different points of the arm, as shown in Figure 30. Although this design would meet most of our requirements, it fails to account for muscle overcompensation some users may be susceptible to – a key functionality of our device. In addition, for patients to use our device at home, another key requirement, we believe that this design would cause additional confusion over proper placement. The fully adjustable sleeve of our current design removes ambiguity in placement by having a clearly defined elbow section and incorporates checking for overcompensation through the upper IMU.

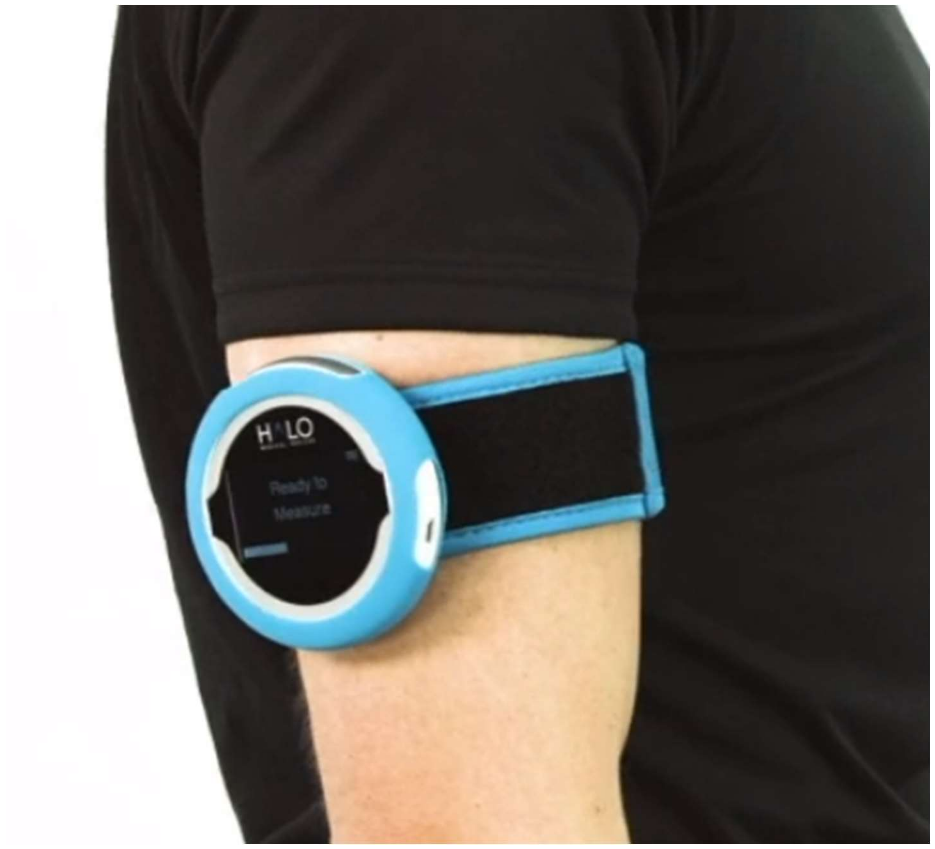


Figure 30: HALO Digital Goniometer [16]

A.5) GUI Software

The S-ROM GUI is critical to the success of the product across all 3 project development stages. The primary options for GUI development our team identified were MATLAB, Python, and Visual Studio Desktop App Developer with C++. Although our team is highly familiar with C++, none of the members have developed a desktop application with C++ before. The customizability of the C++ desktop app was intriguing, but the overall learning curve may be too steep to allow us to meet our critical PoC software requirements (2.3.1.A – 2.3.5.A). GUI development in Python is common and thus has a plethora of online resources available to support development. However, there is still a considerable amount of overhead required to develop a GUI in Python that may impact our ability to complete the project on time. For these reasons, we decided that the MATLAB App Designer tool is the most effective option in terms of both time constraints and supporting all the features we require.

A.6) Processing System

The two options our group discussed for the S-ROM processing system was the Arduino and the Raspberry Pi. Most of the team has familiarity with both boards, so this was not a factor in our decision. Our initial research of 9-axis IMU revealed that the computations can be performed using the Arduino MCU and does not require an operating system (OS). To minimize cost and complexity to allow us to meet the time constraints of the demo day and our outlined cost requirement, 3.7.1.B, we decided to proceed with the Arduino. Further research revealed that the DMP Library supported by the ICM-20948 9-axis IMU requires 35kB of program storage space whereas the Arduino Uno only supports 32kB of flash memory. The Arduino Mega, on the other hand, provides 256kB of flash memory, which is enough to store the DMP library and allow us to perform additional computations if necessary.

Appendix B. Test Plan

We have developed this test plan to validate the sub-systems for the S-ROM PoC prototype. The testing will be separated into three different categories: software, hardware, and system testing.

B.1) Software Test Plan

The software is primarily focused on manipulating sensor data from the Arduino to achieve accurate detection of overcompensation and incorrect exercise form, as well as providing the GUI. The overcompensation and incorrect exercise form testing will be classified under System Tests.

Test: GUI Functionality	**Time:**	**Date:**
Purpose: - Confirm the graphs displayed are an accurate representation of the output data. - Check for bugs in any of the menus and interactable elements. - Evaluate the latency of the displayed data.		
Procedure: 1. Open the S-ROM Application. 2. Select the Flexion Exercise (repeat the sub steps for all 3 exercises supplied in the PoC prototype) a. Start the exercise by pressing the “Start Exercise” button. b. Attempt to reset it by pressing the “reset measurement” button. 3. View the exercise by navigating to the “View Previous Measurements” tab and selecting the corresponding exercise.		
Pass/Fail Checklist: ☐ All interactable elements work as intended by performing their named function. ☐ Plotted data responds to within the latency outlined in Req 2.2.6.B. ☐ Plotted data in main window reflects the motion of the patients arm.		
Notes:		
Contingencies: Issues with plotting latency may require small changes to the MATLAB code at minimum but may require more fundamental changes such as intermediate/outsourced data processing in the worst case.		

B.2) Hardware Test Plan

Test: Sensor Configuration	**Time:**	**Date:**
Purpose: - Determine the optimal digital filter, sensor ranges, and sampling time for the best capture of the movements (changes in speed, accuracy, etc.). - Observe the affect of different configurations for a future calibration implementation. - Confirm proper communication with the MCU.		
Procedure: 1. Select a combination of the digital filters, sampling times, or sensor ranges in the FW. 2. Program the MCU with the selected configurations. 3. Mount one sensor to the end of the test rig arm and set it to resting position (straight down). 4. Begin recording sensor data. Also record a video for future reference. 5. Pull the arm up 90 degrees non-uniformly (try to keep this motion consistent across trials). Return the arm back to resting position. 6. Start a new recording and rotate the arm clockwise 90 degrees at a non-uniform speed. Measure the true position using a compass and return to resting position. 7. Compare the captured results with the video recording. 8. Repeat for a human user.		
Pass/Fail Checklist: ☐ The captured motion matches the general motion of the user/test rig. ☐ S-ROM must report a final position angle within +-6 degrees of the true measurement for each trial to meet requirement Req 2.1.4.A. ☐ The details of the movement are properly represented (minimal noise and attenuation).		
Notes:		
Contingencies: Research of other DMP parameters might be required if the above test yields no clear results.		

Test: Sensor Accuracy	**Time:**	**Date:**
Purpose: - Evaluate the precision and accuracy of our max angles for two axes. - Identify the capabilities of our chosen sensor. - Evaluate the sensor drift (returning to original position).		
Procedure: 1. Mount one sensor to the end of the test rig arm and set it to resting position (straight down). 2. Begin recording sensor data. 3. Pull the arm up to the N degrees at a slow uniform speed. Hold the position and measure the true angle using a goniometer/compass. Return the arm back to resting position. 4. Start a new recording and rotate the arm clockwise M degrees at a uniform speed. Measure the true position using a compass and return to resting position. 5. Repeat for N = 45, 90, 135 and M = 45, 90, 135 for a total of 9 trials.		
Pass/Fail Checklist: ☐ S-ROM must report a final position angle within +-6 degrees of the true measurement for each trial to meet requirement R2.1.4.A. ☐ S-ROM's must report +-6 degrees after returning to the initial position.		
Notes:		
Contingencies: Failure of this test may prompt a change in sensor position on the test-rig arm. Also, the test could be repeated with different speeds.		

B.3) System Test Plan

This section will explore the performance of our product system and help us identify which configurations yield the optimal accuracy and user experience.

Test: Performance Across Different Exercises	**Time:**	**Date:**
Purpose: - Evaluate the S-ROM outputs over our three target exercises. - Observe if the output plots can be matched to its corresponding exercise.		
Procedure: 1. Follow and record the results for all tests defined in System Test Plan. 2. Repeat for Flexion, Abduction, and External Rotation.		
Pass/Fail Checklist: ☐ The results of each exercise can be differentiated from each other. ☐ The results pass every test in System Test Plan.		
Notes:		

Contingencies: Additional sensors in the mechanical design or sensor placement may be required to fully capture the motion.

Test: Sensor Placement and Fusion (Forearm vs. Upper Arm)	**Time:**	**Date:**
Purpose: - Compare the performance of the forearm sensor vs. the upper arm sensor. - Evaluate the output for placement on different parts of the arm.		
Procedure: 1. Mount sensors to the patient under test. 2. Begin recording sensor data. 3. Perform the exercise to N degrees at a slow uniform speed. Hold the position and measure the true angle using a goniometer/compass. Return the arm back to resting position. 4. Move the sensors closer to the elbows. Repeat steps 2 and 3. 5. Move the sensors farther from the elbows. Repeat steps 2 and 3. 6. Repeat for N = 45, 90, 135, for a total of 9 trials.		
Pass/Fail Checklist: ☐ For exercises where both sensors are parallel, their measurements behave similarly on the same axis (flexion, abduction). ☐ The sensor placement yields a minimal difference in output.		
Notes:		
Contingencies: Additional sensors may be needed if not all our exercises can be measured properly. Also, the mechanical design will have to change if the sensors behave very differently for different sensor placements (closer to the lower arm, middle arm, or upper arm).		

Test: Sensor Sensitivity and Initialization	**Time:**	**Date:**
Purpose: - Check how the S-ROM output varies with minor changes in sensor positioning. - Evaluate the effectiveness of our current calibration process.		
Procedure: 1. Mount sensors to the patient under test. 2. Begin recording sensor data.		

3. Perform the exercise to N degrees at a slow uniform speed. Hold the position and measure the true angle using a goniometer/compass. Return the arm back to resting position. 4. Move the sensors two centimeters from its original place. Repeat steps three times. 5. Repeat for N = 45, 90, 135, for a total of 9 trials.
Pass/Fail Checklist: ☐ Results using slight deviations in sensor placement are within the accuracy outlined by Req 2.1.9.B
Notes:
Contingencies: Failure of this test could mean a more rigorous calibration implementation (regarding filter and noise configurations, starting position, etc.), as well as improving the mechanical design to limit sensor movement.

Test: Shoulder and Chest Overcompensation	**Time:**	**Date:**
Purpose: - Compare an overcompensation scenario with normal results. - Observe device capabilities for detecting overcompensation. - Determine whether of not should or chest overcompensation can be differentiated.		
Procedure: 1. Mount sensors to the patient under test. 2. Begin recording sensor data. 3. Perform the exercise with shoulder overcompensation to N degrees at a slow uniform speed. Hold the position and measure the true angle using a goniometer/compass. Return the arm back to resting position. 4. Start a new recording and perform the exercise while overcompensating with the torso at a uniform speed. Hold the position and measure the true angle using a goniometer/compass. Return the arm back to resting position. 5. Repeat for N = 45, 90, 135, for a total of three trials.		
Pass/Fail Checklist: ☐ The overcompensation case causes a significant difference in sensor output. ☐ Shoulder vs. Chest overcompensation sensor outputs are significantly different.		
Notes:		
Contingencies: We might require changing the mechanical design to accommodate for a chest or on-shoulder sensor.		

Test: Mechanical Design Ease of Use	**Time:**	**Date:**
Purpose: - Ensure mechanical design does not restrict user’s ability to put on the device. - Check for potential physical hazards during exercises.		
Procedure: 1. The tester will attempt to place the device on to one arm using only their opposite arm. 2. Perform each of the 3 exercises through the full range of motion.		
Pass/Fail Checklist: ☐ Device can be put on without significant complications or issues. ☐ There are no rigid edges (including fabrics rubbing against skin) affecting the user during any of the exercises.		
Notes:		
Contingencies: Issues arising from this test may result in minor changes in the mechanical design		