ParaPedal:
A Practical Product for Paraplegic Piano Players!

Group 15 – Senior Design

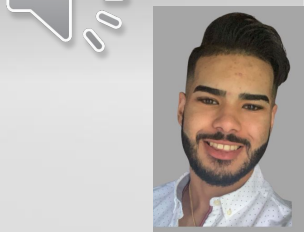
University of Central Florida

Team Members

- Andre llanos(Electrical Engineer)
- Benedict Eberhagen (Electrical Engineer)
- Jason Knaebel(Electrical Engineer)
- Rafael Molina (Electrical Engineer)

Motivation & Background

- Inspired by teammate Benedict, a pianist in a wheelchair
- Problem: Cannot use legs to press piano pedals
- Few assistive devices exist; bulky and limited



Goals

Basic: Sustain pedal, intuitive use, no piano modification

Advanced: 3 pedals, calibration via smartphone app, quiet actuator

Stretch: Breath/muscle input, haptic feedback

Objectives

Basic: Sustain pedal, intuitive use, no piano modification

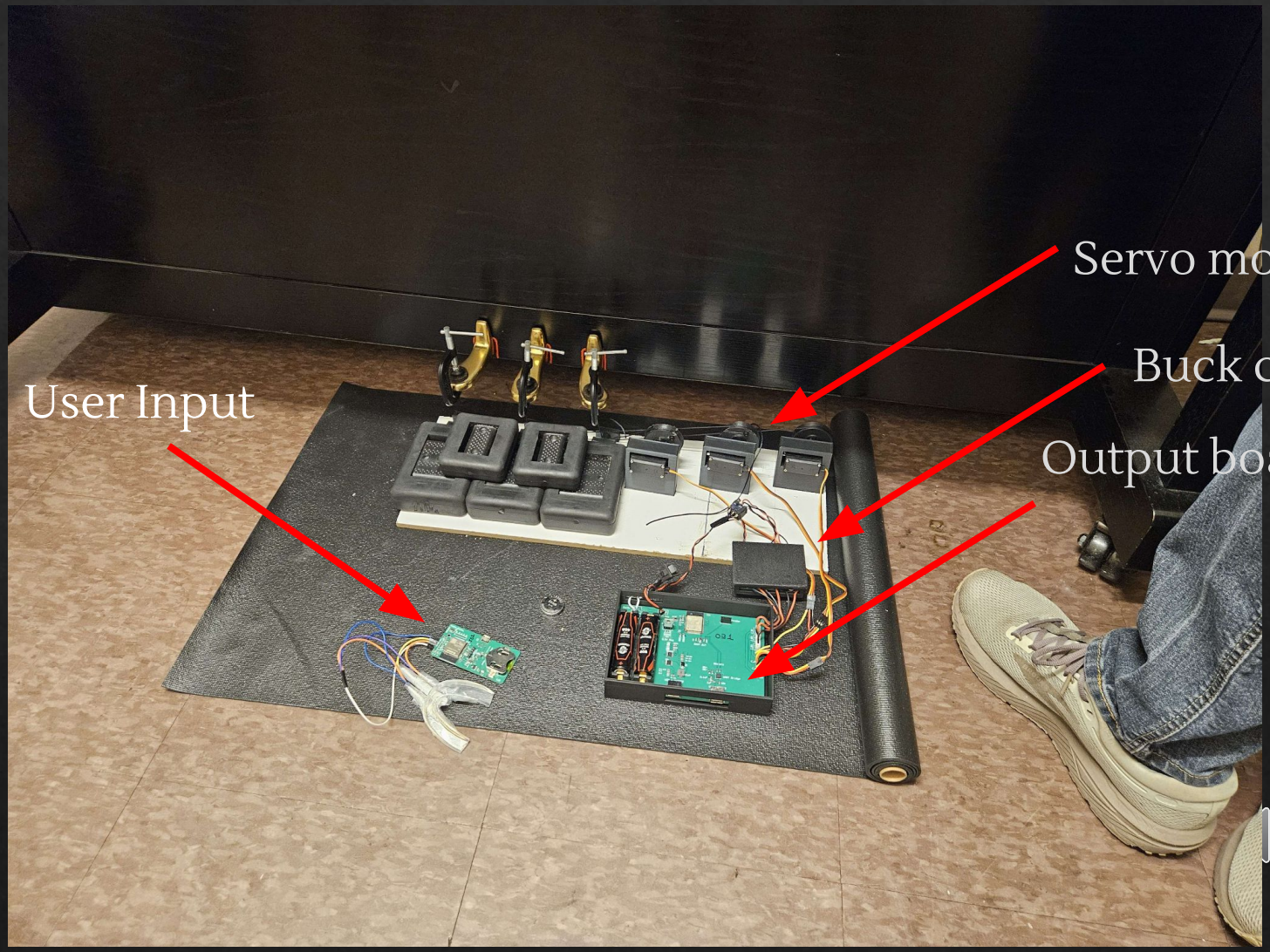
Advanced: 3 pedals, calibration via smartphone app, quiet actuator

Stretch: Breath/muscle input, haptic feedback

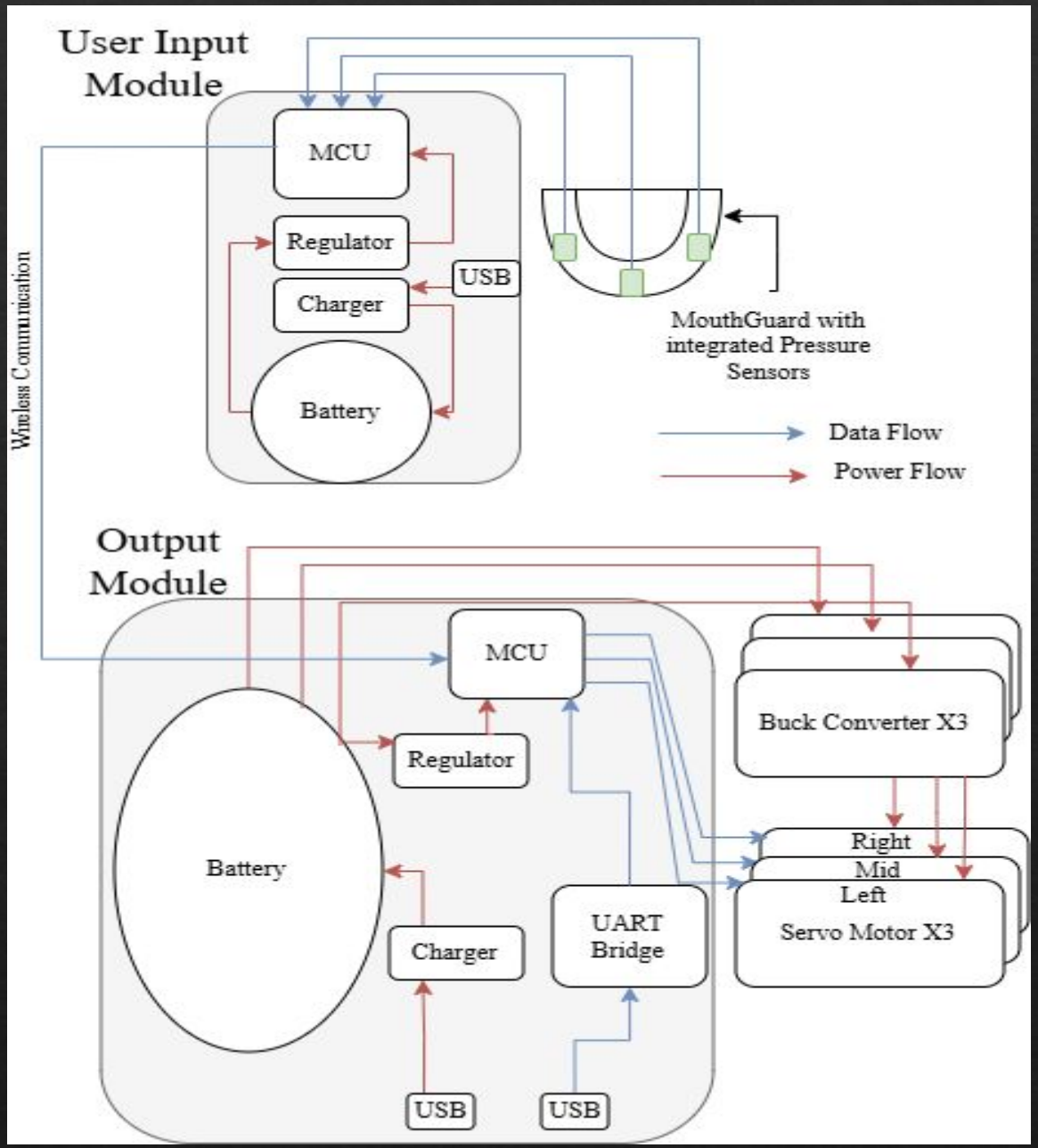
Engineering Specifications

Specification Table	
Motor System	
Motor Response Time	500-700 Milliseconds
Maximum Power Consumption	≤ 25 Watts
System Overall	
Device Weight	5-10 Kilograms
Battery Runtime	≤ 1 Hour
Communication Range	≥ 4 meters
Response Time	≤2 Second
Accuracy	75-85%
System Setup Time	≤ 2 Minutes

High-Level System Overview



System Diagram



Hardware Selection Process

Categories: Mouthpiece, Motor, MCU and Power

Compared multiple technologies for each

Chose final parts based on performance, cost, usability

Mouthpiece & Sensor

OPTION	PROS	CONS	FINAL CHOICE
BBTO Sports Guard	Cheap, easy to modify	Bulky, not discreet	❌
Goyo Night Guard	Comfortable, discreet, sleek	More expensive	✅ Final Mouthpiece
Capacitive Sensor	Non-contact sensing	Moisture sensitive, complex integration	❌
RP-S5 Force Sensor	Biocompatible, compact, simple analog signal, 1–10 N range	Limited range	✅ Final Sensor

- **Final decision for Mouthguard: Goyo Night Guard**
- **Final decision for Sensor: RP-S5 Force Sensor**

Actuator vs Motor

Option	Pros	Cons	Final Choice
Linear Actuator (20N type)	Direct linear motion	Bulky, slow, noisy, higher cost	❌
Linear Actuator (VEVOR 1500N)	Very high force capability	Overkill, heavy, slow, expensive	❌
Servo Motor – MG995	PWM control, decent torque	Lower torque/precision than MG996R	❌
Servo Motor – MG996R	13 kg·cm torque, ~0.17s/60°, compact & quiet	Limited rotation	✅

- **Final decision: Servo Motor MG996R**

Microcontroller

Option	Pros	Cons	Final Choice
Nordic	Good wireless performance	Limited I/O	❌
STM32	Powerful, flexible	Complex, higher learning curve	❌
ESP32	Wi-Fi + Bluetooth, ADCs, PWM, low power	None significant	✅

- **Final decision: Esp32**

Power Systems

User Input Module		
Major Component	**Current / Voltage**	**Power**
Pressure Sensor (RP-S5)	0.5 mA @ 3.3 V	1.65 mW
ESP32-C6 (Main Controller)	25 mA avg @ 3.3 V	82.5 mW
3.3 V Regulator Loss	Output 3.3 V @ 26 mA	10.4mW
Total (User Input)	**≈ 95mW**	

Output Module		
Major Component	**Current / Voltage**	**Power**
ESP32-WROOM (Main MCU)	100 mA @ 3.3 V	0.33 W
3.3 V Regulator – LD33V (LDO)	Input 8.0 V @ 100 mA → Output 3.3 V @ 100 mA	0.47 W(Loss)
Servo Motors (MG996R ×3) via 5 V buck(s)	5 V @ 1A (one servo at a time)	≈ 6.25 W (one servo at a time)
Battery Charger – CN3302 (charging)	-	≈Inactive during operation
Total		**≈ 10 W @ 8 V (≈ 3A)**

Batteries and Recharging Unit

Option	Pros	Cons	Final Choice
LIR2450 Coin Cell	Small, lightweight, rechargeable	Limited capacity	✅ (Mouthpiece Input Board)
2×18650 Pack	High current supply, good capacity	Bulkier, needs balancing	✅ (Motor Board)
Charger – TPB4056A20	Cheap, basic protection	Limited features, 1-cell only	❌
Charger – CN3302	USB input, 2-cell support, safe charging	Slightly higher cost	✅

Final decision:

- **Input and Output Battery: LIR2450 Coin Cell and 2×18650 Pack**
- **For Battery charging : CN3302**

Servo Motor Regulators

Option	Pros	Cons	Final Choice
MP9928GF-Z	**90% efficiency, handles motor spikes, ~3A+ to use**	**Large and more complex to implement**	❌
XC6204B502MR-G	**Flexible voltage options**	**Cannot supply motor currents (<1A)**	❌
MP2338	**>90% efficiency, up to ~3A**	**Near limits under motor transients**	❌
LM2596	**High efficiency, handles up to 3 A for motor loads, easy to adjust**	Bulkier module compared to an LDO	✅

- **Final decision: LM2596**

Input and Output Regulators

Option	Pros	Cons	Final Choice
MCP1700T-3302 (Input Board)	Ultra-low quiescent current (~1.6 µA); simple LDO; great for coin-cell battery life	Max 250 mA; not suitable for heavier peripherals	✅
LD33V (Output Board)	Up to ~800 mA; stable 3.3 V for ESP32 + display; easy to integrate	Higher quiescent current (~6 mA); less efficient than a buck	✅
AMS1117-3.3 (Alternative)	Cheap, widely available; up to 1 A	High dropout (~1.1 V) and runs hot under load	❌
XC6204B-series (Alternative)	Compact, low noise, simple BOM	Limited output current (<1 A) and protection features	❌

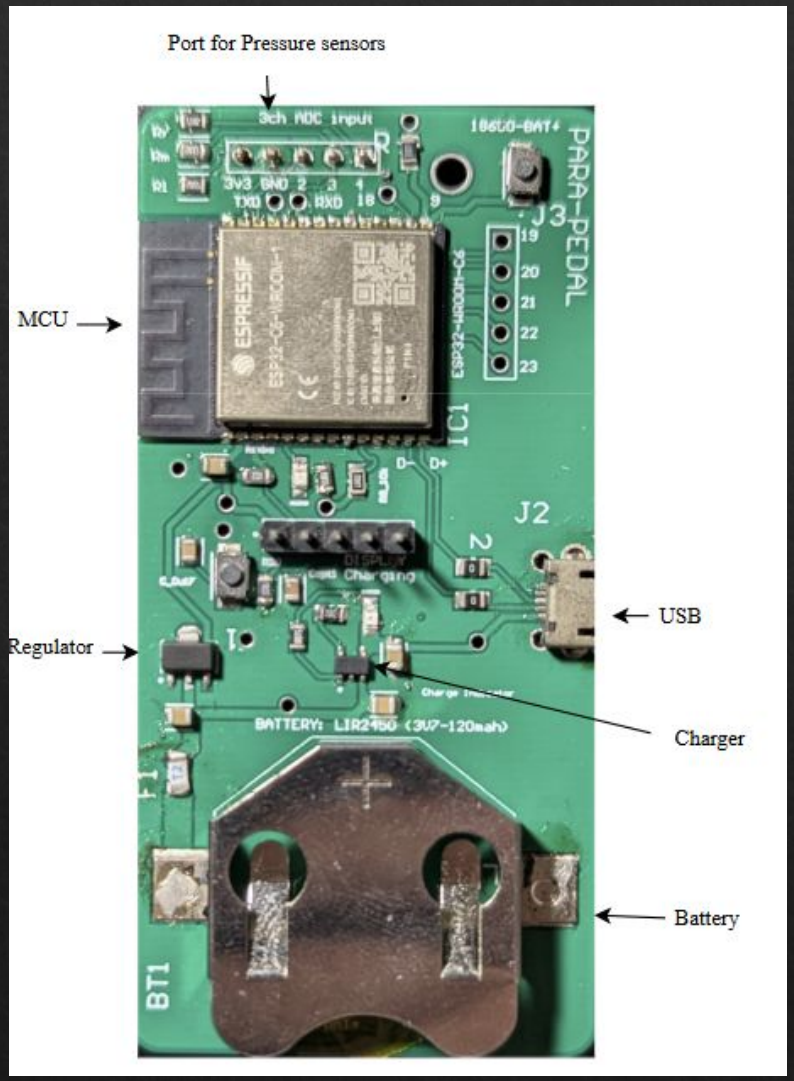
Final decision:

- **Input regulator: MCP1700T-3302**
- **Output Regulator:** LD33V

Final Hardware Summary

Subsystem	Final Choice	Why
Power	LIR2450 + 2×18650 + LM2596 + CN3302	Light, efficient, safe
Mouthpiece	Goyo Night Guard	Comfortable, sleek, hidden
Sensor	RP-S5 Force Sensor	Simple, reliable, safe
Actuator	MG996R Servo	High torque, fast, affordable
MCU	ESP32	Wi-Fi + Bluetooth, flexible, low power

User Input Board



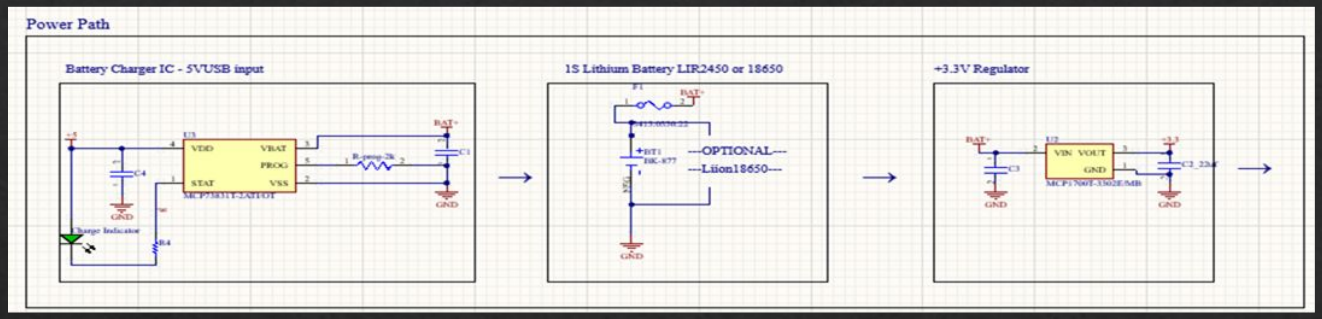
Primary Function

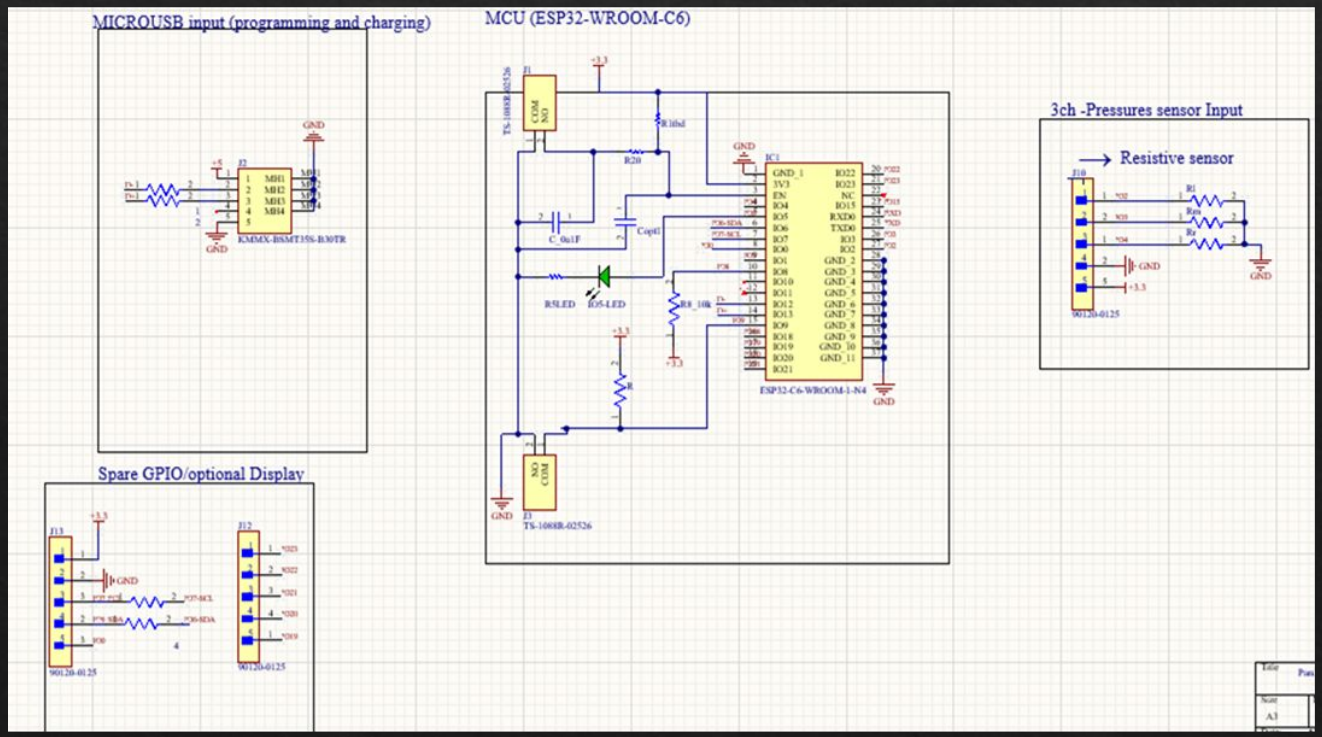
- To sense the user input
- Transmit this data to the output controller

Main Components

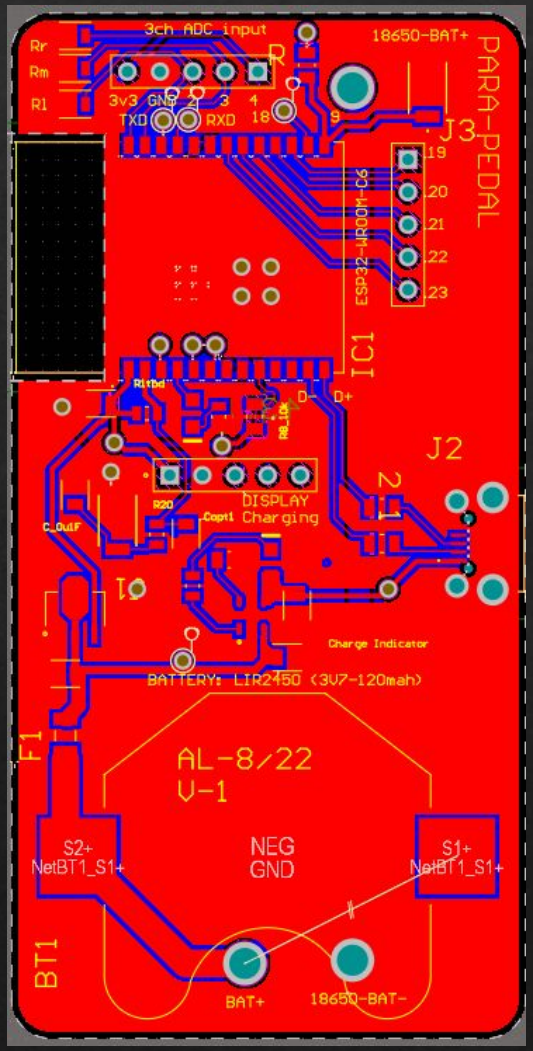
- MCU :ESP32 - C6
- Battery :LIR2450 120mah
- Charger: MCP73831T
- Regulator: MCP1700T

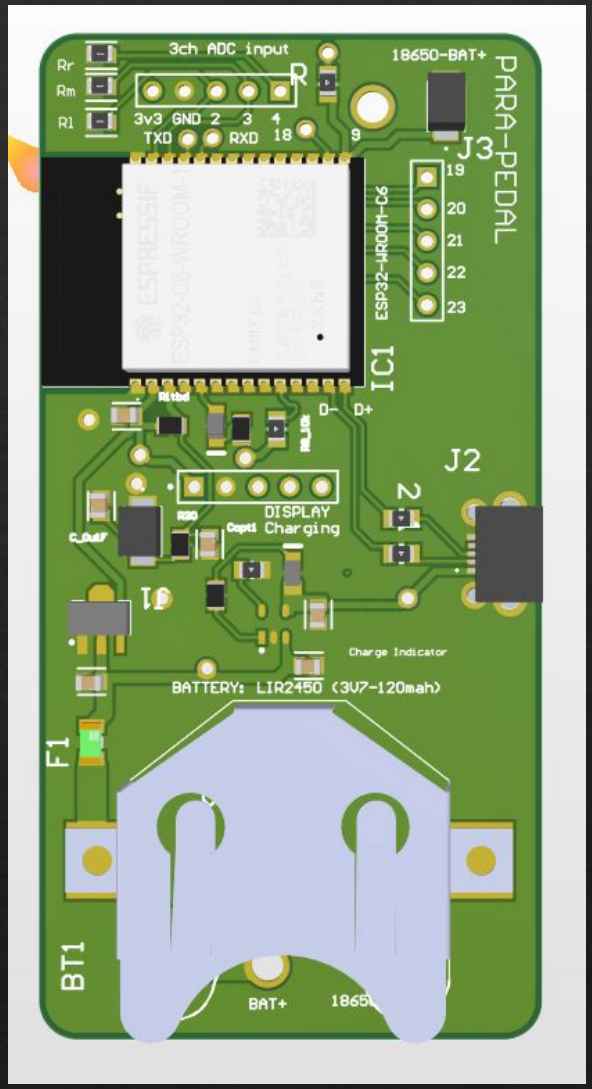
User Input Power Regulation and Control



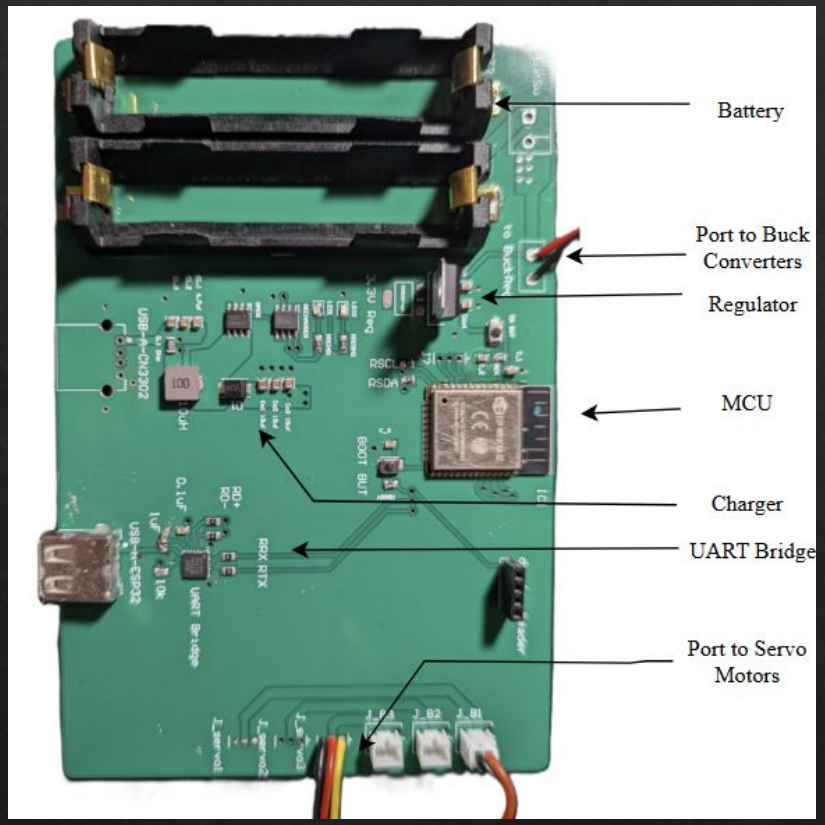


User Input Layout





Output Board



Primary function

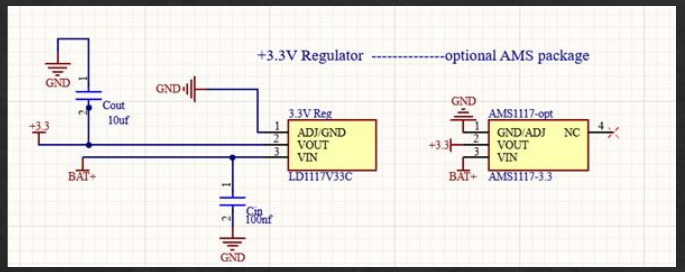
- to receive data wirelessly from input board
- Output PWM to the servo for actuation

Main Components

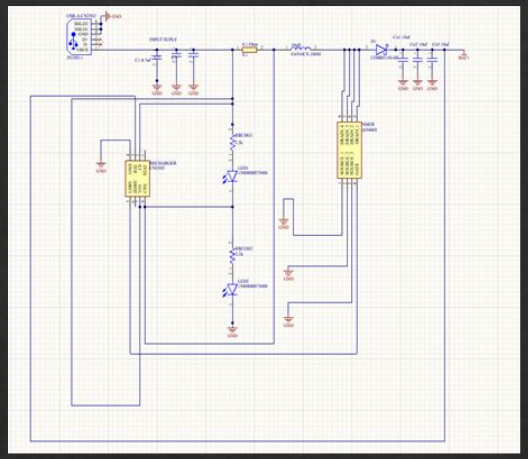
- MCU :ESP32 -32E
- Battery :2 X 18650 2200mah
- Charger: CN3302
- Regulator: LD-33V

Output board Power management and control

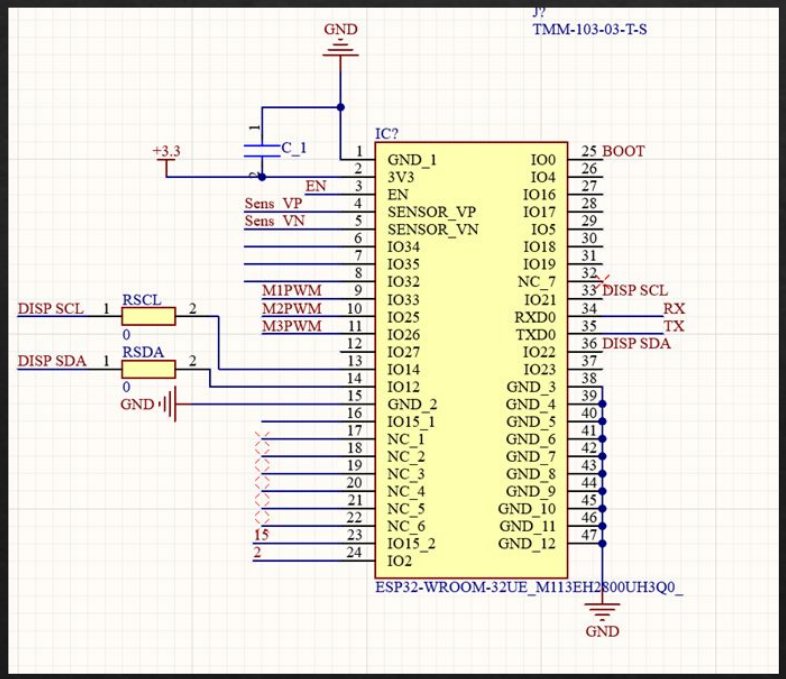
7.4V to 3.3V regulator



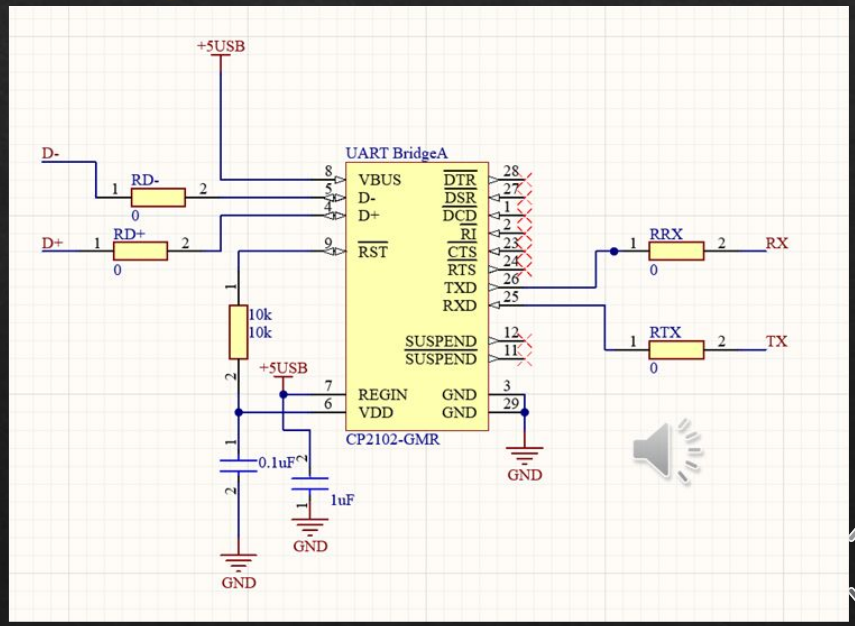
2S Lithium battery charger



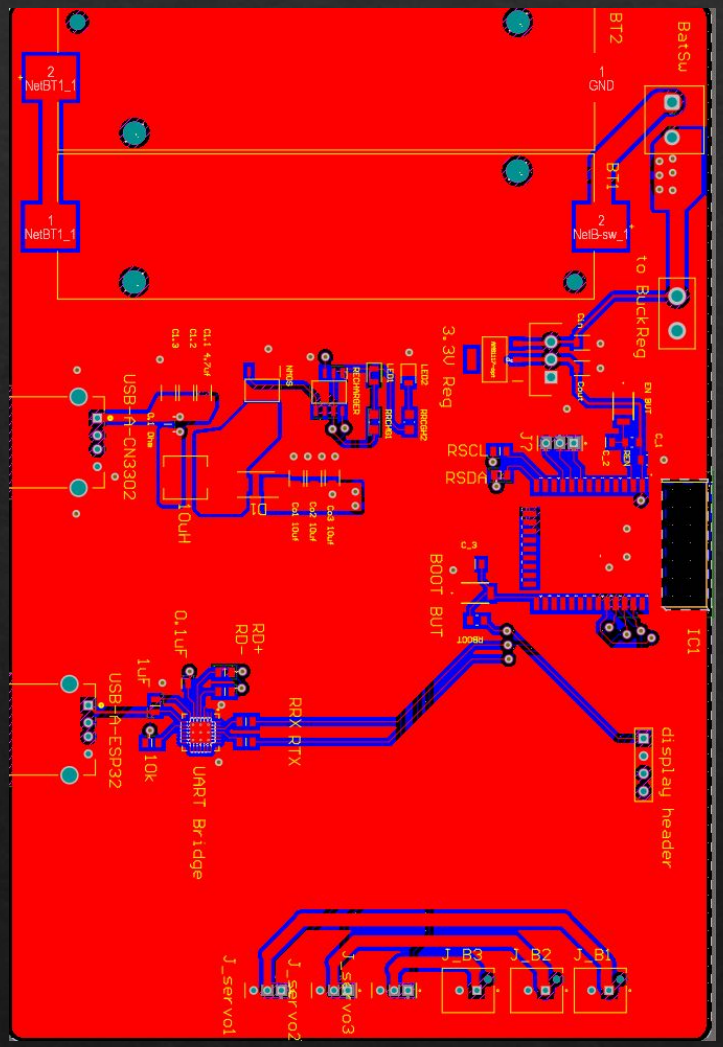
MCU: ESP 32 - WROOM

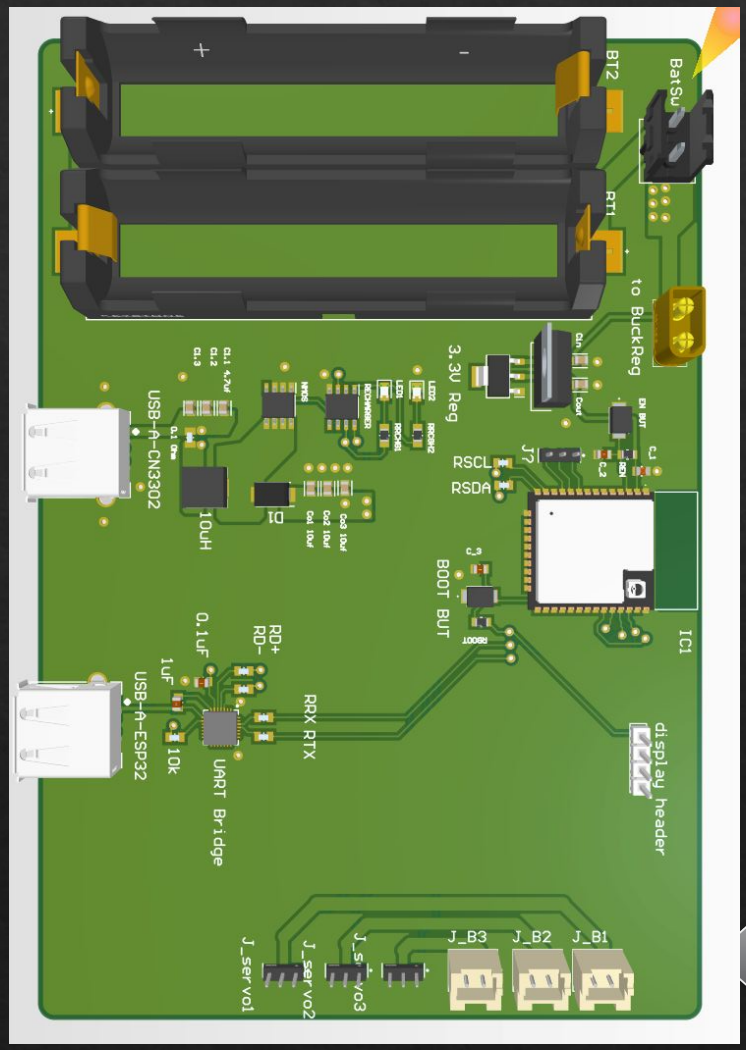


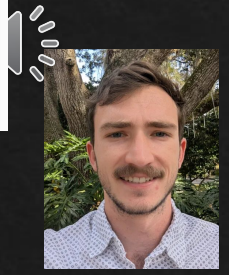
UART Bridge :CP2102



Output Layout







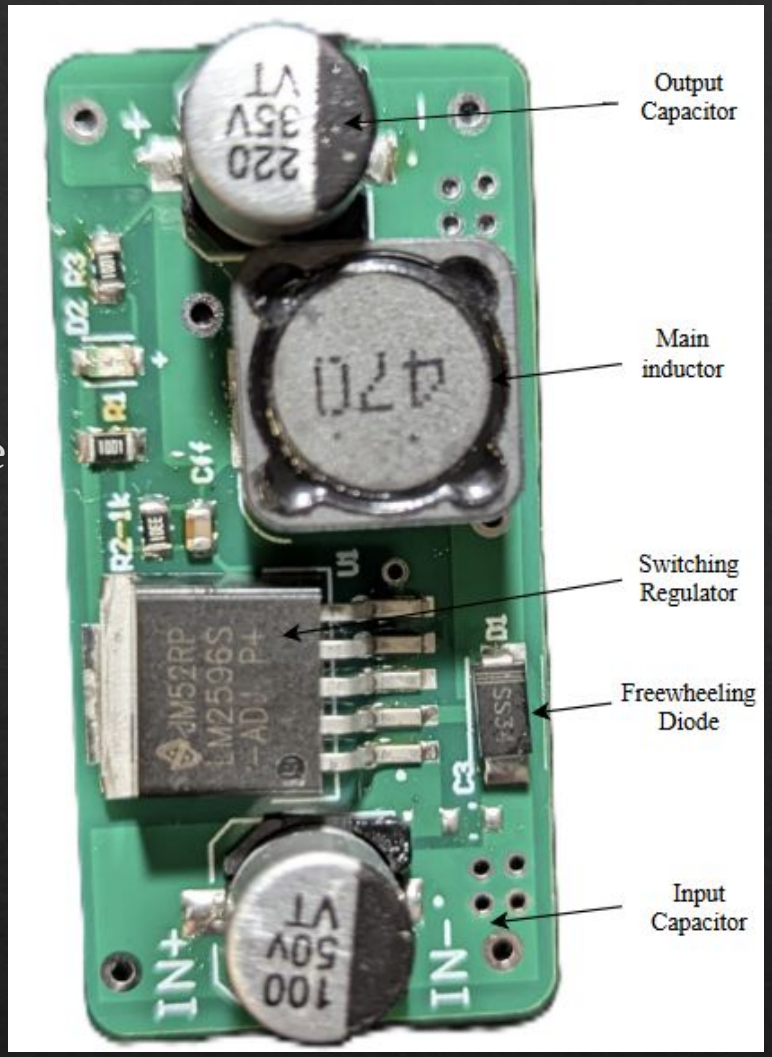
Servo Buck Converter

Primary function

- Provide sufficient current capacity to the Motors at 5V from Battery source
- Max current = 2.5A

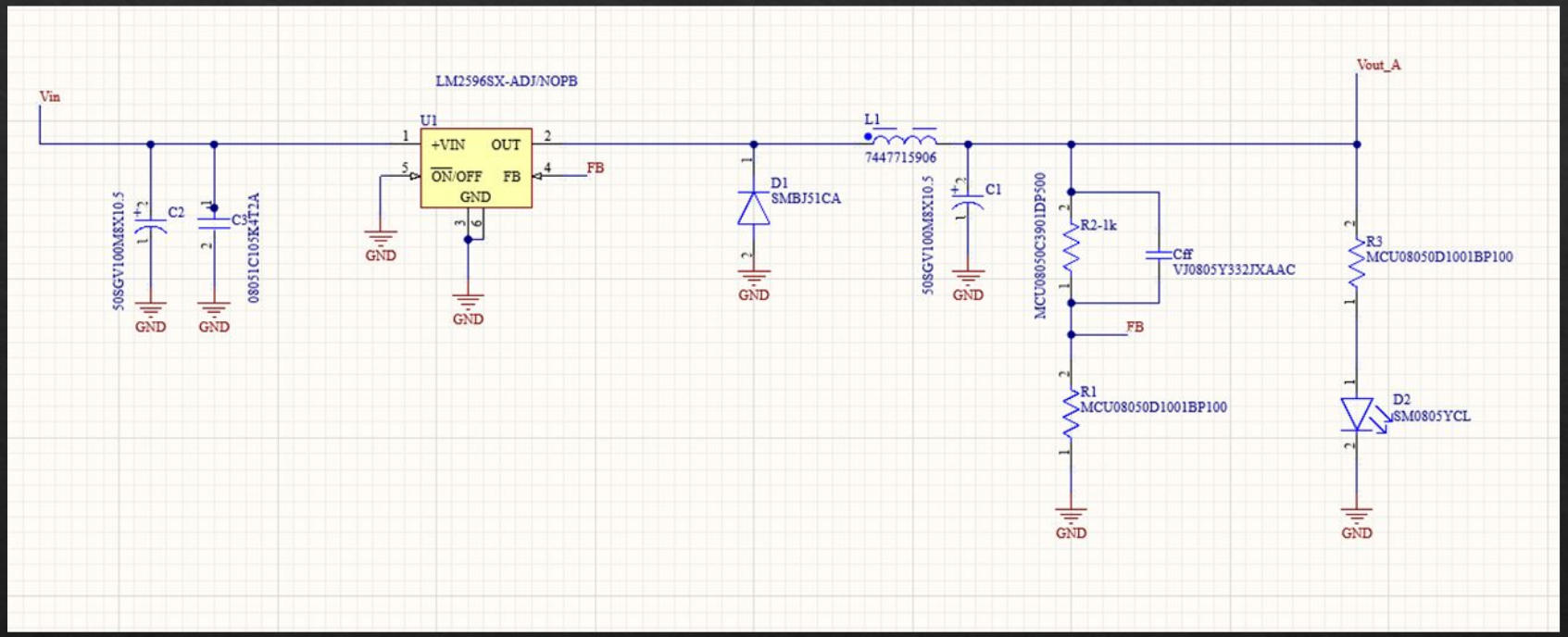
Main Components

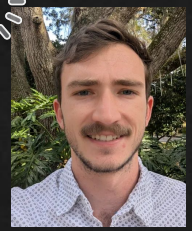
- Buck controller : LM2596-ADJ
- Freewheeling Diode: SS34
- Inductor : 47uH



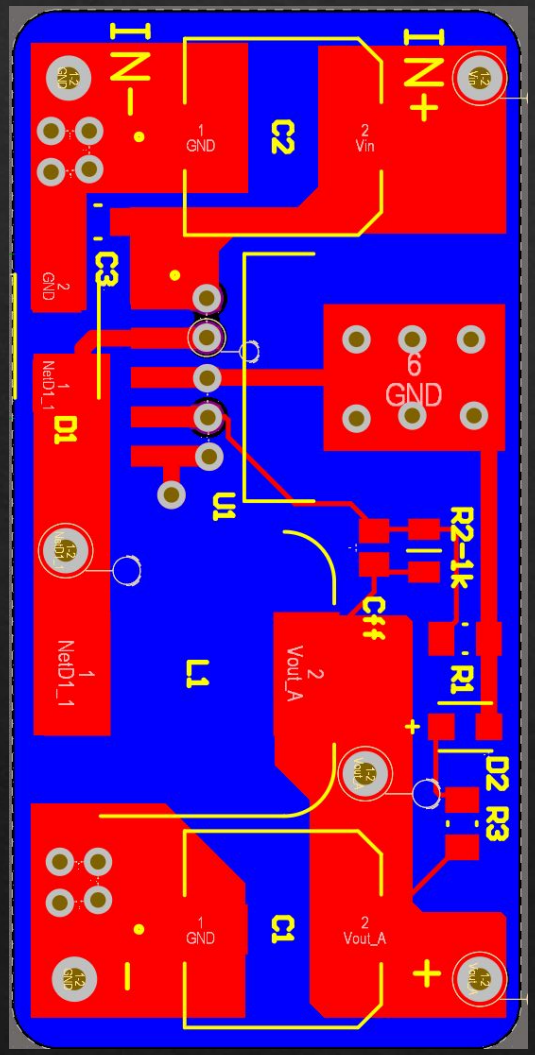
Buck Converter Schematic

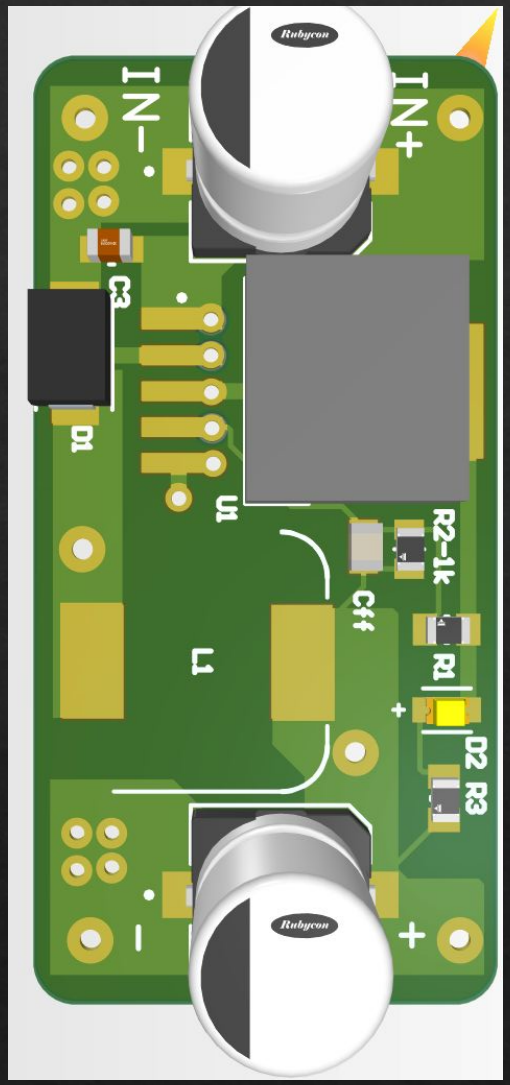
Buck Converter: LM2596

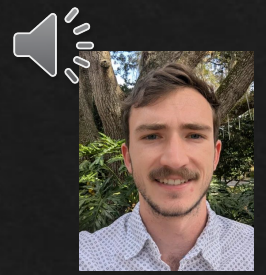




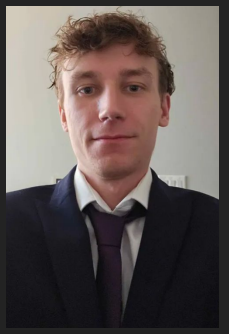
Buck converter Layout





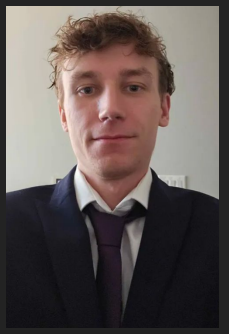


Selection of Software



FreeRTOS	Arduino
Good multitasking	Limited multitasking (single-threaded)
Deterministic RTS	Non-deterministic RTS
Good Hardware support	Good Hardware support
High RAM usage	Low overhead
Smaller community libraries	Strong Community libraries

Selection of Software

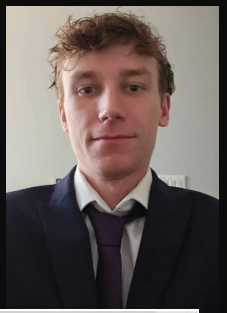


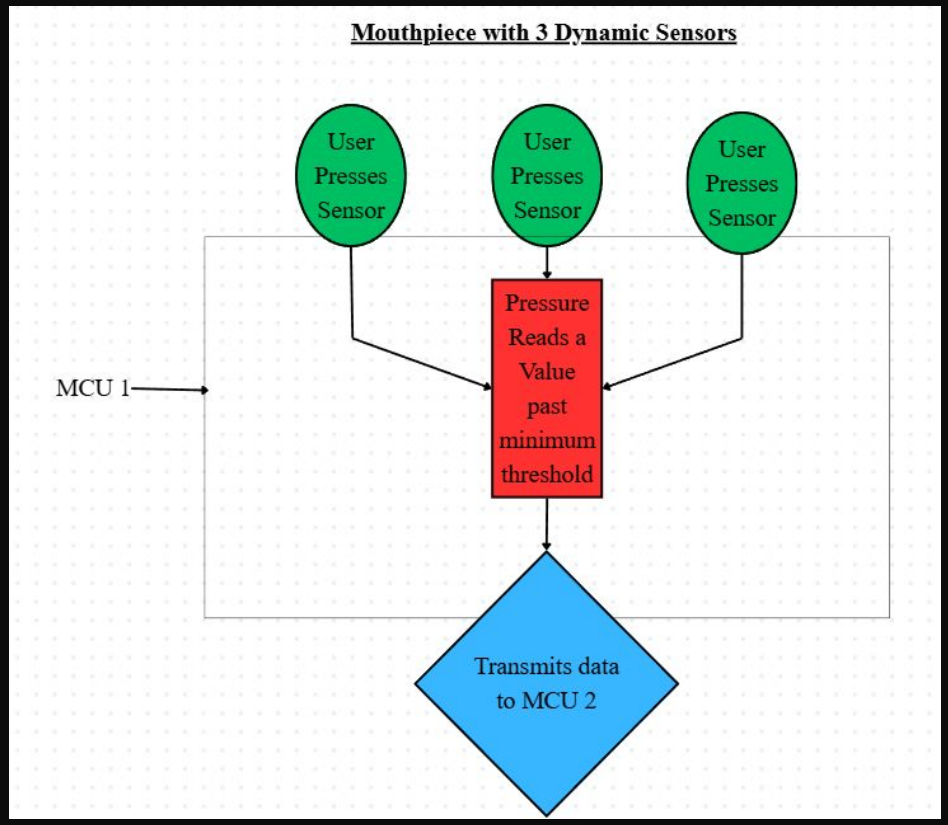
Python	C++
Vast and easy to use ecosystem	Complex Syntax
Better memory management	User-managed memory
Slower communication	Faster communication
Abstract hardware access	Direct Hardware control
Higher memory usage	Lower memory usage

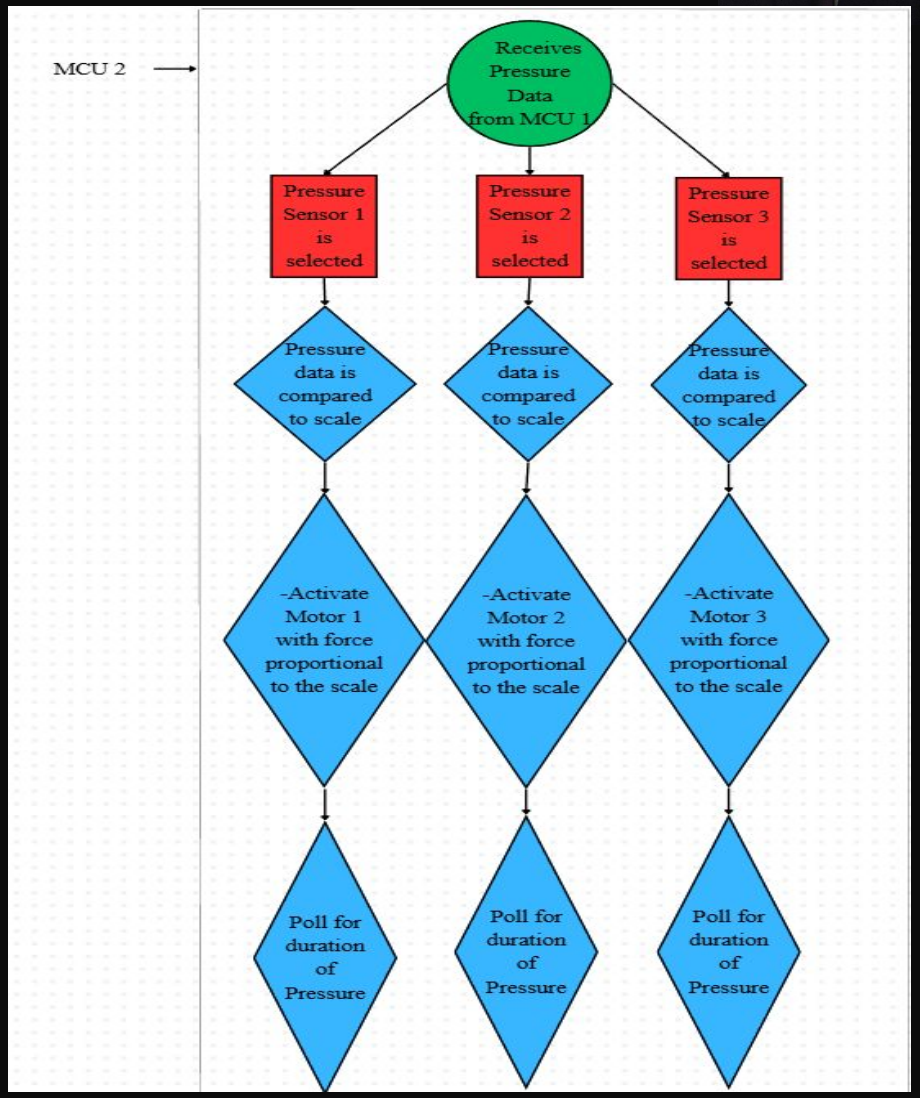
Selection of Communication Protocols

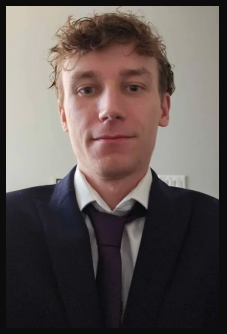
Wired	Wifi	Bluetooth	ESP-now
Great complex protocol handling	Great complex protocol handling	Ok complex protocol handling	Poor complex protocol handling
Rangle limited by wire size	Longer range 50-100m	Short range 10-40m	Good range 50-100
Great Ability to handle multiple devices	Ok Ability to handle multiple devices	Great Ability to handle multiple devices	Low Ability to handle multiple devices
Good for high throughput	Good for high throughput	Great for short bursts of information	Only small data packets
Medium Power consumption	Higher power consumption	Low power consumption	Lower power consumption
Little to no Latency	High Latency	Ok Latency	Low Latency

Software design

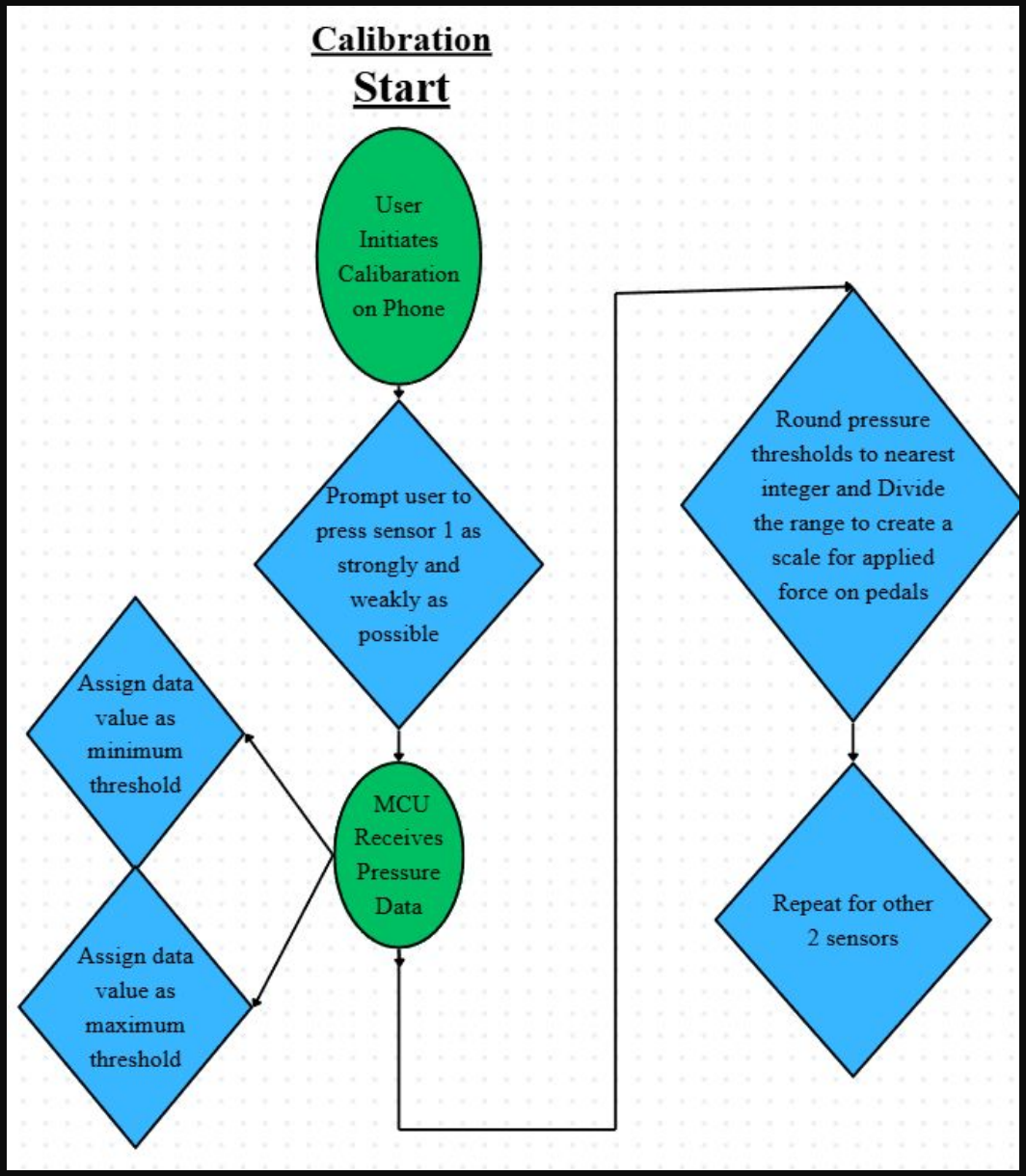






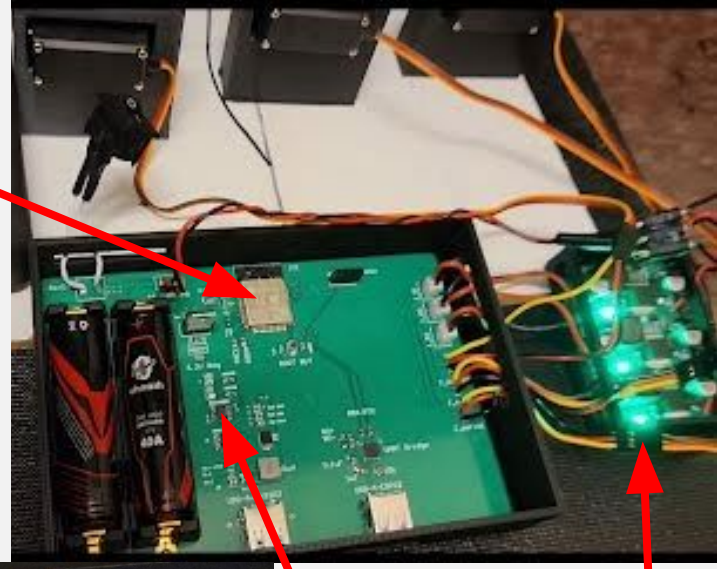


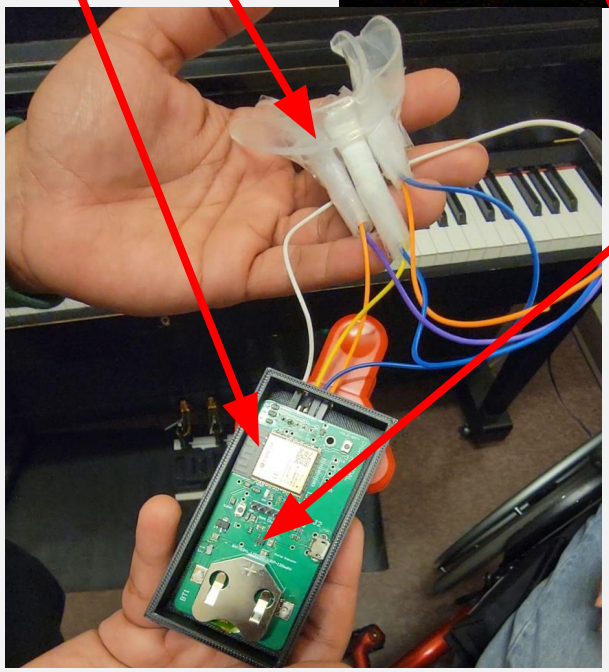
Software design



Testing

- **Functional Testing**: Verifying if the system fulfills specifications and requirements for both hardware and software.
- **Quantitative Benchmarks**: Confirming a ≤2s response time, ≥ 4m wireless communication range, ≥ 1hr battery life, and regulator testing at maximum load(5V, 3A).
- **User Experience Evaluation:** Feedback from our teammate Benedict who has paraplegia in terms of comfort, usability, and ease of learning.





Control Software Testing

- The PWM generation system was tested by simulating sensor input values via software variables.
- Motor responded with a 500ms average, transitioning smoothly between “pedal on” and “pedal off” states.
- Logic loop and signal pathways from the controller to the servos proven functional.

```
22  static const char *TAG = "PRESSURE_STREAM";
23
24  // --- Pins (ESP32-C6: GPIO2/3/4 are ADC-capable) ---
25  #define CH0_GPIO 2   // Sensor 1
26  #define CH1_GPIO 3   // Sensor 2
27  #define CH2_GPIO 4   // Sensor 3
28
29  // --- Streaming rate & averaging ---
30  #define STREAM_HZ 10
31  #define STREAM_DELAY pdMS_TO_TICKS(1000 / STREAM_HZ)
32  #define AVG_SAMPLES 32
33
34  // --- Wi-Fi / ESP-NOW ---
35  #define WIFI_CHANNEL 1   // Set same channel on sender/receiver
36  static const uint8_t PEER_MAC[6] = { 0x7B, 0x1C, 0x3C, 0xCB, 0xD0, 0xF8 }; // <-- REPLACE with receiver's STA MAC
37
38  static adc_oneshot_unit_handle_t adc1;
39  static adc_cali_handle_t cali = NULL;
40  static bool cali_enabled = false;
41
42  /* ---------- Payload ---------- */
43  typedef struct __attribute__((packed)) {
44      uint32_t ms;
45      int16_t v0;
46      int16_t v1;
47      int16_t v2;
48      int8_t  units; // 0 = mV, -1 = raw_counts (negative raw returned)
49  } pressure_pkt_t;
50
51  /* ---------- ADC calibration ---------- */
52  static bool adc_calibration_init(adc_unit_t unit, adc_atten_t atten, adc_cali_handle_t *out) {
53      adc_cali_handle_t h = NULL;
54      bool ok = false;
55
56  #if ADC_CALI_SCHEME_CURVE_FITTING_SUPPORTED
57      adc_cali_curve_fitting_config_t cfg = { .unit_id = unit, .atten = atten, .bitwidth = ADC_BITWIDTH_DEFAULT };
58      if (adc_cali_create_scheme_curve_fitting(&cfg, &h) == ESP_OK) ok = true;
59  #elif ADC_CALI_SCHEME_LINE_FITTING_SUPPORTED
60      adc_cali_line_fitting_config_t cfg = { .unit_id = unit, .atten = atten, .bitwidth = ADC_BITWIDTH_DEFAULT };
61      if (adc_cali_create_scheme_line_fitting(&cfg, &h) == ESP_OK) ok = true;
62  #endif
63
64      *out = h;
65      if (ok) ESP_LOGI(TAG, "ADC calibration enabled");
66      else    ESP_LOGW(TAG, "ADC calibration not available; printing raw counts if needed");
67      return ok;
68  }
```

Input Board code

```
99      void calibration_task(void *arg);
100
101     //=========================================================================
102     // --- Initialization
103     //=========================================================================
104
105     void init_wifi() {
106         Serial.println("Initializing Wi-Fi (AP+STA)...");
107         WiFi.mode(WIFI_AP_STA);
108         if (strlen(AP_PASSPHRASE) == 0) WiFi.softAP(AP_SSID, nullptr, WIFI_CHANNEL);
109         else WiFi.softAP(AP_SSID, AP_PASSPHRASE, WIFI_CHANNEL);
110         WiFi.begin(); // start STA interface without connecting - required for esp-now mac access
111         delay(200);
112         WiFi.macAddress(s_sta_mac);
113         IPAddress apIP = WiFi.softAPIP();
114         Serial.printf("AP %s IP %s CH %d\n", AP_SSID, apIP.toString().c_str(), WIFI_CHANNEL);
115         Serial.printf("MAC: %02X:%02X:%02X:%02X:%02X:%02X\n",
116                       s_sta_mac[0], s_sta_mac[1], s_sta_mac[2],
117                       s_sta_mac[3], s_sta_mac[4], s_sta_mac[5]);
118     }
119
120     void init_espnow() {
121         Serial.println("Initializing ESP-NOW...");
122         if (esp_now_init() != ESP_OK) {
123             Serial.println("ESP-NOW init failed");
124             return;
125         }
126         esp_now_register_recv_cb(espnow_recv_cb);
127         Serial.println("ESP-NOW ready");
128     }
129
130     void init_pwm() {
131         Serial.println("Initializing PWM...");
132         ledcSetup(PWM_CH0, PWM_FREQ, PWM_RESOLUTION);
133         ledcSetup(PWM_CH1, PWM_FREQ, PWM_RESOLUTION);
134         ledcSetup(PWM_CH2, PWM_FREQ, PWM_RESOLUTION);
135         ledcAttachPin(PWM_CH0_PIN, PWM_CH0);
136         ledcAttachPin(PWM_CH1_PIN, PWM_CH1);
137         ledcAttachPin(PWM_CH2_PIN, PWM_CH2);
138         // Initialize to middle position
139         ledcWrite(PWM_CH0, map_adc_to_pwm_calibrated(0, 0));
140         ledcWrite(PWM_CH1, map_adc_to_pwm_calibrated(0, 1));
141         ledcWrite(PWM_CH2, map_adc_to_pwm_calibrated(0, 2));
142         Serial.println("PWM ready");
143     }
144
145     void init_display() {
146         Serial.println("Initializing display...");
```

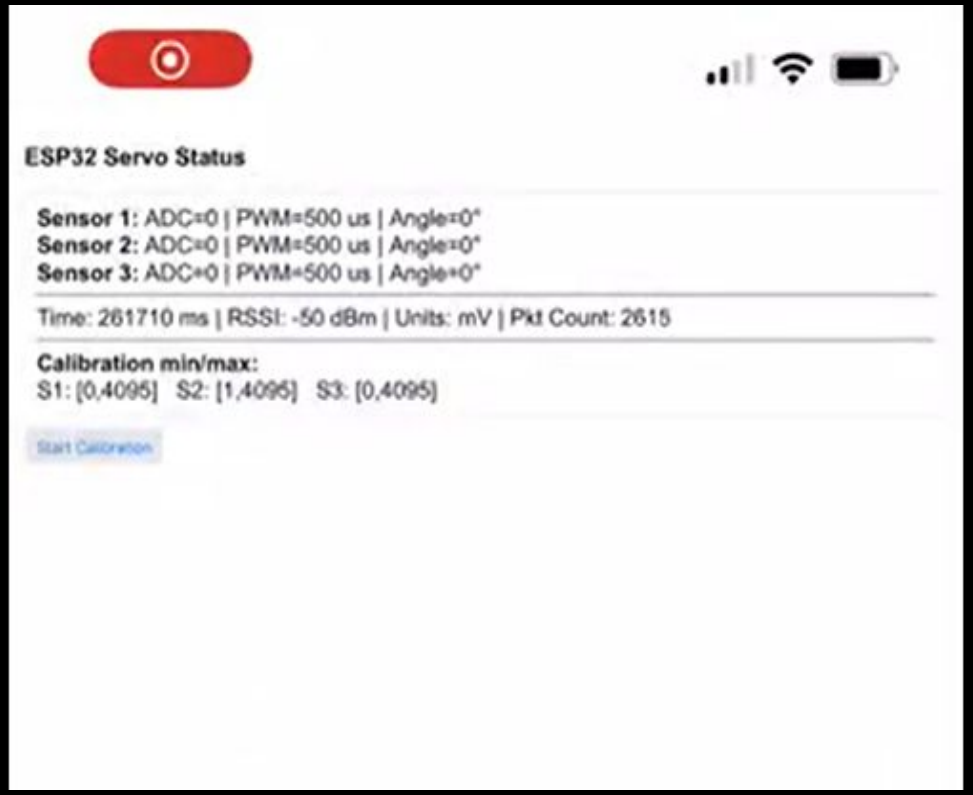
Output Board code

```
267
268 void handleRoot() {
269     String html = R"rawliteral(
270 <!doctype html>
271 <html>
272 <head>
273 <meta charset="utf-8"><title>ESP32 Servo Status & Calibration</title>
274 <style>
275 body{font-family:Arial;margin:16px}
276 .card{border:1px solid #ddd;padding:12px;border-radius:8px;margin-bottom:10px}
277 button{padding:8px 12px;font-size:16px;border-radius:6px}
278 .counter{font-weight:bold}
279 </style>
280 </head>
281 <body>
282 <h2>ESP32 Servo Status</h2>
283 <div id="status" class="card">Loading...</div>
284 <button id="calBtn">Start Calibration</button>
285 <div id="calInfo" class="card" style="display:none"></div>
286
287 <script>
288 async function fetchStatus(){
289   try {
290     const res = await fetch('/status');
291     const j = await res.json();
292     const s = document.getElementById('status');
293     if(!j.have_packet) {
294       s.innerHTML = '<i>No packet yet</i>';
295     } else {
296       let html = '';
297       html += '<div><b>Sensor 1:</b> ADC='+j.v0+' | PWM='+j.pwm0_us+' us | Angle='+j.angle0+'°</div>';
298       html += '<div><b>Sensor 2:</b> ADC='+j.v1+' | PWM='+j.pwm1_us+' us | Angle='+j.angle1+'°</div>';
299       html += '<div><b>Sensor 3:</b> ADC='+j.v2+' | PWM='+j.pwm2_us+' us | Angle='+j.angle2+'°</div>';
300       html += '<hr>';
301       html += '<div>Time: '+j.ms+' ms | RSSI: '+j.rssi+' dBm</div>';
302       html += '<hr>';
303       html += '<div><b>Calibration min/max:</b></div>';
304       html += '<div>S1: ['+j.min0+','+j.max0+']   S2: ['+j.min1+','+j.max1+']   S3: ['+j.min2+','+j.max2+']</div>';
305       s.innerHTML = html;
306     }
307
308     // Show calibration info if active
309     const ci = document.getElementById('calInfo');
310     if(j.calibration_active) {
311       ci.style.display = 'block';
312       ci.innerHTML = '<b>Calibration active</b><br>Press sensor: <b>Sensor '+(j.current_sensor+1)+'</b><br>Seconds left: <span class="counter">'+j.seconds_left+'</span>';
313     } else {
314       ci.style.display = 'none';
```

Functions Display Web-interface code

Calibration Testing

- The test was initiated by activating the "setup mode" button on the device's web page.
- During the subsequent 5-second automated sampling period, the user applied minimal and maximal tongue pressure to each of the three sensors,
- Upon completion measured ADC value ranges were stored the as the new operational thresholds.
- To confirm, post-calibration inputs were applied to the mouthguard. The servo motors were observed to actuate with a full and proportional range of motion in direct response to the newly calibrated pressure inputs, confirming the calibration process was successful.



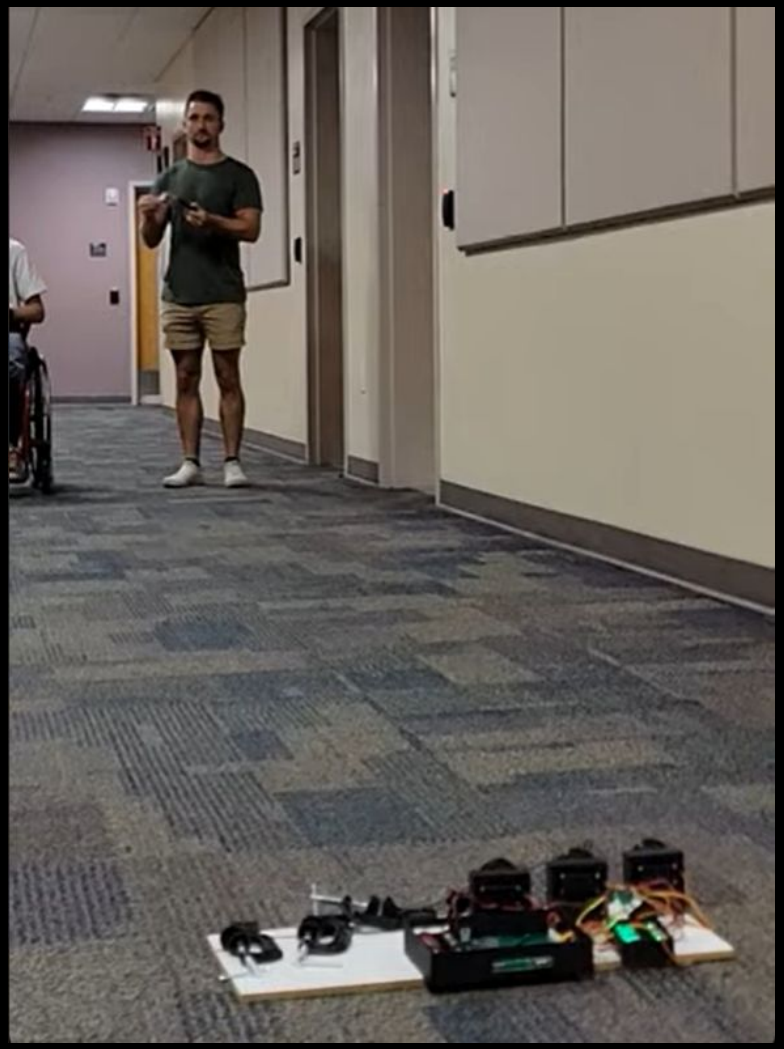
CalibrationTest.mov

Software Bugs & Solutions

- **Problem 1:** Wasn't able to transmit the full range of the users' input.
- **Solution:** Replaced ADC_6db with ADC_11db, increasing our sensor voltage from 1.1V to 3.3V.
- **Problem 2:** Servo Motor twitching, nearly unresponsive.
- **Solution:** Decreased the PWM frequency from 80Hz to 50Hz.
- **Problem 3:** Insufficient motor torque for the final output setup
- **Solution:** Increased regulator output from 5V to 6V by replacing feedback resistor

Communication Range Testing

- The motor was consistently actuated from the input device and an increasing range.
- The motor consistently actuated with a delay of below 1s at 5m at beyond
- These results exceeded our system requirements for communication range



https://youtube.com/shorts/2e0glyle1Q8

Battery Life Testing

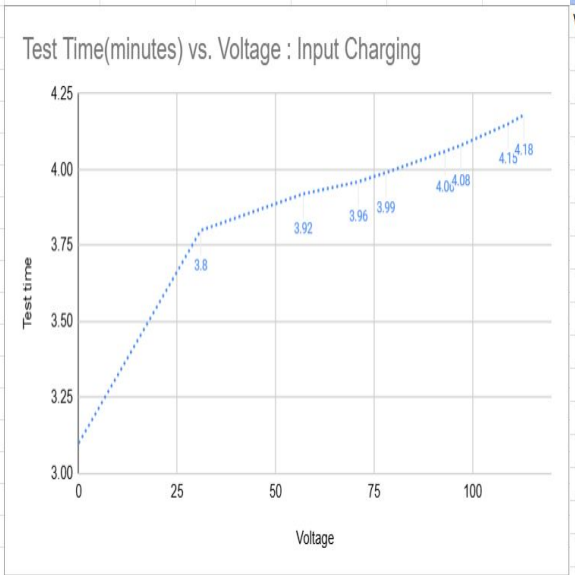
- Input board:
  - Output board connected to a DC power supply.
  - Constant data transmission with the Output Board to consistently drain the battery.
  - Batteries dropped from 4.1V, to 3.6V over 1 hour.
- Output Board:
  - Actuated all three servo motors in sequence consistently
  - Starting voltage was 8.2 V, ending at 8.1V one hour later

Battery Charger Testing

- The Input Board recharged from 3.1V to 4.18V data in 113 minutes.
- The Output Board recharged from 7.44V to 8.11V in 119 minutes.
- The only problem we encountered was replacing BQ25887 recharging unit with a CN3302 unit.

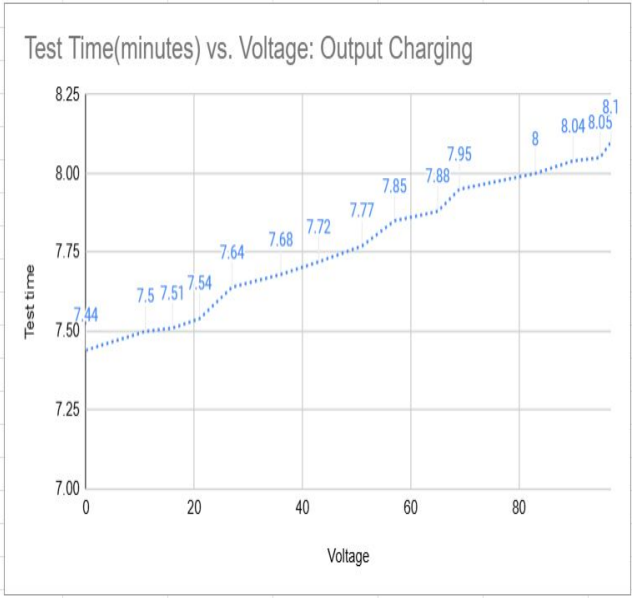
INPUT CHARGING

Voltage	Time	Test time
3.1	3:23	0
3.8	3:54	31
3.92	4:20	57
3.96	4:34	71
3.99	4:41	78
4.06	4:56	93
4.08	5:00	97
4.15	5:12	109
4.18	5:16	113



OUTPUT CHARGING

Voltage	clock Time	Test time
7.44	5:35	0
7.5	5:46	11
7.51	5:51	16
7.54	5:56	21
7.64	6:02	27
7.68	6:11	36
7.72	6:18	43
7.77	6:26	51
7.85	6:32	57
7.88	6:40	65
7.95	6:44	69
8	6:58	83
8.04	7:05 AM	90
8.05	7:10	95
8.1	7:12	97
8.11	7:34	119



Regulator Testing

- The buck converter was tested by supplying power through an adjustable DC power supply set to 7.4V and an adjustable load.
- The adjustable load began at 2A, [0.5A above the servo motors' usual current draw (1.5A)].
- The load was slowly increased the Working Load Current for the servos (2.5A).
- The output voltage remained within the acceptable range.

7.40V , 2.44A DC Supply

2.5A Current

5.11V output

Budget

Budget							
Comment	Description	Designator	Footprint	LibRef	Quantity	Manufacturer Part Number	Cost
MP2338GTL	Battery to 5V Buck				10		$11.30
MS621FE-FL11E	MS Lihium Rechargeable Battery	Lion Battery?	SEIK-MS621FE-FL11E-2_V	CMP-00029-00000-1	2		$13.29
ESP32-WROOM-32D	WIFI MODULE 32MBITS SPI FLASH	PeripheralController	FP-ESP32-WROOM-32D-MFG	CMP-194065-000003	1	ESP32-WROOM-32D	$5.38
Boot Select button	Push Button Normally Opened	S1		Push Button NO	1		$1.94
CP2102-GMR	Single-Chip USB to URAT Bridge	U?	SLAB-QFN-28-3125x3125TP_M	CMP-2000-04924-1	1		$8.88
LP2966IMM-1833/NOPB	Dual 150mA Ultra Low-Dropout Regulator	U?	MUA08A_N	CMP-0062-02215-2	10		$15.40
BQ25887	Battery Manager				2		$10.00
18650 Battery Case					2		$9.99
18650 Batteries					2		$3.00
Satin String Spool					1		$6.99
Servo Pulley					20		$3.99
MG996R	Servo Motor				4		$18.00
RP-S5	Resistive Pressure Sensors				2		$9.50
Mouthguard					20		$8.99
						Total Budget	$122.66

- Compared to our budget expectations, we spent 509% more than expected through R&D at $624.49 spent in total.
- Removing the R&D expenses, the production cost (without 3D printing cost) we exceeded the budget by 205% at $251.53 production cost.

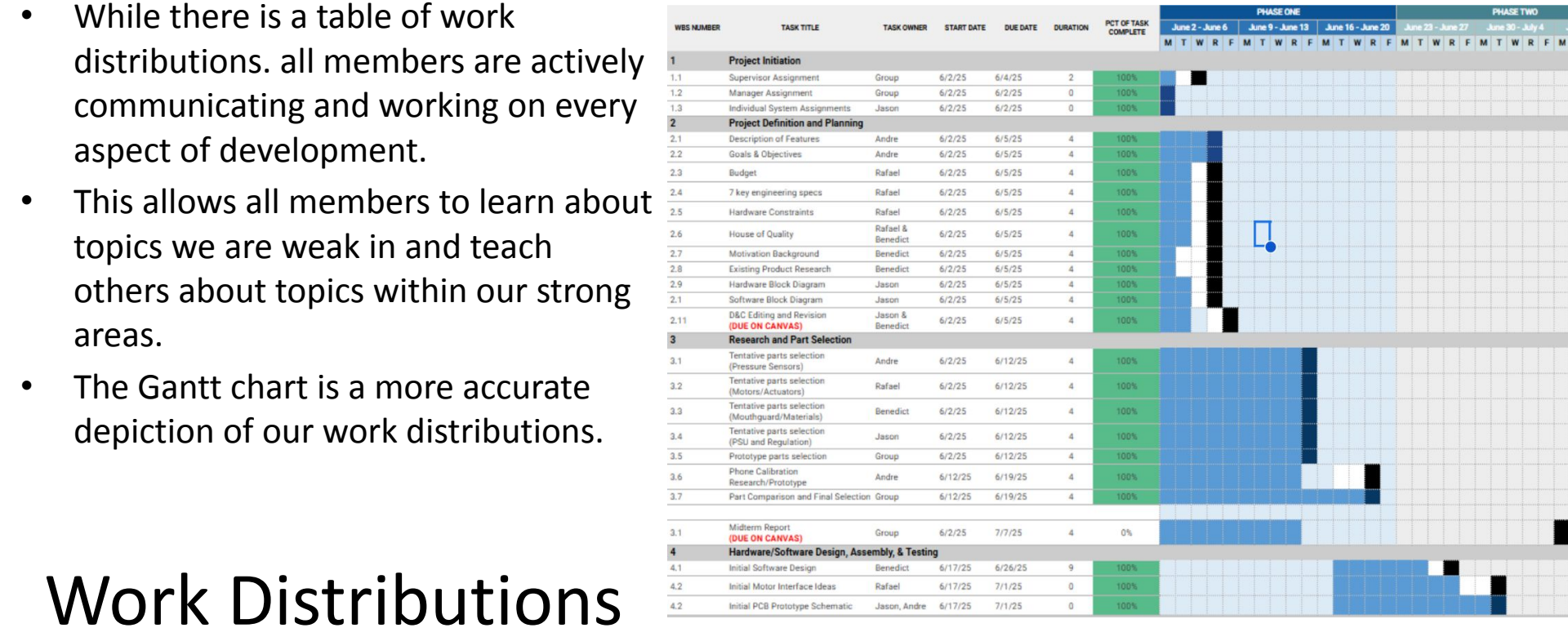
Production Cost -w/o 3D printing							
Comment	Description	Designator	Footprint	LibRef	Quantity	Manufacturer Part Number	Cost
Digikey component order	Components				1		$106.63
JLC PCB order 2	Final Boards				1		$91.85
MG996R	Motors				3		$22.00
Mouthguards	Mouthpeices				1		$9.00
RP-S	Resistive Pressure Sensor				3		$15.05
Satin String Spool	String				1		$7.00
						Total Poduction	$251.53

Work Distributions

- While there is a table of work distributions. all members are actively communicating and working on every aspect of development.
- This allows all members to learn about topics we are weak in and teach others about topics within our strong areas.
- The Gantt chart is a more accurate depiction of our work distributions.

Gantt Chart

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Project Initiation**					
1.1	Supervisor Assignment	Group	6/2/25	6/4/25	2	100%
1.2	Manager Assignment	Group	6/2/25	6/2/25	0	100%
1.3	Individual System Assignments	Jason	6/2/25	6/2/25	0	100%
2	**Project Definition and Planning**					
2.1	Description of Features	Andre	6/2/25	6/5/25	4	100%
2.2	Goals & Objectives	Andre	6/2/25	6/5/25	4	100%
2.3	Budget	Rafael	6/2/25	6/5/25	4	100%
2.4	7 key engineering specs	Rafael	6/2/25	6/5/25	4	100%
2.5	Hardware Constraints	Rafael	6/2/25	6/5/25	4	100%
2.6	House of Quality	Rafael & Benedict	6/2/25	6/5/25	4	100%
2.7	Motivation Background	Benedict	6/2/25	6/5/25	4	100%
2.8	Existing Product Research	Benedict	6/2/25	6/5/25	4	100%
2.9	Hardware Block Diagram	Jason	6/2/25	6/5/25	4	100%
2.1	Software Block Diagram	Jason	6/2/25	6/5/25	4	100%
2.11	D&C Editing and Revision (DUE ON CANVAS)	Jason & Benedict	6/2/25	6/5/25	4	100%
3	**Research and Part Selection**					
3.1	Tentative parts selection (Pressure Sensors)	Andre	6/2/25	6/12/25	4	100%
3.2	Tentative parts selection (Motors/Actuators)	Rafael	6/2/25	6/12/25	4	100%
3.3	Tentative parts selection (Mouthguard/Materials)	Benedict	6/2/25	6/12/25	4	100%
3.4	Tentative parts selection (PSU and Regulation)	Jason	6/2/25	6/12/25	4	100%
3.5	Prototype parts selection	Group	6/2/25	6/12/25	4	100%
3.6	Phone Calibration Research/Prototype	Andre	6/12/25	6/19/25	4	100%
3.7	Part Comparison and Final Selection	Group	6/12/25	6/19/25	4	100%
3.1	Midterm Report (DUE ON CANVAS)	Group	6/2/25	7/7/25	4	0%
4	**Hardware/Software Design, Assembly, & Testing**					
4.1	Initial Software Design	Benedict	6/17/25	6/26/25	9	100%
4.2	Initial Motor Interface Ideas	Rafael	6/17/25	7/1/25	0	100%
4.2	Initial PCB Prototype Schematic	Jason, Andre	6/17/25	7/1/25	0	100%



Work Distributions

Name	Primary Tasks	Secondary Tasks
Jason	Manager, Power	PCB
Andre	PCB Design	Coding
Rafael	Motors	PCB
Benedict	Coding	PCB

Final Tests & Conclusion

- Finally, our final product met all constraints, with some 3D Printed mechanical adjustments

Regulator Array

Output Board

Input Board

MouthPiece