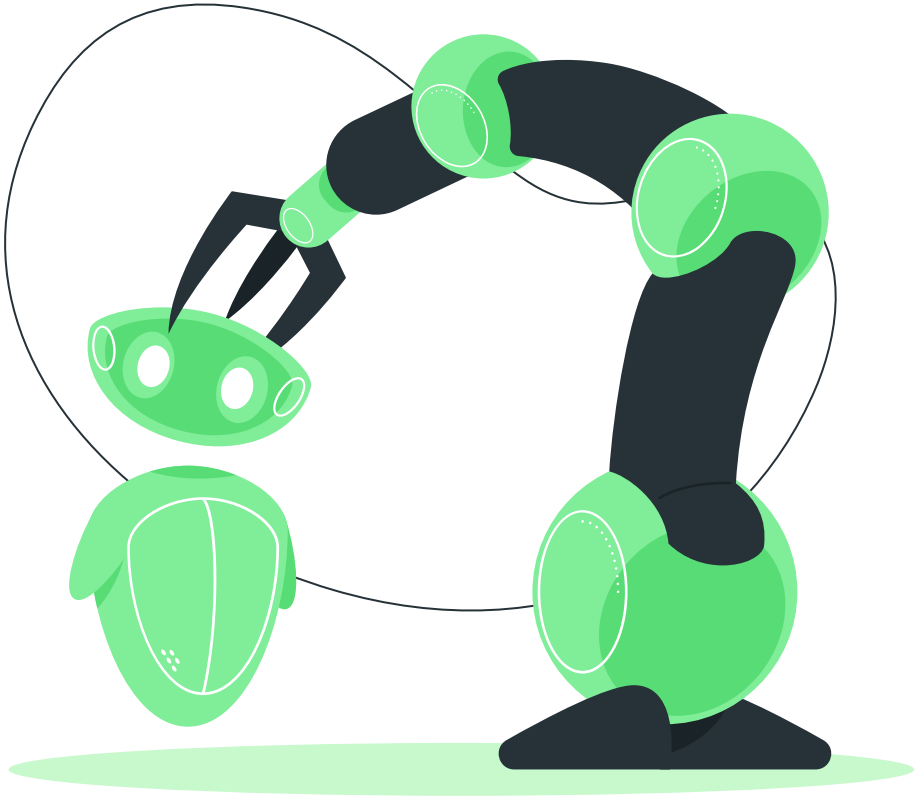
RTOS Based Teleoperated Robotic Hand

Capstone Project by
Momoe Nomoto

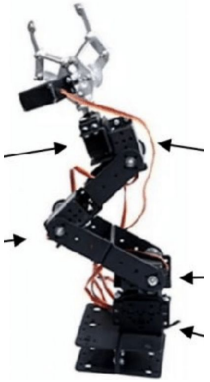


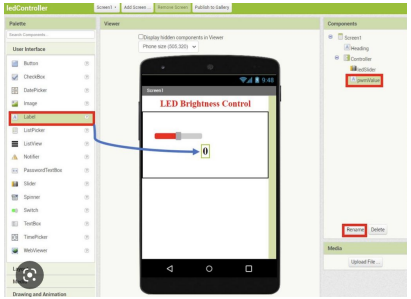
Introduction

Teleoperated robotic hand applications: Biomedicine, defense, space, agriculture

Inspiration: step up from embedded class project

Previous studies: uses Arduino, no RTOS, uses Bluetooth 2.0, WiFi, SPI, etc.





Taking it a step further

This project aims to construct a high-accuracy low-latency teleoperated robotic hand system controlled by a user-worn data glove through wireless communication. The robotic hand will replicate the movements of the operator's fingers using electrical signals generated from flex sensors on the data glove.

Goal: to optimize the responsiveness and accuracy of the robotic hand to human hand motion.

Project Timeline

Weeks 1-4

Idea Formulation and Research (Campus closed)

Weeks 5-9

Designing, learning, building, and refining the architecture (software and system)

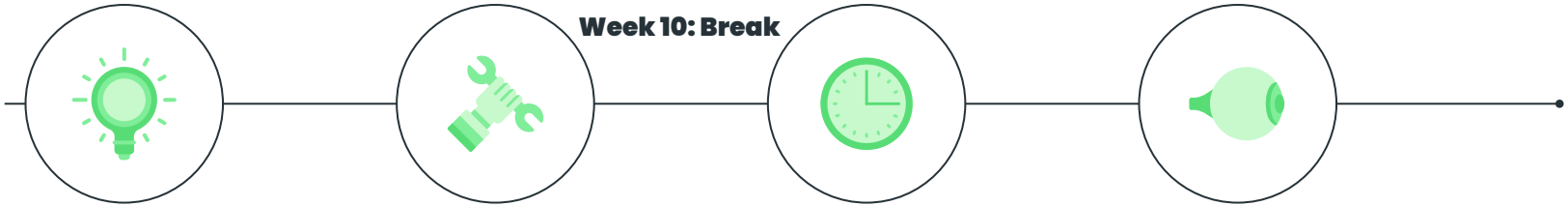
Week 10: Break

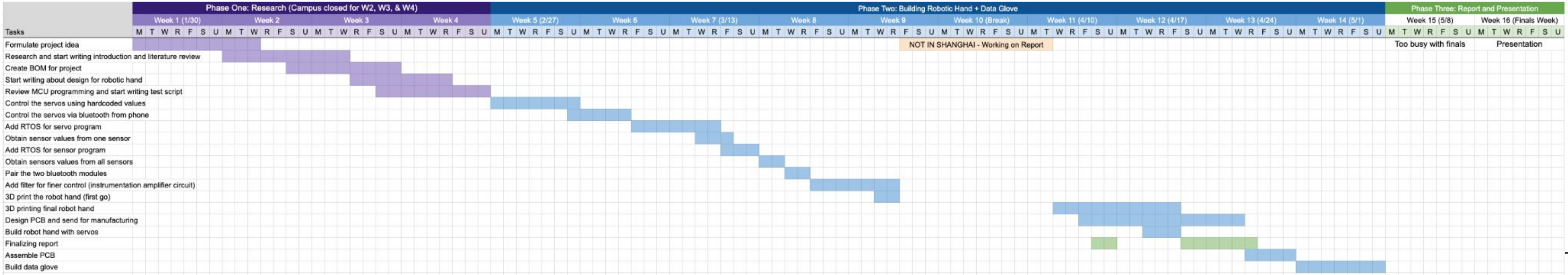
Weeks 11-14

Constructing and assembling the physical components (robot hand and data glove)

Weeks 12, 13, 15, 16

Finalizing the documentation, report and presentation

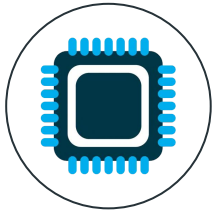




https://docs.google.com/spreadsheets/d/1nb2-rKGNAHlcRiBsdyGjEpayc8wCKy6AGg6D42CsivQ/edit?usp=sharing

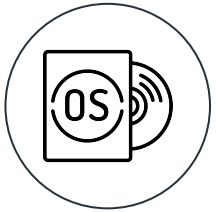
System Design

Features Of The System



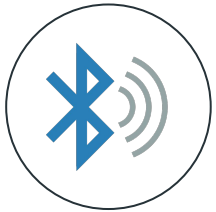
Microcontroller

STM Arm®
Cortex®-M7-core-based
STM32F746I



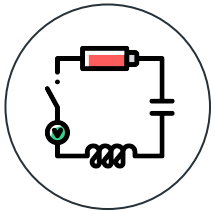
Real Time OS

Keil CMSIS-RTOS API v2



Wireless Comm Protocol

Bluetooth Low Energy 5.0



Signal Filtering

Signal Conditioning
Circuit for Flex Sensors

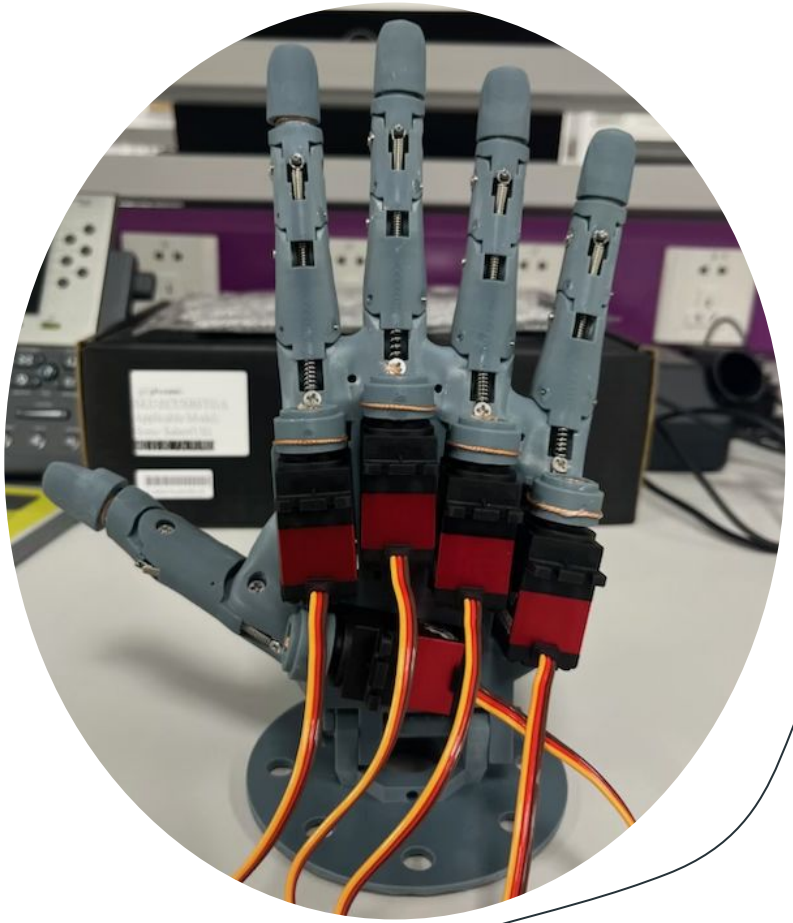
1

Hardware

Robotic Hand + Data Glove

Robotic Hand

- 3D printed hand from in-house printer (open source project)
- Five JX PDI-6225MG-300 servo motors (high torque, wide angle)
- Fishing wire to act as tendons
- Tensioning with springs and needles
- 6V power supply to increase speed (external to power source of MCU)
- Up to 1.5A per servo when working

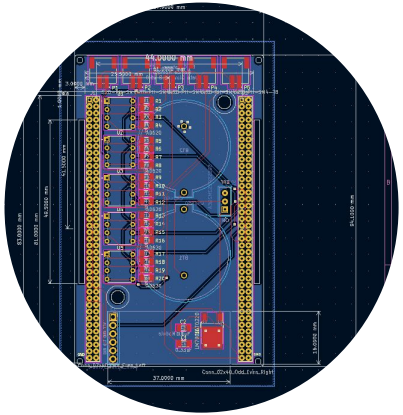


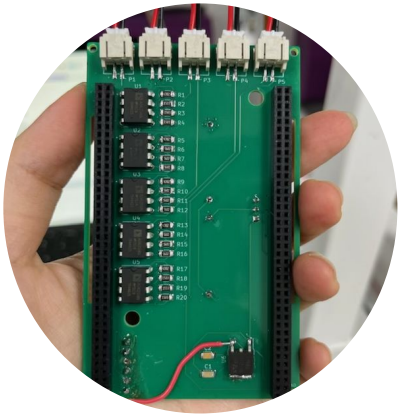
Data Glove: the glove

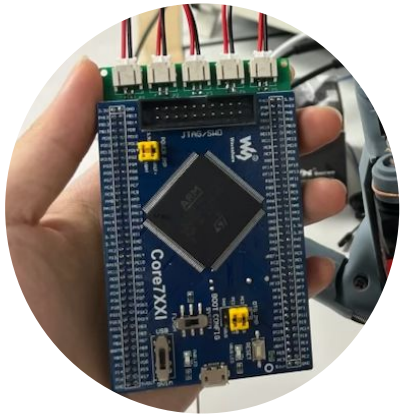
- Five flex sensors
  - Variable resistors: 10kΩ-24kΩ for longer & 26kΩ-70kΩ for shorter
- Electrical tape and glue

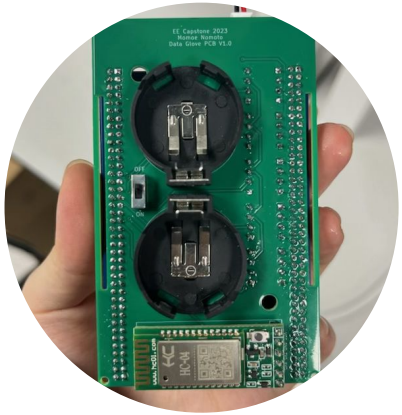
Data Glove: the controller

- Self designed PCB using KiCad
- Conditioning circuit first prototyped on breadboards
- Collected materials
- Self assembled

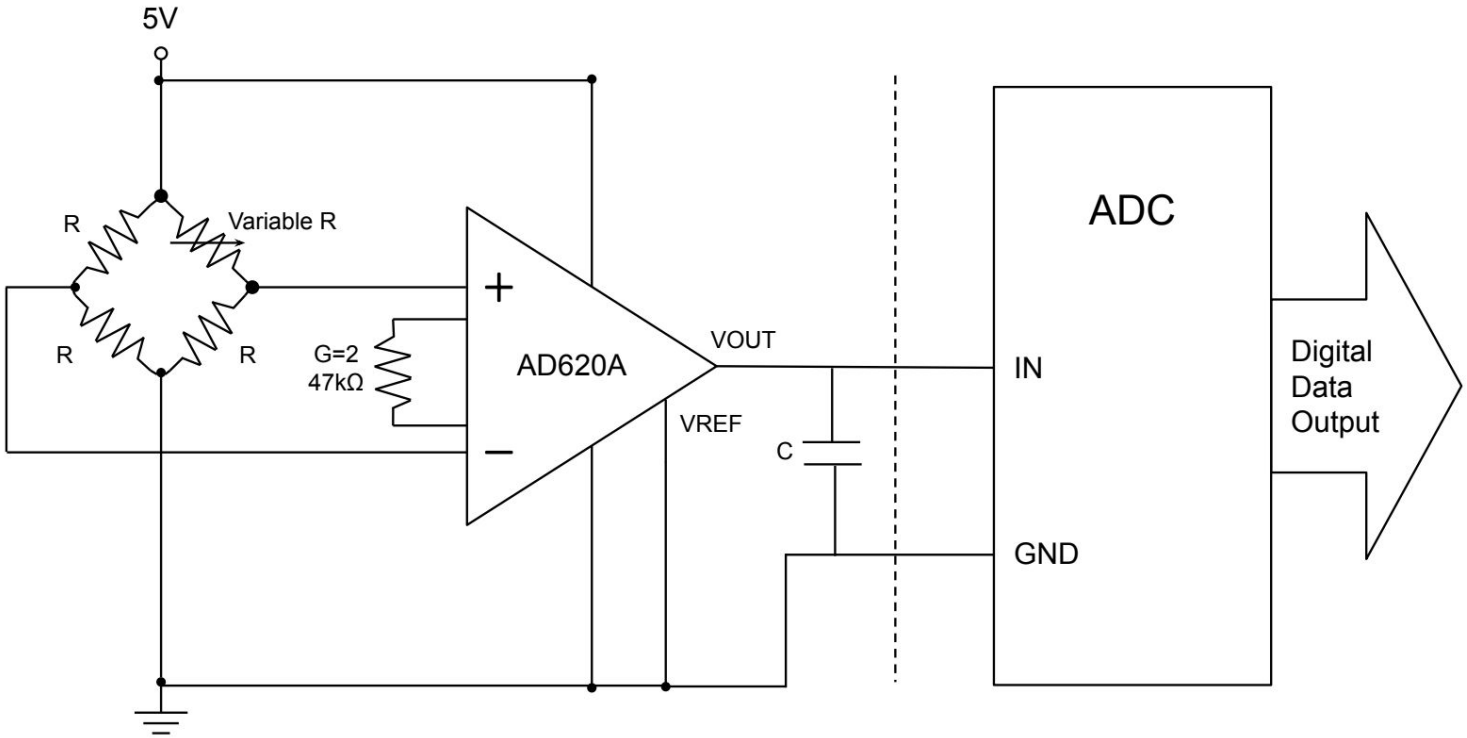








Conditioning Circuit



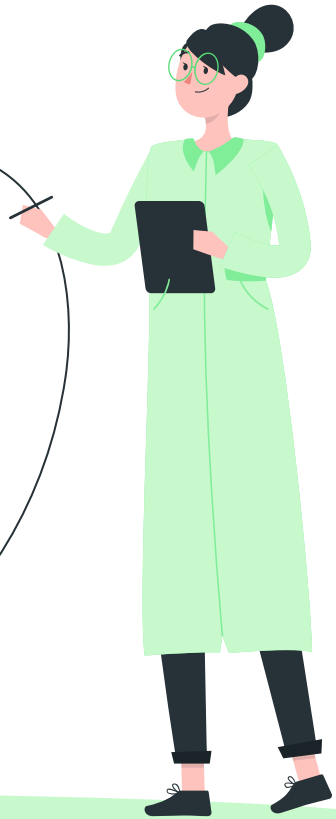
Communication Protocol

- HC-04 Bluetooth Module
- Bluetooth Low Energy 5.0
- Data transfer speed up to 2 Mbps
- A range of 10m
- Consumes only around 7mA of current when connected
- Baud rate of the module can be adjusted from 1200 bps to 921600 bps, but for this project, the baud rate of 9600 bps is used

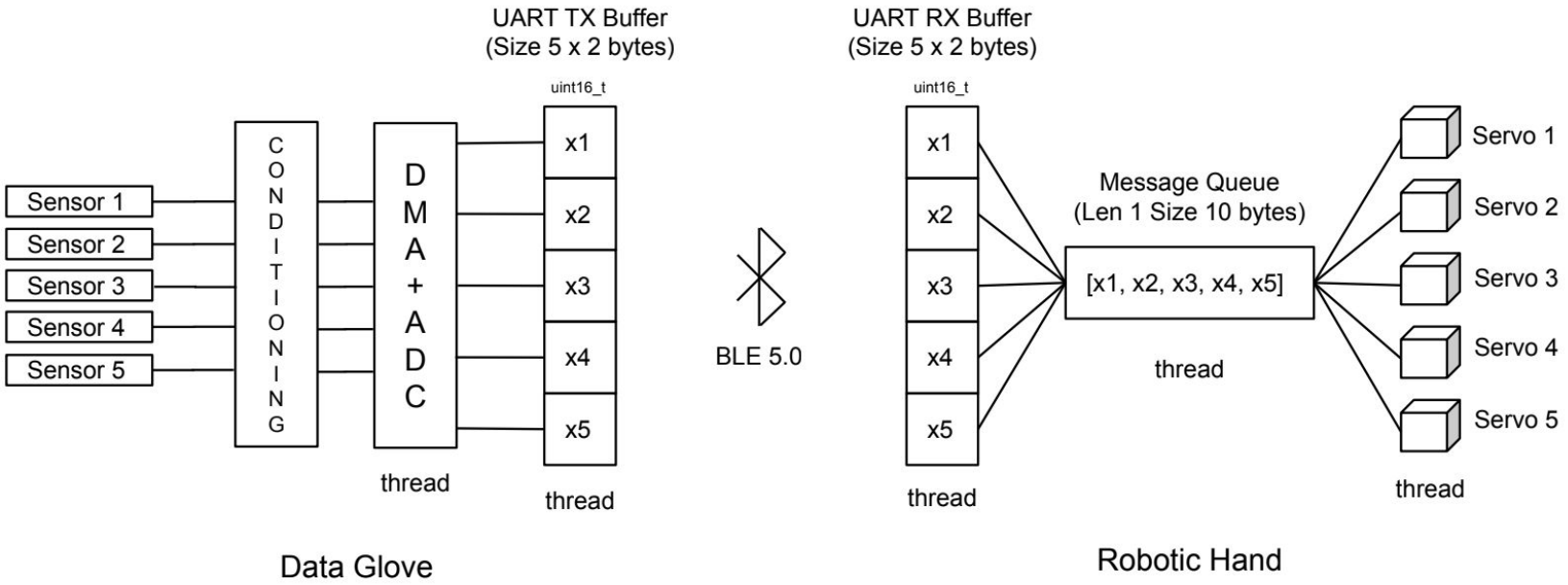
2

Software

Architecture + RTOS Design



Communication Architecture



ADC 0-3.3V converts to 0-4095 and maps to 500 to 2500 (CCRx)

One-way communication

RTOS Program

Non-blocking, save resources

Data Glove:

- Four threads created: TIM, ADC+DMA, UART, and LCD (low priority)
- Multiple event flags
- Five ADC channels

Robotic Hand:

- Three threads created: UART, SERVO, and LCD (low priority)
- Multiple event flags
- One message queue
- Two timers, Five channels

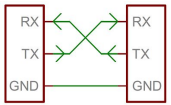
Data Glove Program

Timer (56.25Hz)
Generating interrupt at every rising and falling edge, triggering ADC measurements at a frequency of 112.5Hz

5 channels
A single conversion is performed for each channel of the group.
DMA mode enabled
Transfer of converted data in ADC_DR to a specified location
A transfer completion interrupt is generated

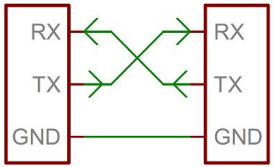
IRQ handler for the DMA raises the event flag for another thread to map the digital values into the range of the PWM duty cycle time from 500 to 2500. A threshold is set to determine if a new value should be saved in the UART TX buffer or not. If all values are 0, only the LCD thread event flag is raised. Otherwise, both UART and LCD event flags are raised.

❗ Bottleneck
Limited by ADC conversion rate



The UART thread generates an interrupt at data transmission to signal transmission completion.

Robotic Hand Program



UART thread continues to receive data (at a rate of 9600 bps) and at the event of UART data reception, an interrupt is generated and a callback function is called.

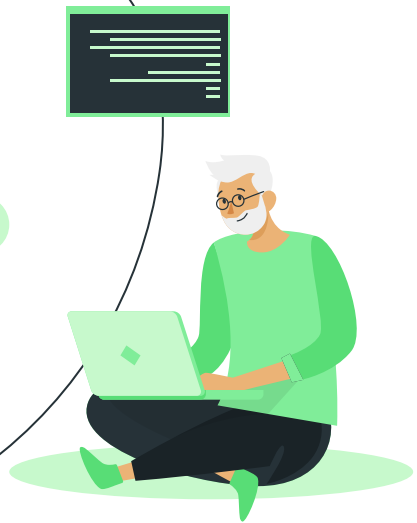
❗ Bottleneck

Callback function
Copy the contents of the buffer to a static shared memory array, releasing the buffer and allowing it to be filled with new information as the other threads handle the contents of the shared array. The array is then added to a message queue to wake up the servo thread, and two event flags are raised.

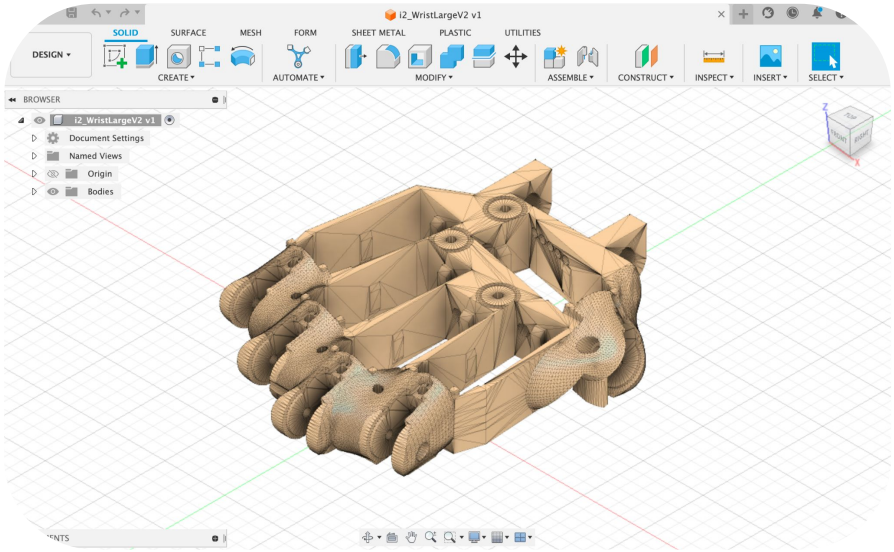
First event flag allows the UART thread to continue receiving data. Second event flag wakes up the LCD thread for printing.

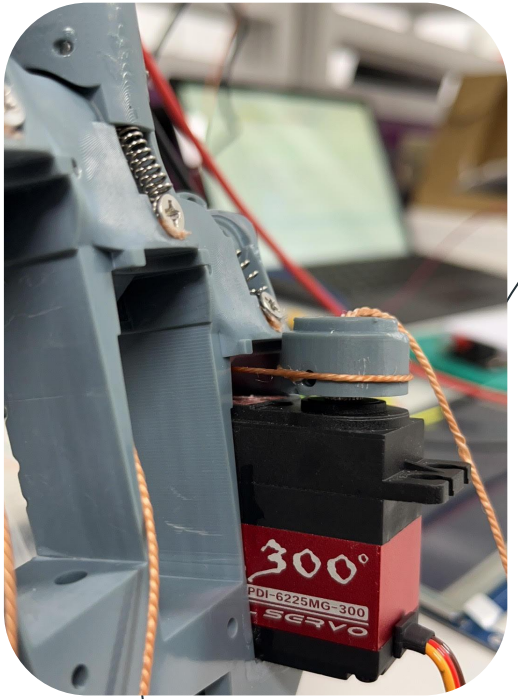
Once the servo thread receives a message in the queue, it assigns the values to the corresponding servo motors. If any of the value is 0, the position of the servo motor remains unchanged.

Experiments



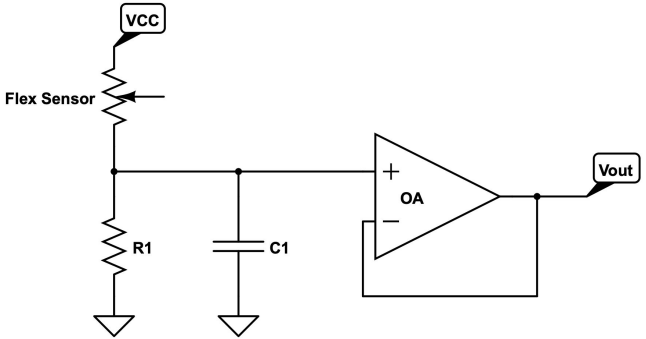
3D printing of hand

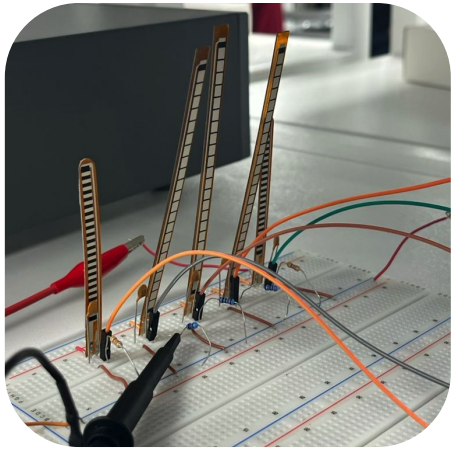


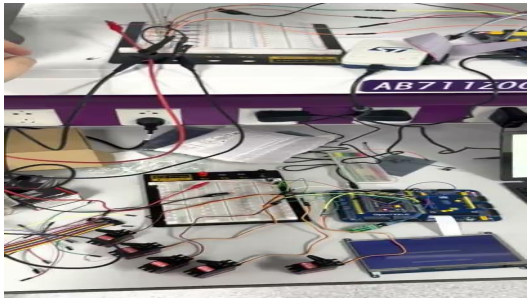


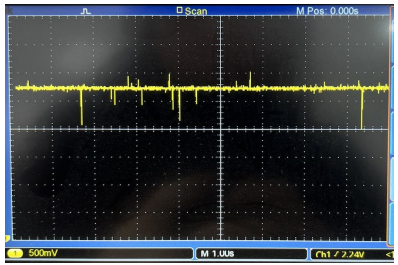
Conditioning Circuit Experiments

Initially:

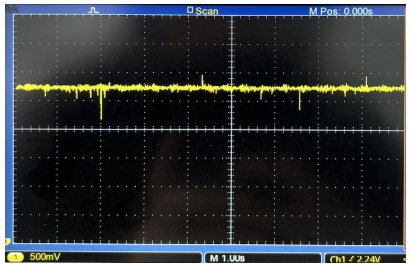




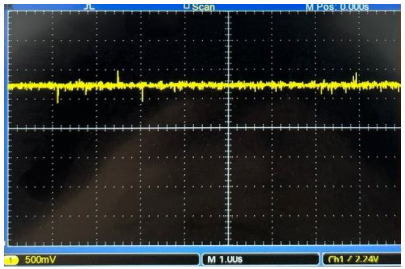




no capacitor



50 pF capacitor



250 pF capacitor

Bigger capacitors slow down the response time

Not Good Enough

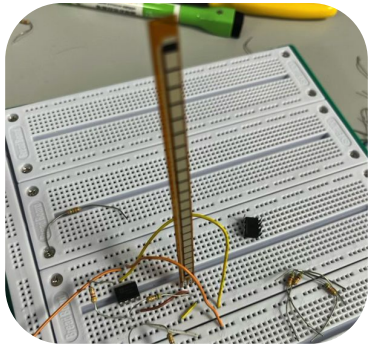
Conditioning Circuit Experiments

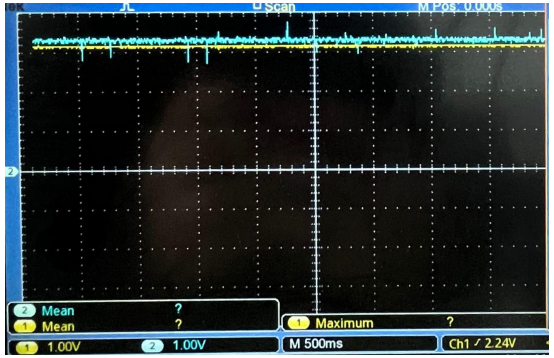
Next: wheatstone bridge circuit

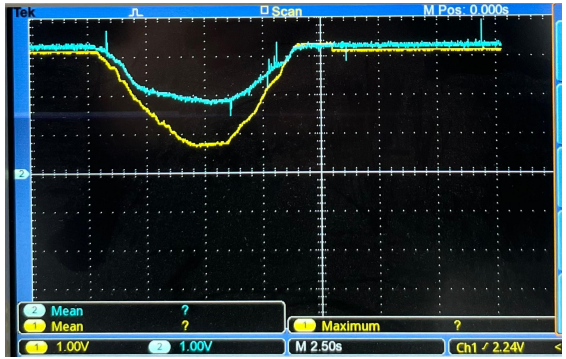
Not Good Enough

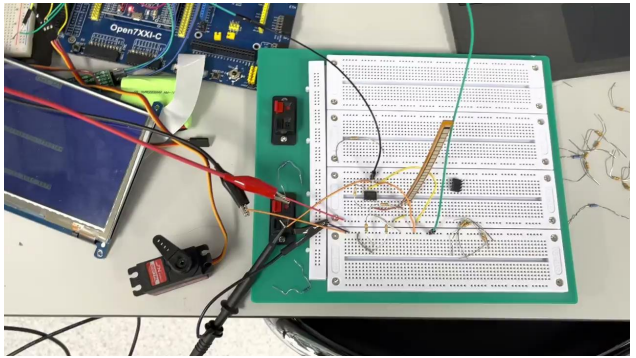
Finally: wheatstone + instrumentation amplifier

Good



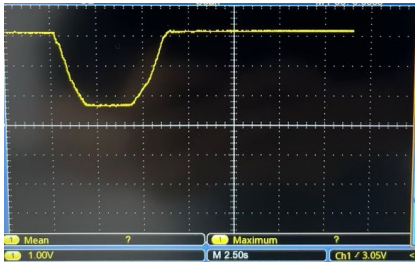




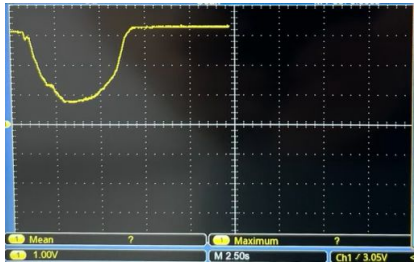


Tuning R and C Values

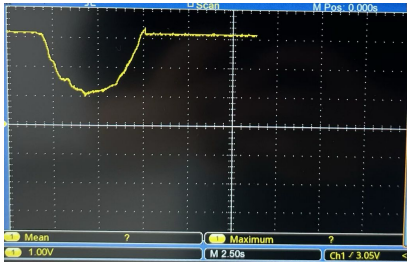
Long resistor



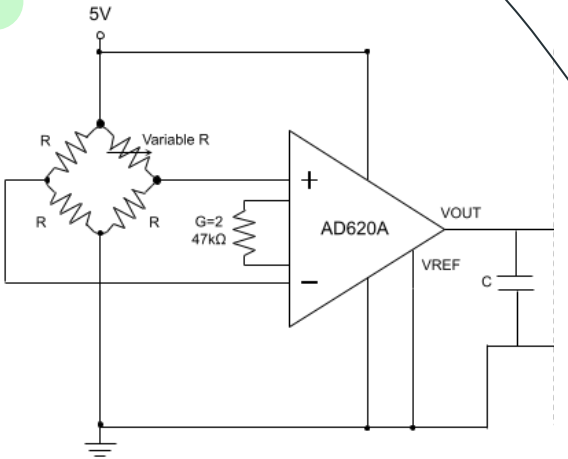
51 kΩ resistors

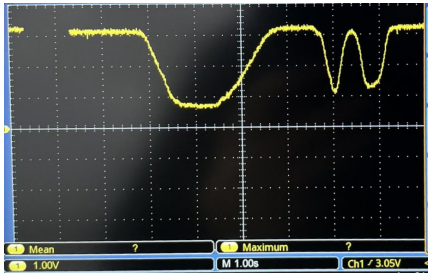


56 kΩ resistors

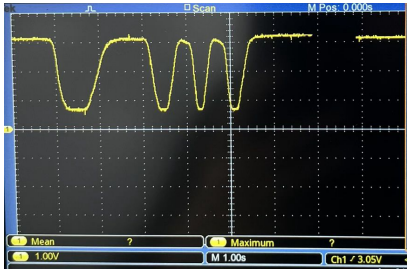


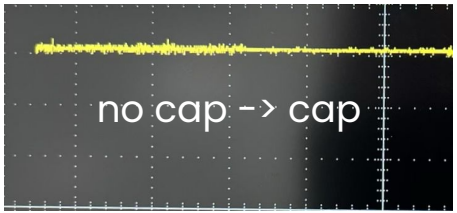
61 kΩ resistors





no capacitors added





200pF capacitor added at the output voltage

$$G = \frac{49.4k\Omega}{R_G} + 1 \Rightarrow G = \frac{49.4k\Omega}{47k\Omega} + 1 = 2.05$$

$$V_R^1 = 5\frac{56k\Omega}{56k\Omega+10k\Omega} = 4.24V$$

$$V_L^1 = 5\frac{56k\Omega}{56k\Omega+56k\Omega} = 2.5V$$

$$A(V_R^1 - V_L^1) = 2.05(4.24V - 2.5V) = 3.567V$$

RTOS Design

Initially: Polling method

Later: Threads + Event Flags

Every rising and falling edge:

$$f_{measurement} = \frac{f_{CLK}}{(P+1)(ARR+1)} = \frac{108\ MHz}{(15+1)(59999+1)} = 112.5\ Hz$$

Any faster than that this, ADC conversion rate cannot keep up with the speed of the timer that initiates the conversion.

Software Filter

Choices:

- Removing LSB
- Removing second LSB as well (limit potential values to (2500-500)/100 = 20)
- Moving average filter
- Threshold - 0 if within threshold

In addition:

- Check if all values are 0, if so, don’t send

PCB Design

Challenges:

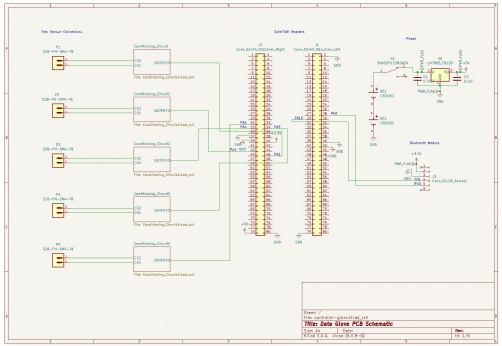
- Inability to power from the coin batteries (second largest capacity)
- Bluetooth module not powered from 3V3 pin of MCU

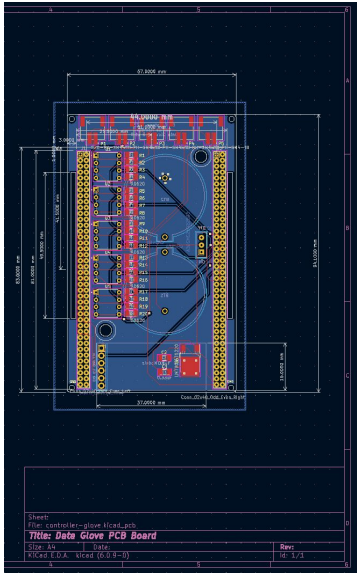
Cause:

- Coin battery not enough to power the overclocked MCU
- 5V voltage regulator recommended input voltage 7V instead of 6V

Solution:

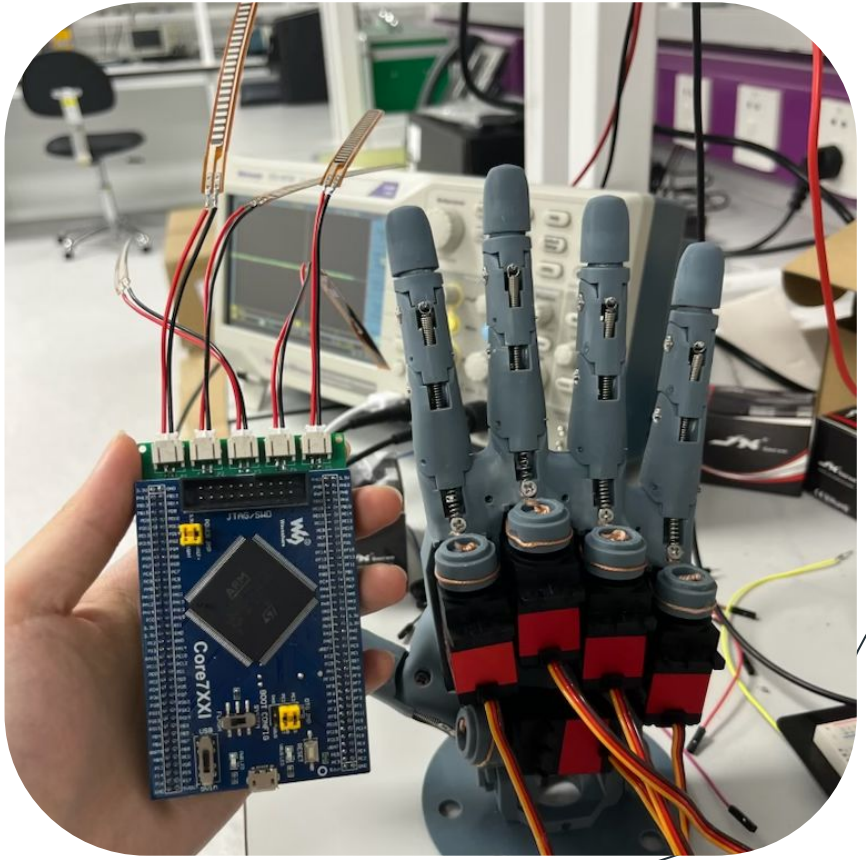
- USB input and rerouting manually



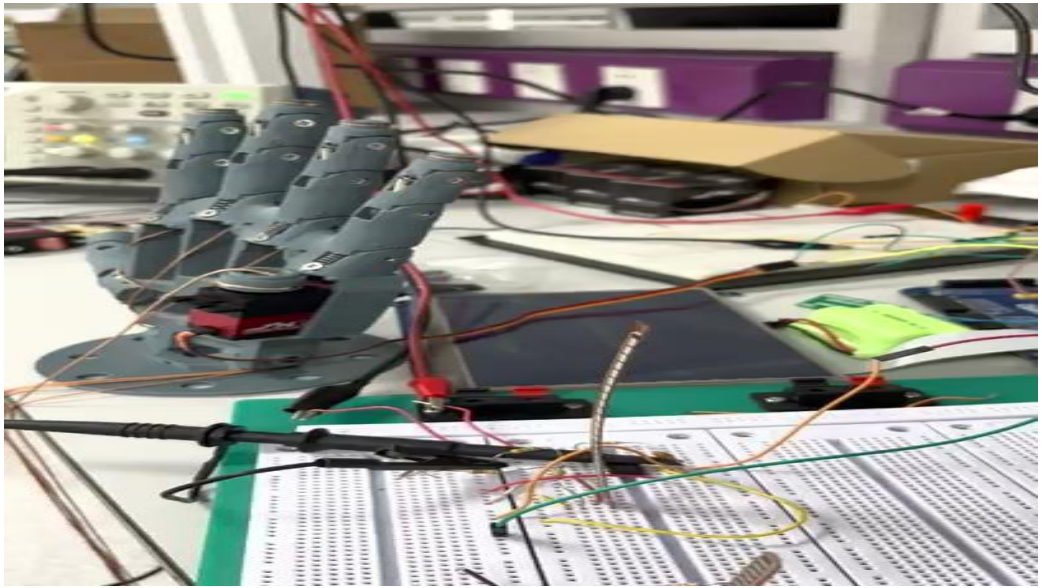


Results

It’s finally working!

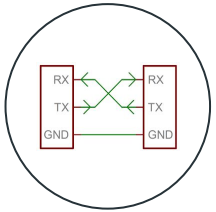


Despite challenges



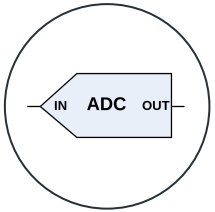
After assembling, last step was to tune the angles of my finger to the angle of the robot fingers. Especially for the thumb where the range of motion is much smaller.

Bottlenecks



UART Baud Rate

Limited the overall data transfer rate to 117 bps



ADC Conversion Rate

Limited the sampling of the flex sensors to 112.5 Hz

Limitations & Future Improvements

1. Addressing the bottlenecks

Data transfer rate can be increased by adjusting the baud rate of both BLE communication and UART interface. Sampling rate of the flex sensors can potentially be increased by using the interleaving mode, allowing faster ADC conversion by using two ADCs on the same input channel

2. 3D printing

Further refinement of the 3D printing parts can ensure that every part is correctly sized to accommodate the servo motors and printed motor gear parts (Incorrect sizing of parts)

3. PCB design

Design a smaller PCB that can directly fit onto the glove, using the STM32F7 chip instead of the development board. Focus on power delivery – higher capacity, small, at least 7V (E.g. 7.4V Li-ion batteries)

Limitations & Future Improvements

4. Software

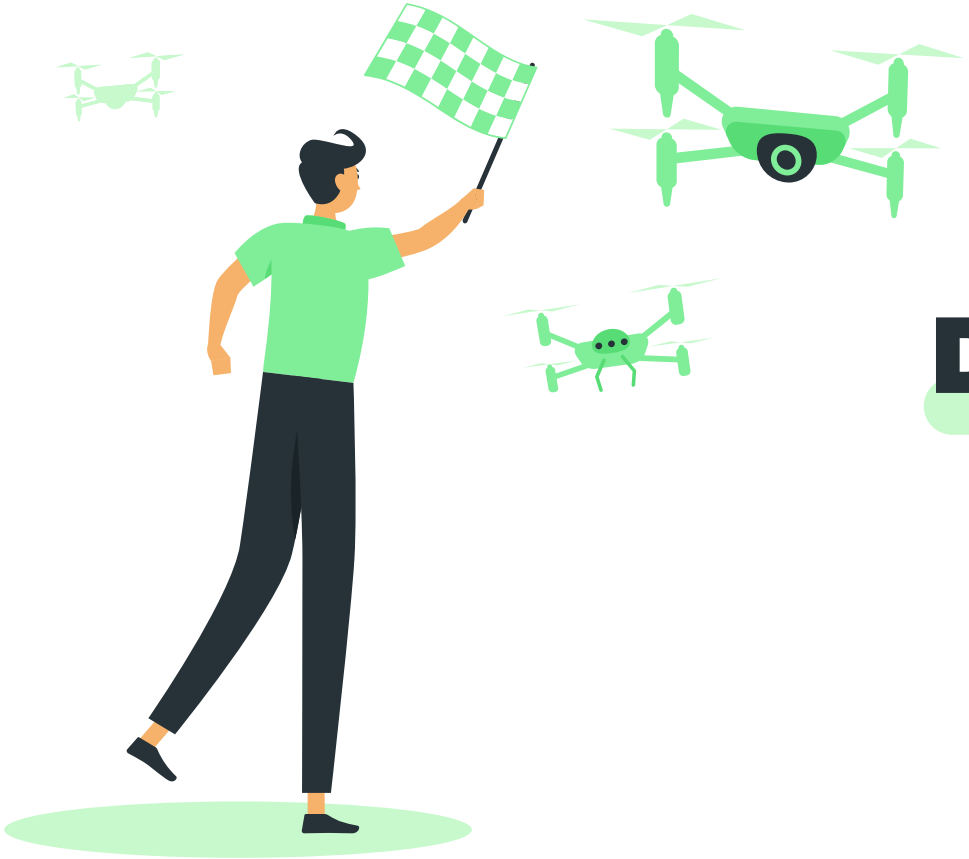
Improvements in interrupt handlers to reduce latency
Decrease power consumption and alleviate the strain on the batteries – low power mode when not in use, or slowing down the system clocks and gating the clocks to the APBx and AHBx peripherals when they are not in use (cost-benefit analysis to determine the most appropriate approach)

5. Signal processing

Inconsistencies in voltage levels are observed (voltage level is unstable and slightly increases when the flex sensor remains bent at a certain angle, making the signal indeterminate)
Choice of instrumentation amplifier

6. Mismatch between the size and proportions of the robotic hand and my hand

Out of scope for this project as this requires modifying the 3D model significantly



Demo Time!

Thank you to everyone who has helped me along the way. Professor Ding for helping me edit my paper, and special thanks to Professor Corcolle for helping me so much with the 3D printing, teaching me how to use RTOS, figuring out circuit issues, and answering my countless questions.

Thank you!