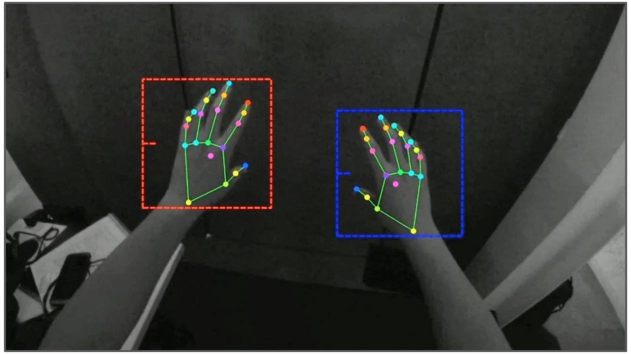
Wireless EMG Sleeve for Hand Gesture Recognition

Team 57

Problem

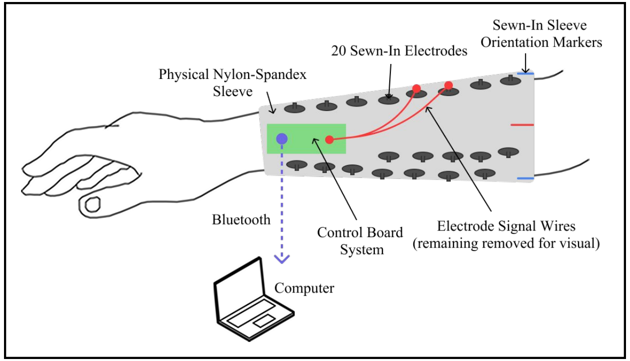
Virtual reality (VR) environments are not immersive enough leading some users to experience cybersickness (characterized by discomfort). Part of immersion is related to how VR systems handle hand interactions.

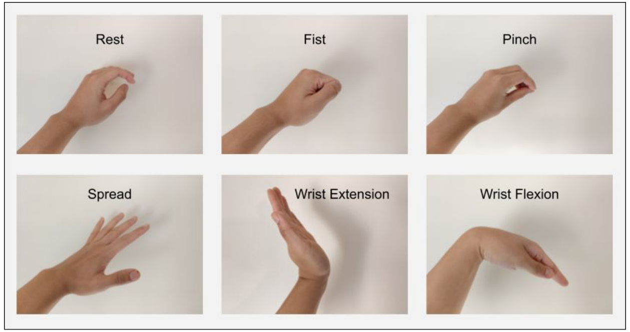
Modern VR Controllers



Computer Vision Hand Tracking

Wireless EMG Sleeve for Hand Gesture Recognition

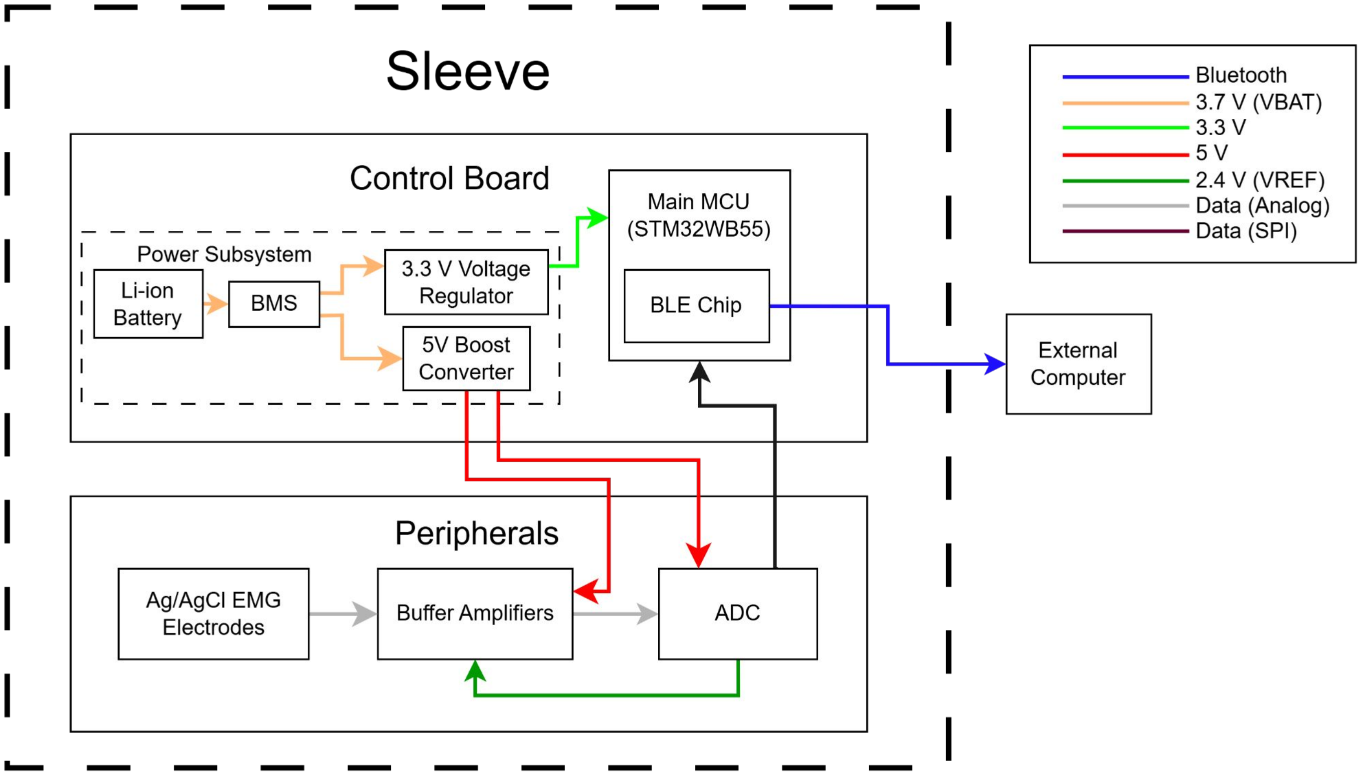




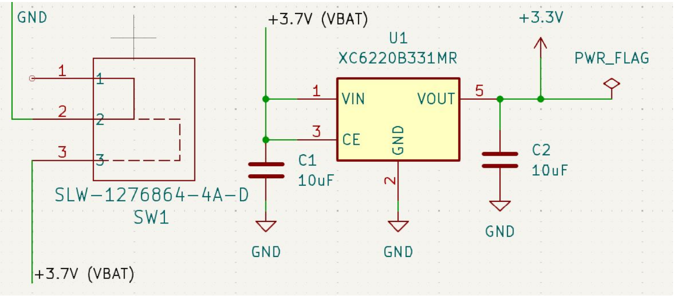
High-Level Requirements

1. Reliability in Discerning Gesture
   Verification: Achieve an 70% accuracy in correct classification between the 6 gestures over 30 random user tests.
2. Unrestrained Device Use
   Verification: Classification result from each user test sent wirelessly between our device and an external computer from a distance greater than 10 meters.
3. System Operational Time
   Verification: Ensure the device can be used for 1 hour without need of a battery recharge.

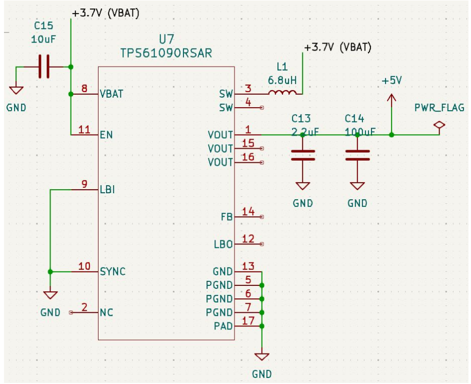
Block Diagram



Power Subsystem

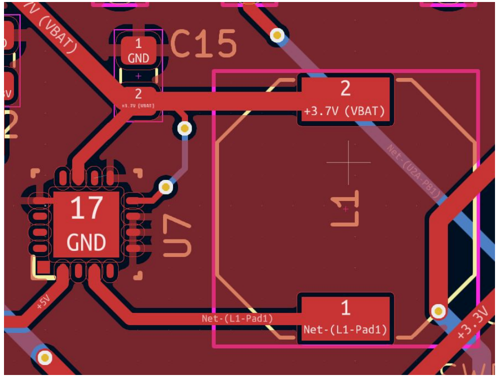


Switch and 3.3 V Regulator



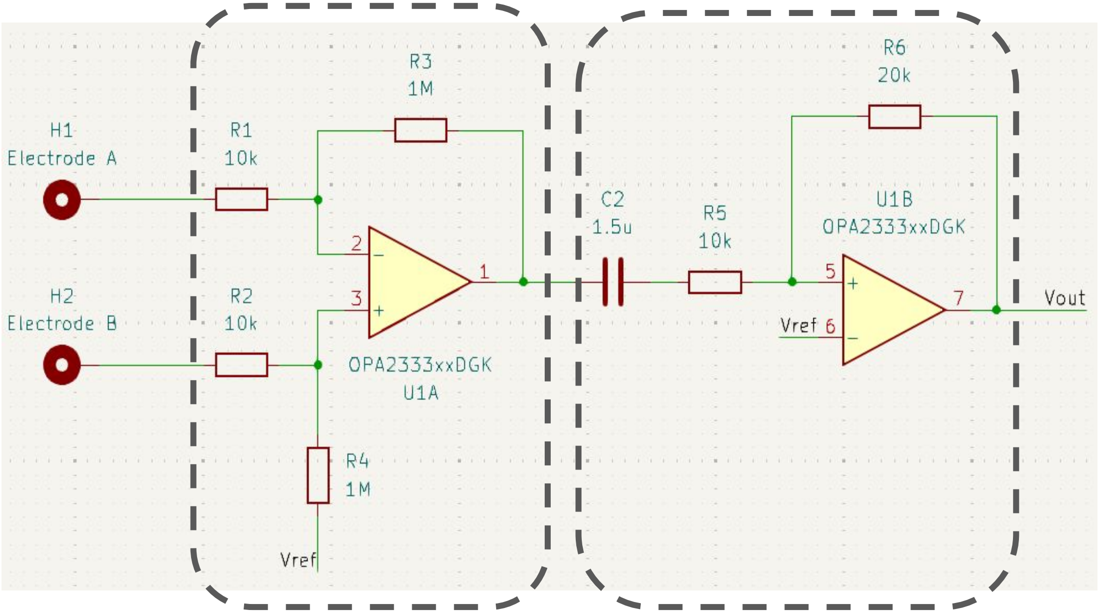
5 V Boost Converter

Requirement	Verification
There must be switch to remove power from the device during operation.	1. Use a precision voltmeter to measure the voltage across the voltage regulator output to universal ground. 2. Assure voltage measured is 3.3V within 0.3V 3. Turn off the power switch, repeat step one, assure there is no voltage difference.
The external battery must be able to supply a constant 3.7V within 0.3V.	1. Use a precision voltmeter to measure the voltage across the battery
The XC6220 Voltage Regulator must reliably regulate the battery voltage to 3.3V within 0.3V.	1. Use a precision voltmeter to measure the voltage across the voltage regulator to universal ground.
The circuit associated with the TPS61090 must reliably regulate the battery voltage to 5V within 0.2V.	1. Use a precision voltmeter to measure the voltage across the voltage booster to universal ground.



Poor layout of inductor, capacitor and 5 V boost converter

EMG Subsystem - Preamp board



Differential Amp
(Gain = 100)

High Pass Filter
(Gain = 2, Cutoff = 10.6 Hz)

Gain:

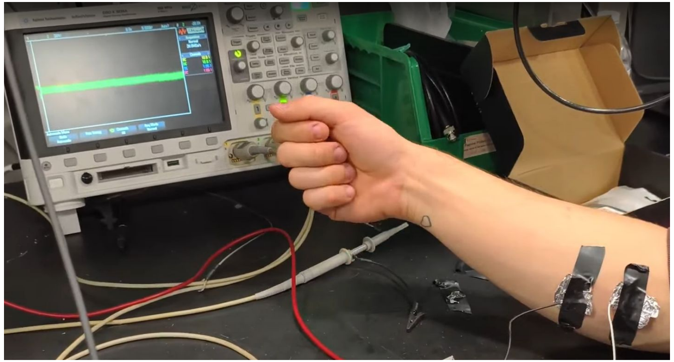
- Expected sEMG: **±10 mV range**.
- Desired range: 0 V – 5 V (Vref=2.4 V)
- Max gain: 2.4 V / 10 mV = 240
- Chose gain of **200**

HPF Cutoff:

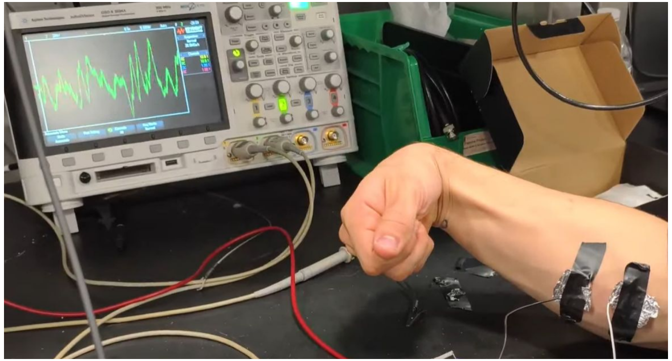
- EMG frequencies higher than 10 Hz
- Chose low cutoff of 10.6 Hz

EMG Subsystem - Sensing Board Test

Rest

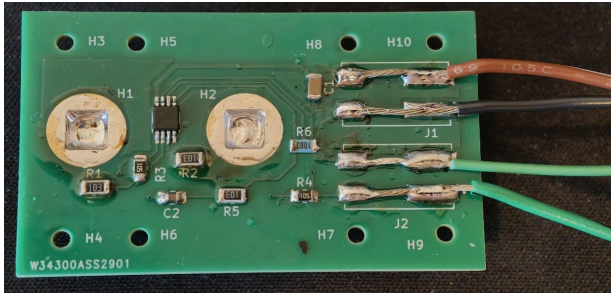


Flexion



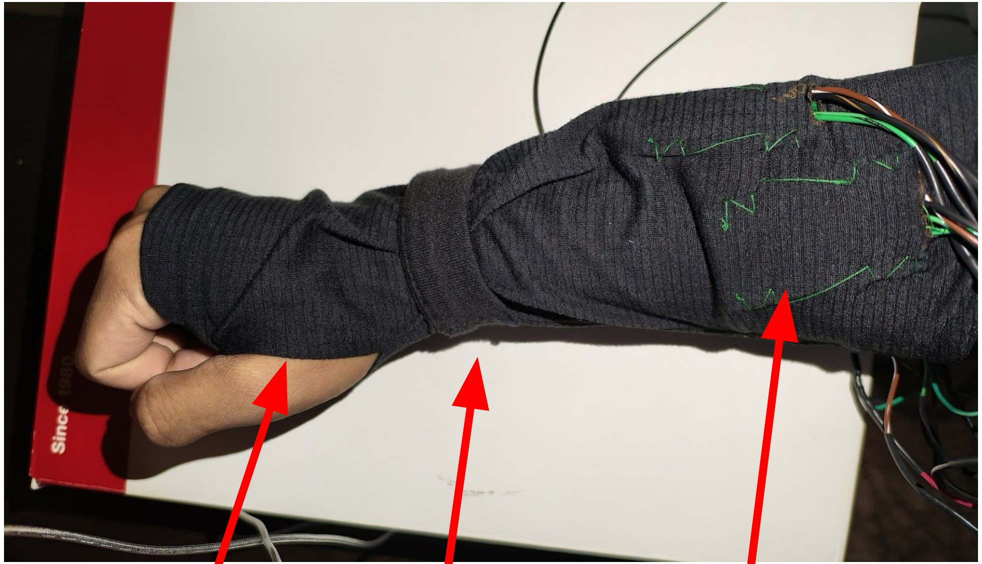
Requirement	Verification
The EMG array must be able to detect muscle stimulation signals in the 10 mV range.	1. Use a signal generator to input known mV-range signals into the EMG array (1 or 2 mV for example). 2. Analyze the output of the EMG array using an oscilloscope. 3. Compare the input and output signals to verify accurate detection. 4. Repeat the test for various signal amplitudes within the mV range. 5. Document the results, including any discrepancies between input and output signals.
The amplifier circuit must amplify mV range signals with a gain of 200 within 10 mV.	1. Use a signal generator to input known mV-range signals into the EMG array (1 or 2 mV for example) 2. Analyze the output of the EMG array using an oscilloscope. 3. Compare the input and output signals to assure a gain of 200. 4. Repeat the test for various signal amplitudes within the mV range.

Sleeve Subsystem - EMG Sensor Placement



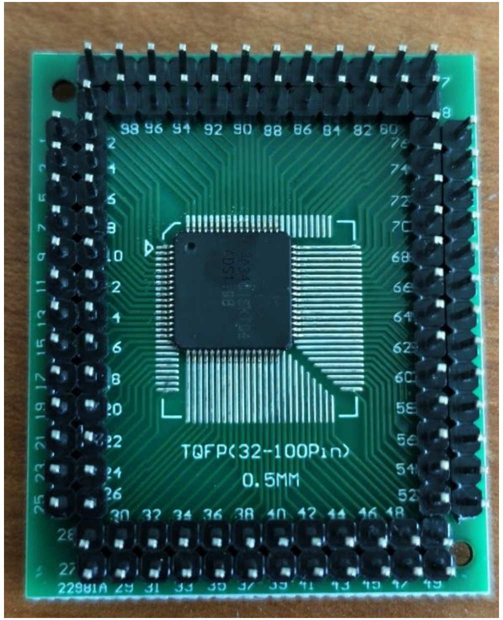
EMG Preamp Board

EMG Arm Sleeve

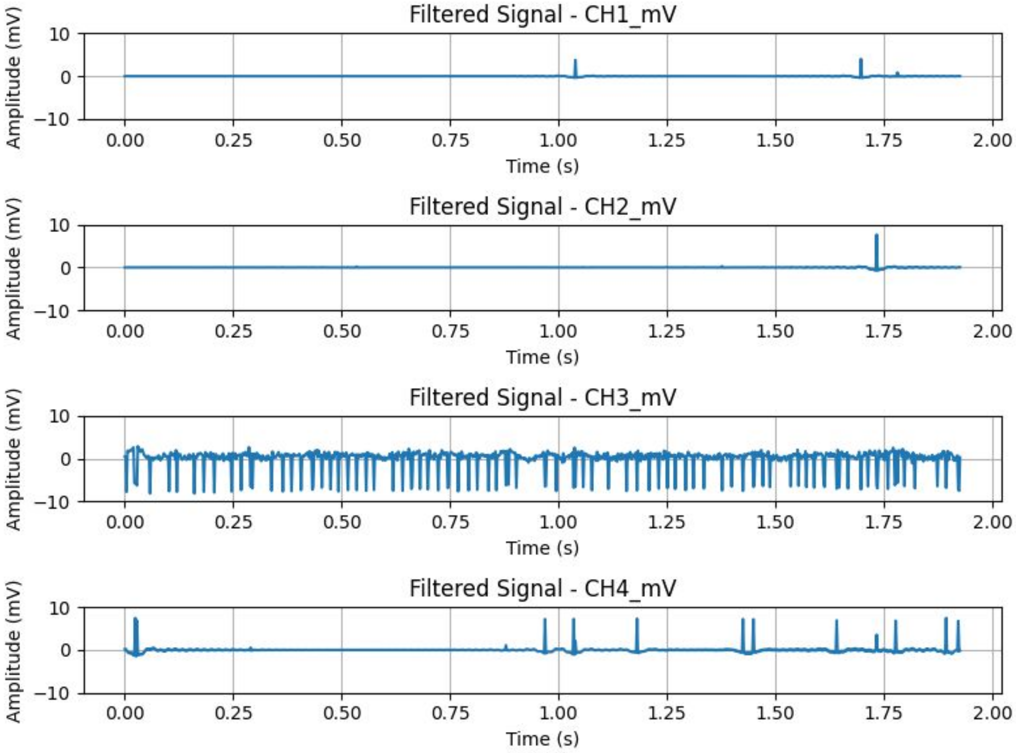


Requirement	Verification
The sleeve diameter when maximally stretched must be ≤ 10 cm.	1. Use a flexible measuring tape to measure the sleeve's diameter when fully stretched. 2. Take measurements at multiple points along the sleeve's length. 3. Record all measurements and ensure they are ≤ 10 cm. 4. Document the results in a table, including the maximum measured diameter.
The sleeve must allow for consistent EMG sensor placement between uses.	1. Put on the EMG Sleeve 2. Use a pen to mark sleeve placement using the sleeve holes 3. Take off, and put on sleeve once more 4. Assure that previous pen marks align with sleeve holes.

Control Board Subsystem - ADC



ADC on breakout board



Filtered EMG data for wrist flexion

Processing Subsystem

timestamp	CH1_mV	CH2_mV	CH3_mV	CH4_mV	gesture
1745896227	0	6.226	0.732	0	1
1745896227	6.958	17.579	0	1.392	1
1745896227	0	3.369	115.213	0.513	1
1745896227	6.592	4.657	0	1.685	1
1745896227	6.738	4.541	-0.22	0	1

Requirement	Verification Procedure
The STM32WB55MMG must process EMG and IMU sensor data with a latency of 100ms.	1. Setup: Simulate continuous EMG and IMU sensor data being fed into the STM32WB55MMG MCU. 2. Logging Execution Time: Use built-in timers to record timestamps (a) when data is received, (b) when classification is completed. 3. Pass Criteria: Ensure the total latency (time from receiving sensor data to classification result output) is $\leq$100ms in 95% of cases.
The STM32WB55MMG must execute TensorFlow Lite models efficiently for real-time gesture inference.	1. Setup: Convert a pre-trained gesture recognition model into TensorFlow Lite format optimized for the MCU. 2. Test Execution Time: Run inference on test data and log the duration using the MCU's internal clock. 3. Pass Criteria: Confirm that inference time is $\leq$ 50ms per sample while maintaining classification accuracy $\geq$80%.
The system must maintain high-speed SPI communication with the EMG/IMU sensors at $\geq$1 Mbps.	1. Setup: Send a fixed-size data packet (e.g., 100 bytes) from the sensors to the STM32WB55MMG over SPI. 2. Measure Throughput: Use a logic analyzer or timestamp comparison to measure SPI transfer speed. 3. Pass Criteria: Ensure the transfer rate is $\geq$1 Mbps with an error rate of $\leq$1% over 1000 transfers.
The STM32WB55MMG must correctly classify gestures with at least 80% accuracy.	1. Setup: Record 30 user-performed gestures and log classification outputs. 2. Compare with Ground Truth: Match each classification result against manually labeled gestures. 3. Pass Criteria: Ensure $\geq$80% of gestures are correctly classified.
The processing system must remain operational without crashes or memory leaks over a 1-hour continuous test.	1. Setup: Run the gesture recognition system continuously for 1 hour with simulated or live sensor data. 2. Monitor Stability: Check for MCU resets, memory allocation failures, or performance degradation. 3. Pass Criteria: No unintended reboots, crashes, or increasing memory usage over the full test period.

Processing Subsystem: EMG Signal Classification Model

Layer Type	Neurons	Activation	Normalization	Dropout
Input Layer	40	-	-	-
Dense Layer 1	50	ReLU	BatchNorm	-
Dropout Layer 1	-	-	-	0.2
Dense Layer 2	50	ReLU	BatchNorm	-
Dropout Layer 2	-	-	-	0.2
Output Layer	6	Softmax	-	-

Dense layer 1: Learns initial high-level representations from raw features
Dense layer 2: Further abstracts input

Processing Subsystem: Model Training Methodology

Dataset Structure:

- 30 trials per gesture class, 2 seconds each
- Sampling Rate: 2khz

Training Split:

- Stratified 70%/30% Train-Test Split
- Early Stopping (monitoring Validation loss)
- Trained 100 epochs

Feature extraction:

Mean: Baseline offset or average contraction

Standard Deviation: Signal fluctuation/spread

RMS: Signal power

MAV: Overall activity level (smoothed abs value)

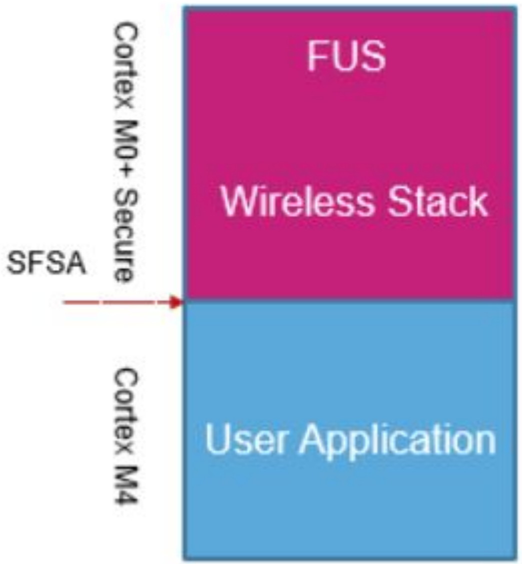
Zero-Crossings: Oscillation frequency — helps detect movement types

Processing Subsystem: Classification Results

Class	Precision	Recall	F1-Score	Support
Extension	1.00	0.76	0.87	17
Fist	1.00	0.85	0.92	20
Rest	1.00	1.00	1.00	21
Pinch	0.74	0.81	0.77	21
Flexion	1.00	1.00	1.00	18
Two	0.74	0.94	0.83	18
Accuracy			0.90	115

- Precision: Correct positive predictions / total predicted positives
- Recall: Correct positive predictions / total actual positives
- F1-Score: Harmonic mean of precision and recall
- Support: Number of true samples for each class

Bluetooth Subsystem



STM32WB series uses a dual-core architecture:

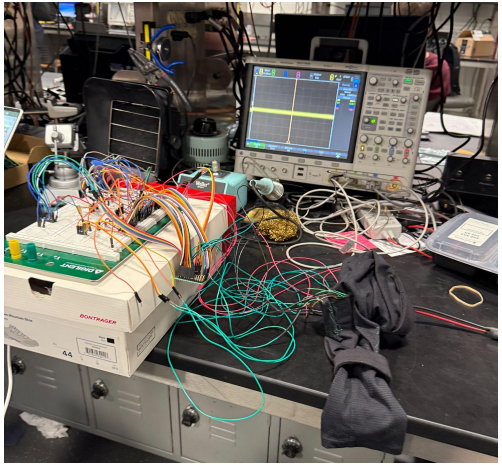
CPU1: Arm® Cortex®-M4 → runs our application code
communicates via IPCC
CPU2: Arm® Cortex®-M0+ → runs encrypted BLE stack

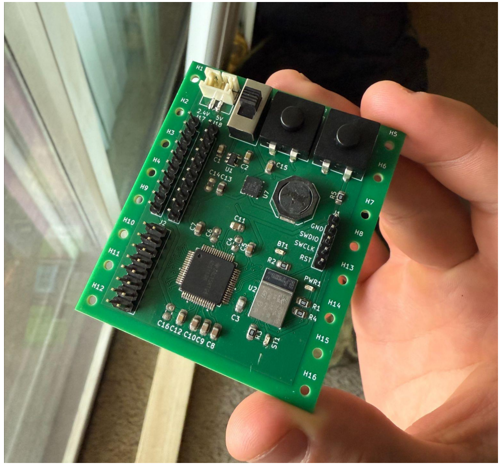
STM32WB chips come preloaded with a root security firmware (FUS)

- Entered APP_BLE_Init()
- SHCI_C2_BLE_Init() crashes (system halt)
- problem with the BLE stack binary on CPU2

Future Work

- Handling Noisy Signals
  - Analog Filters, Perfecting the PCB
- Streamlining Classification
  - More Electrodes, Realtime Classification through Bluetooth Protocol
- Standalone Device
  - Testing on multiple users
  - Designing VR Environment for Practical device testing





Wireless EMG Sleeve for Hand Gesture Recognition

Team 57