ECE 445 FINAL REPORT

By

Ankur Prasad

Matthew Uthayopas

Tunc Gozubuyuk

Final Report for ECE 445, Senior Design, Spring 2026

TA: Mingrui Liu

6 May 2026

Project No. 30

Abstract

The following report describes the design process, implementation and verification of an American Sign Language (ASL) Robotic Hand. It is a tendon driven 3D printed system that is designed to help in teaching ASL. It contains two subsystems which are a flex sensor glove that is worn by an instructor as well as a seven-servo motor robot hand which are both controlled via an ESP32-S3. The system is trained by an instructor performing signs wearing a glove; a DBSCAN machine learning algorithm captures sensor data and extracting optimal angles per letter and creating a lookup table. During classroom use, the hand either cycles through the stored letter poses or holds a certain letter via a command from the Android app through BLE. The system is successfully able to sign nine ASL letters (A,B,D,F,I,L,U,W,Y) with over an 85% accuracy which was independently recognized by more than 3 ECE students.

Contents

1. Introduction

American Sign Language (ASL) serves as an important mode of communication for the Deaf and hard-of-hearing community. In ASL, proficiency is judged not on vocabulary, but on the execution of hand poses and gestures. As such, effective instruction requires a clear visual demonstration and, more importantly, immediate feedback so that learners build gestures with severe inaccuracies. As it stands, the current educational methods fall short of providing the nuanced, 3-D spatial feedback that is essential for mastering ASL. This feedback gap is acutely felt in learning environments such as high school classrooms, where a single instructor is responsible for making corrections and guiding numerous students. The resulting inefficiency can lead to student frustration and poor retention that translates into a systemic barrier to inclusive ASL education. We created a tendon-driven bionic hand that is capable of performing nine of the letters of the alphabet in the American Sign Language, designed as an interactive and cost-efficient tool. At a high level, the system consists of two components: a robotic hand and a glove sensor. The robotic hand will contain five movable fingers that will change positions to achieve the ASL letter positions. The sensor glove will be worn by a proficient user and capture their finger movements, and wirelessly transmit that data to the actual robotic hand. The robot hand will then assume the same pose, which allows the hand to serve as a physical three-dimensional demonstration tool that students can observe from any angle while practicing their own signs or the robot hand will cycle through letters on its own so the user can follow along and mimic the robot arm's movements. This system is designed to remove the burden of a constant one-to-one feedback loop from the teacher to the student and will enable self-guided practice with a physical tool to give immediate physical feedback.

1.1 Block Diagram

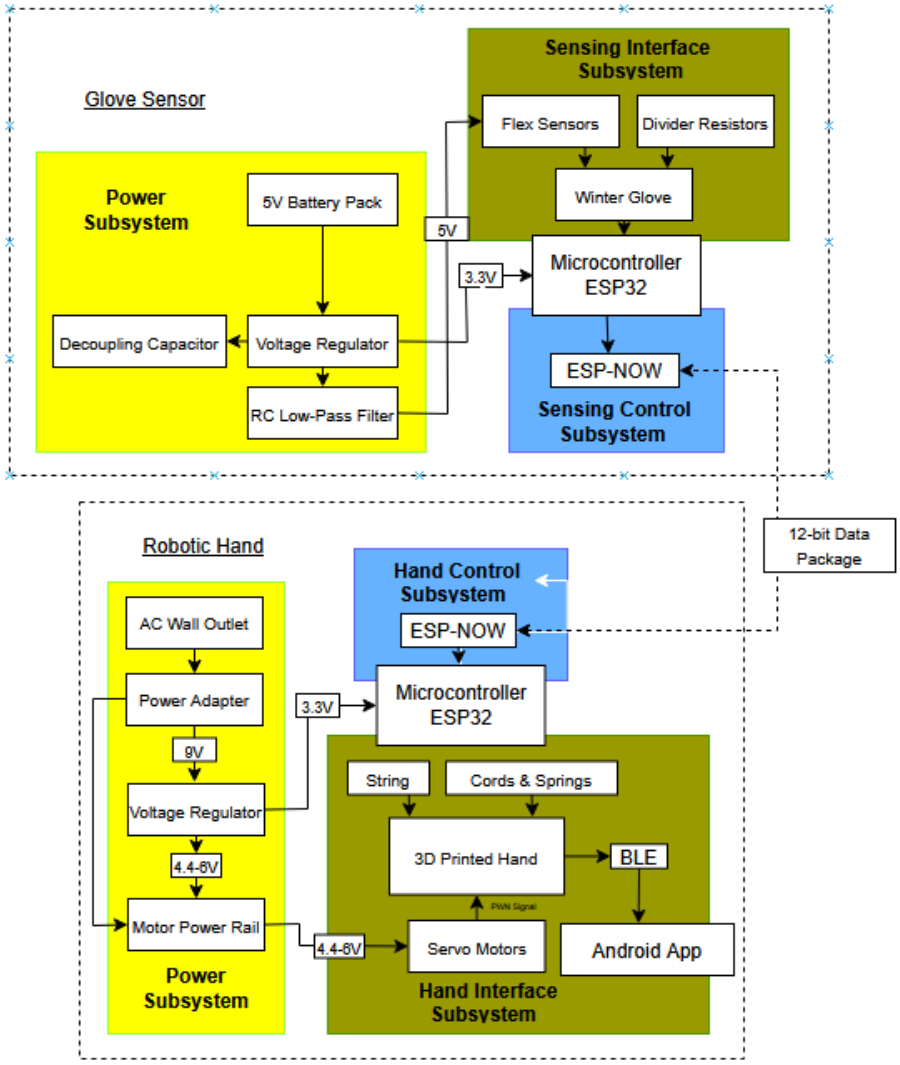

Figure 1 Block Diagram of Design

Figure 1 describes the project in its entirety. It contains the two main subsystems which are the Glove Sensor and the Robotic Hand. Each main subsystem contains three subsystems that are responsible for helping the other subsystems. These are power subsystems, control subsystems, and interface subsystems. The power subsystems will ensure that every component on our design receives the correct amount of voltage and current, the control subsystems will determine what the glove and hand should be doing, and the interface subsystems will communicate between the two main subsystems.

2 Design

This section discusses the design rationale along with the details of implementation for each subsystem in the project. Section 2.1 covers the physical design of the robotic hand and overall structure. Section 2.2 and 2.3 will cover elements of the glove system. Section 2.4 and 2.5 will cover elements of the robot hand control and electronics.

2.1 Physical Design

The robotic hand was 3D printed using a Bambu Lab Carbon X1 FDM printer with PLA filament. The primary design of the robotic hand was adapted from the open-source Aero Hand model made by Tether IA [3]. The actual hand measured approximately 22cm long and 20cm wide with at depth of 15cm. It contains 4 identical fingers which contain three joints (DIP joint, PIP joint, and MCP joint) which are all capable of 90 degrees of rotation making them bend like an actual human hand. Each finger had 5 pieces which connect to each other via 2mm x 14mm pins with some of the pieces have bearings inside that are held in with 2mm x 10mm pins. A Kevlar string runs through the interior channel of each finger from the top of the finger through the bottom and then to a spool mounted on a servo motor that is placed on a servo cage. It mimics a traditional tendon driven design that the actual human hand utilizes in the sense that when the servo motor rotates, it winds the string which draws the finger into flexion. Springs were placed at the Metacarpophalangeal Joint and Proximal Interphalangeal Joint to provide restoring force to return the finger back to extension. The Distal Interphalangeal joint uses a looped string approach to provide restoring force thar proved to be insufficient.

Moving on to the thumb, which is a difficult part of the design, requires three degrees of freedom due to replicating a saddle joint. The thumb needs to do flexion (the top joint bending), opposition (rotating inwards to touch other fingers), and adduction (moving toward the index finger). Three servo motors are dedicated to these movements in order to achieve the desired movement. For the most optimal and fluid movements, the servo motor responsible for opposition was placed inside the hand. For the other two movements, similar to the fingers, a Kevlar string was run through the different points of the thumbs, brought down and connected to a spool which was then mounted onto a servo.

The servo cage is also 3D printed using the same Bambu Lab Carbon X1 FDM printer with PLA filament and it is designed to house the 6 remaining servo motors. The cage consists of a rectangular frame with open cutouts to reduce the overall weight while maintaining structural integrity. It features a modular design where each servo motor has its own bracket where the servo motor is put in rectangular bay allowing the spool to be mounted on the inside. The top of each bracket features a slight overhang with a hole (of diameter 3mm) in the center to provide the string with a secure dedicated path to ensure it does not get tangled. The brackets connect in a squared structure where the top is wider than the bottom using M2 and M3 screws.

The connector between the robot hand and palm was a point of failure that required multiple iterations where the initial design, a pilar design, broke under the load of six servo motors pulling at full range of rotation. It was then designed to have connected pillars with a large contact where the palm can rest on.

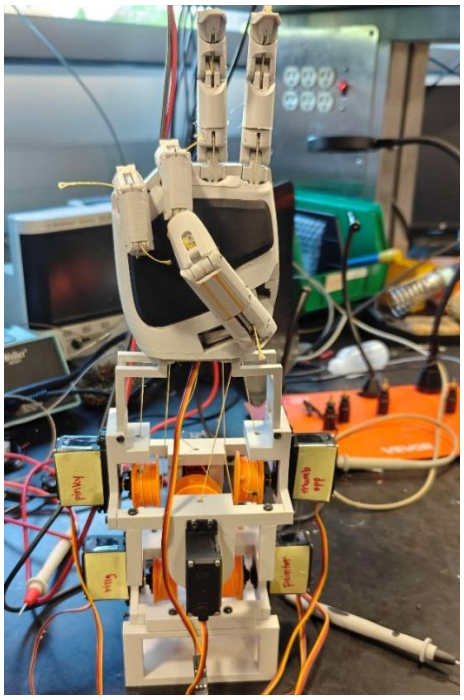

Figure 2 3D-printed robotic hand. Each of the four fingers is actuated by one servo motor via a tendon string; the thumb uses three servos. The servo cage and PCB are housed in the forearm enclosure.

2.1.1 Design Procedure

We selected an ESP32-S3-WROOM-1 for both of our PCBs because it supports both ESP-NOW and BLE while being able to integrate native USB for serial data logging during training. This eliminates the need for separate radio modules. For example, if we were to choose an STM microcontroller, then we may have had to add a separate Bluetooth module to both of our PCBs. Although this would improve latency, it would only do so slightly and the time and cost it would take to implement outweigh the benefits.

We considered three ways to move each finger. First, we thought of using rigid linkages with servos at each joint, but this would not work since we would have to utilize way too many servos and would change the structure of the hand. Then we considered a tendon driven system using strings which was the more optimal choice due to the geometry required along with the resources available to us.

The glove system sends signals through the flex sensors attached to it. In our initial planning of the project, we were deciding between these flex sensors or computer vision for capturing data. Computer vision was rejected because it would require using a camera, which contains would contain more variables for bugs due to lighting and multiple objects in the camera that aren't the hand. The flex sensors are simpler, since each sensor gives a direct correspondence between sensor bend and finger angle. Each sensor forms a voltage divider with a reference resistor, making calculations for the robotic hand simpler. This output is sent to the robotic hand as a data package.

Both PCBs require a 5V and 3.3V rail of power. The 5V rail is used to power the servo motors while the 3.3V rail is used to power the microcontroller and the flex sensors. Initially, we were going to use a linear regulator from the 9V power source, but servo motors draw a combined stall current of 2.5A,

meaning that the power dissipation is approximately (9 – 5)V x 2.5 A = 10 W. This meant that our system would be at a high risk of thermal shutdown, so we switched to a buck converter (MP1584), which is able to output 3A of current minimizing heat generation. For the 3.3V rail, the ESP32 microcontroller and flex sensors draw at most 500mA of current meaning that we can use a voltage regulator (LM1117-3.3) for ease of implementation without additional thermal management.

For communication between the two PCBs, we initially attempted to implement BLE for sending signals. However, this method of communication contained a latency of 100-150 ms and a more complicated pairing procedure than alternative solutions. Thus, we switched to the ESP-NOW protocol, which is built into ESP32 MCUs.

2.2 Glove Sensor System

The glove sensor system is responsible for sensing and training American Sign Language movements which will be used for the robotic hand. This will be done by capturing the user's finger and joint movements via 6 flex sensors. These flex sensors are held onto the glove through stitching and slits manually created on a wearable glove.

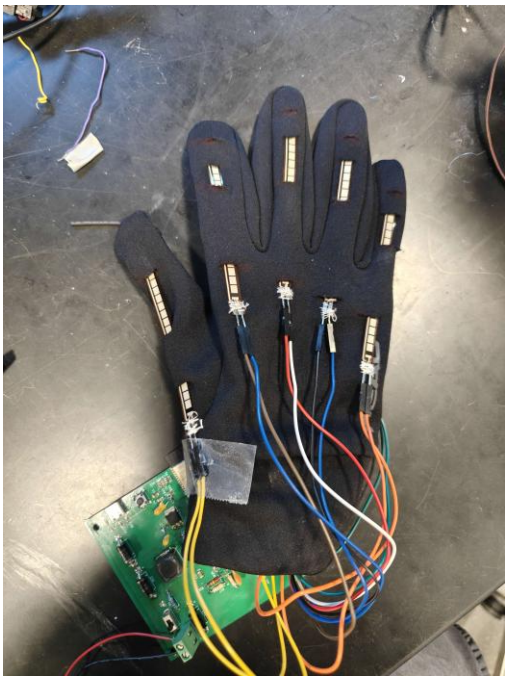

Figure 3 Flex Sensor Glove. Each of the four fingers has a flex sensor connected to it while the thumb has two (one above and one underneath). The flex sensors are attached to the PCB via 2x1 conn pins.

2.2.1 Hand Interface Subsystem

The control subsystem converts ADC voltage values into finger-bend angles using the six flex sensors mounted on the glove. Each flex sensor forms a voltage divider with a fixed 10 kΩ reference resistor driven from the 3.3 V rail on the PCB. As the user bends the finger, the flex sensor resistance increases,

lowering the overall voltage value the flex sensor outputs. Typically, the flex sensor outputs a value of around 1800 ADC when rested and outputs around 400 ADC when fully bent. The exact equation for finding the voltage output through the flex sensors is the following:

$$V_{out} = 3.3V \times R_{ref} / (R_{ref} + R_{flex})$$

Here, R_{ref} is the 10 kΩ fixed reference resistor and R_{flex} is the variable flex sensor's resistance. The six divider outputs are mapped to separate GPIO pins located on the ESP32 microcontroller. It is important to note that these GPIO pins (GPIO1, GPIO2, GPIO4, GPIO5, GPIO6, GPIO7) are dedicated ADC-capable pins on the microcontroller since some pins do not support ADC values. Each pin is configured for 12-bit analog input, where readings are in the range of 0-4095. The output voltage at each node stays within the ADC input range of 0-3.3V. Each flex sensor is connected to the PCB through 2-pin connectors where one pin takes in 3.3V of power and the other is mapped to the microcontroller. The ADC values are also calibrated to get a baseline of where a resting, half bend, and full bend are on the glove for each sensor since they all give different readings. The following is a table of this:

Sensor Axis	ADC Extended	ADC 90° Bend	ADC Full Bend	GPIO Pin
Pointer	1750	890	810	GPIO1
Middle	2085	1515	1257	GPIO2
Ring	1762	1315	1040	GPIO4
Pinky	1950	1125	885	GPIO5
Thumb Flex	1860	N/A	1195	GPIO6
Thumb Opp.	1645	N/A	1390	GPIO7

Table 1 Flex sensor ADC calibration ranges (12-bit, 0–3.3 V). Extended = finger straight; Full Bend = finger fully closed.

2.2.2 Control Subsystem

The control subsystem manages what the microcontroller does with the ADC acquisition from the 6 flex sensors. It also takes care of the wireless transmission between the two main subsystems for the glove and hand. Since we are using an ESP32-S3-WROOM-1, we have an already on-chip ESP-NOW, USB support, and BLE radio. Thus, we were able to design the PCB for the glove in a standard way with a push button for EN and BOOT while also having a USB port.

The glove will be able to take the ADC readings from the flex sensors and convert them into servo angles via piecewise linear interpolation across the anchor points from Table 1 and transmit the angles to the robotic hand to mimic at approximately 20Hz. This is done by mapping the fingers to a seven-integer angle vector [pointer, middle, ring, pinky, thumb_flex, thumb_opp, thumb_abd) and sending them via ESP-NOW.

The control subsystem also contains a way to export readings from the six flex sensors into a CSV file when the PCB is plugged in through a USB cable. This CSV file is used as a training pipeline to train our robotic hand to perform signs at high accuracy. The training pipeline is as follows:

1. An instructor or someone proficient in American Sign Language wears the glove and performs each ASL letter 20-30 times, while also holding each pose for approximately one second.
2. Raw 12-bit ADC readings from all six flex sensors are timestamped and logged to a laptop through the USB serial interface into a CSV file.
3. The CSV file is put into a Python post-processing script which will apply DBSCAN clustering algorithm (epsilon = 12, min samples = 3) for each letter.
    a. Rows where three or more sensor channels read 5 or below are discarded (this indicates sensor disconnection or noise) before clustering
4. Largest cluster of consistent angle readings are identified and computed for each letter, yielding a canonical seven-dimensional servo angle vector.
5. Angle vectors are stored as a lookup table and uploaded to the robotic hand's ESP32 microcontroller

2.2.3 Power Subsystem

The power subsystem focuses on providing the correct voltage and current for the microcontroller and flex sensors which both operate at 3.3 V. It accepts two input sources which are a 9 V battery pack via a barrel jack or 5 V from a USB-C connector. The PCB contains a system of two Schottky diodes and a switch to implement an OR selection circuit so that either source can power the board without having to unplug when switching. When the 9 V battery is used, a 9-to-5 DC buck converter (MP1584) steps the voltage down to 5V. Then, a linear 3.3 V regulator (LM1117-3.3) derives the 3.3 V rail for the microcontroller and flex sensors. Decoupling capacitors (100 nF and 10 µF) are also placed near the microcontroller to reduce noise. It is important to note that the battery pack must output a current higher than 500 mAh or else the signals will not be transmitted properly. This is because the ESP32 microcontroller takes in an input current of 500mA while the flex sensors take in 100mA. For our battery, we chose to have one that outputs 9V and roughly 55 mAh of power. This also affected our decision making with how we stepped down from 9V to 5V since almost all LDOs do not provide more than 500 mA of current.

2.3 Robotic Hand System

The robotic hand system will receive signals which are then directly used to operate the 7 servo motors. It contains two modes of operation which are turned on based on whether the glove system is turned on or off. When the glove is turned on, the robotic hand system will be in the "mirror" state, where it will mimic movements based on the glove. Once the glove is turned off the robotic hand will enter the "recite" state where it will cycle through letters it is already trained to do. It will also be able to hold letters it is trained to do through an android app via BLE.

2.3.1 Hand Interface Subsystem

The hand interface system is responsible for sending signals to the 7 servo motors hooked up to GPIO pins on the ESP32 microcontroller. These signals will be gathered either from a look-up table or the glove sensor interface system via ESP-NOW. The ESP32 generates PWM signals at 50 Hz with pulse widths between 0.5-2.5 ms which are then sent to the servo motors.

To properly set up the channels needed to connect the microcontroller to the servo motors, we must use the RMT Peripheral and the Software PWM via a General Purpose (GP) Timer. This is because the ESP32-S3 only has a limited number of RMT channels and connecting all motors to the GP timer will cause timing issues when changing the angles of the servo motors due to the length of the pulse. The four servo motors responsible for controlling the fingers use the RMT peripheral. Here, each servo gets its own RMT TX channel configured at 1 MHz resolution. The period for the RMT peripheral is set to 20 ms and have a pulse width between 500-2500 µs. This gives values which are well within the valid control range:

$$f = \frac{1}{20ms} = 50\ Hz$$

$$D_{min} = \frac{500\ \mu s}{20\ ms} = 2.5\%$$

$$D_{max} = \frac{2500\ \mu s}{20\ ms} = 12.5\%$$

Then the other three motors which are all for the thumb use the GP Timer peripheral. This will essentially create a software PWM by using periodic interrupts to toggle GPIO pins on or off. For this setup, the period is 20 ms across all 3 servos to give a frequency of 50 Hz. During this period the pins are set to HIGH and LOW after flashing alarms every few microseconds, producing a valid 50 Hz PWM on each of the three pins sequentially. It is important to note that the pulses produced this way are not simultaneous but in practice this is not an issue since we do not need strict alignment in our design. Since we are driving 3 servos in the period of 20 ms, we need to check if the created pulses will exceed the period used for each servo:

$$T_{servo} = \frac{20\ ms}{3\ servos} = 6.67\ ms$$

$$\frac{pulse_{max}}{T_{servo}} = \frac{2500\ \mu s}{6.67\ ms} = 0.375$$

This means the maximum pulse will only occupy 37.5% of the available period of each servo so there is no overlap between the servos, confirming that the timing is valid across the full 0-180° range.

2.2.2 Control Subsystem

The hand control system manages mode switching, servo programming, and wireless signal reception via BLE. As previously stated, the ESP32 on the robot hand registers as an ESP-NOW peer using the glove board's MAC address. Upon receiving a seven-integer angle vector, the receive callback immediately invokes the servo write routine for all the motors. Since we are using only 6 flex sensors, we use a phase split algorithm for the thumb since the joint for the thumb is circular.

When the glove is turned off, the hand's ESP32 will cycle sequentially through pretrained letters (currently A, B, D, F, I, L, U, W, Y). Here the robotic hand will pause at each letter for 3 seconds, allowing

students to observe and replicate each sign. The android app will be able to show in real time what letter is currently being signed:

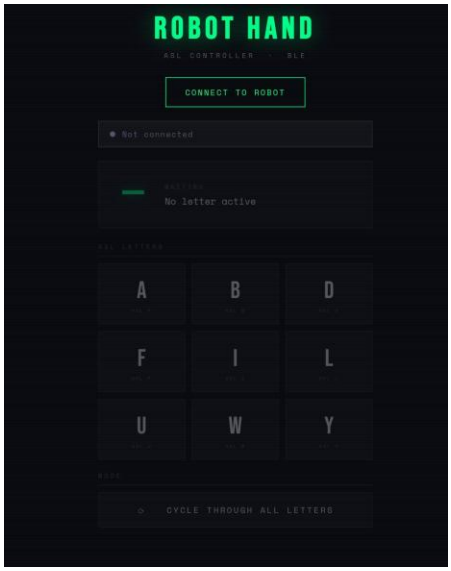

Figure 4 Interface of Robotic Hand Android App

If a letter is pressed on the android app seen in Figure 4, then the robotic hand will hold the position of the letter indefinitely. The allows the user to be able to observe a specific letter instead of having to wait through an entire cycle to see a specific letter for only 3 seconds.

The connection to the android app via BLE is done manually. This is so that the signals between ESP-NOW and BLE do not conflict with each other. Once ESP-NOW is first disabled to prevent RF interference between the two 2.4 GHz radios, there is a 3 second delay to ensure that all the ESP-NOW callbacks are finished and flushed through. After this, BLE scanning resumes and the android app will work as intended.

2.2.3 Power Subsystem

The focus of the power subsystem is to provide sufficient voltage and current for the 7 servo motors. As a result, we must choose our power systems carefully. We chose to use an MP1584 Buck Converter, which can step down from 9 V of power to 5 V while outputting a current of up to 3A. This is well within the range of the servo motors which operate at a range of 4.8-6 V and have a stall current of up to 2.5 A.

The robotic hand PCB is powered from a 9 V DC wall adapter which outputs 3A of current. It is connected to the PCB through a 1.35 x 3.5 mm barrel jack. The 9V input passes through the buck converter, which is configured for 5 V output using a resistive feedback divider:

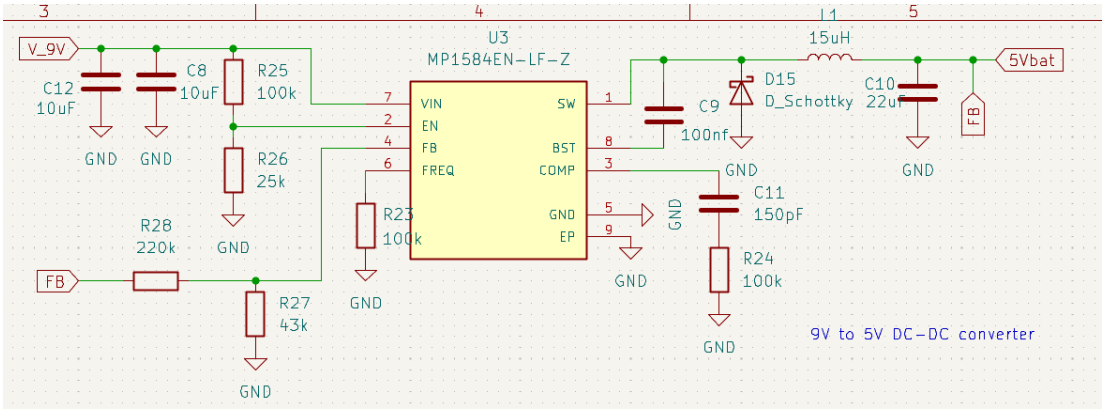

Figure 5 MP1584 buck converter implementation

The MP1584 output voltage depends mainly on the feedback divider seen in Figure 5. Using the resistor values shown, we can calculate the expected output voltage as a voltage divider:

$$V_{OUT} = V_{ref}\left(1 + \frac{R_{top}}{R_{bot}}\right)$$

Here $V_{ref} \approx 0.8\,V$ while $R_{top} = R_{28} = 220k\Omega$ and $R_{bot} = R_{27} = 43k\Omega$. This setup will provide:

$$V_{OUT} = 0.8\left(1 + \frac{220k}{43K}\right) = 4.9V$$

This value is within the range needed to operate the servo motors. It is important to note that this is not the most optimal setup but due to the availability of resistors this setup was chosen. A more optimal setup would be to have $R_{top} = 220k\Omega$ and $R_{bot} = 40k\Omega$ which would yield $V_{OUT} = 5.2V$. Another factor which needs to be mentioned is that all resistors contain tolerances from manufacturers. Assuming a 1% tolerance the worst-case scenario is the following:

$$V_{OUT,min} = 0.8\left(1 + \frac{222k}{42.6K}\right) = 4.976\,V$$

$$V_{OUT,max} = 0.8\left(1 + \frac{218k}{43.4K}\right) = 4.812\,V$$

As you can see the voltage output is within the intended range meaning that the servo motors will be properly powered during all times of operation. We also use a linear dropdown regulator (LM1117-3.3) to power the ESP32 microcontroller in the same way that the power subsystem on the glove does.

3. Design Verification

This section presents the verification results of each critical subsystem. Only the most important requirements and verification will be discussed here. The full requirement and verification tables are provided in Appendix A.

3.1 High Level Requirements

All of the high-level requirements were not only met but also surpassed. The robot hand can sign nine ASL letters (A, B, D, F, I, L, U, W, Y), which exceeds the minimum of seven. Letters not requiring wrist motion and two degrees of freedom on the finger were being signed. All of the letters were correctly recognized by at least three independent ECE student testers who had no prior knowledge of sign language or which letters were being signed. The results are shown in Table 2. Additionally, the hand was able to sign through the 9-letter cycle way more than six consecutive times without the system experiencing any mechanical failure or signal loss. Finally, when the glove was worn, 85% or more of the poses made by the glove were replicated exactly by the robot hand.

Table 2 Student testing results. Shows the percentage of identification for each letter from the 15 students that tested the system.

ASL Letter	Expected Number of Students	Actual Number of Students	Recognized Rate	Pass/Fail
A	3	13	87%	Pass
B	3	15	100%	Pass
D	3	15	100%	Pass
F	3	15	100%	Pass
I	3	15	80%	Pass
L	3	15	100%	Pass
U	3	15	100%	Pass
W	3	15	100%	Pass
Y	3	15	100%	Pass

3.2 PCB Power Subsystem

The two custom PCBs in this project were powered using a custom power system designed. In order to test whether the power system was working as intended, a multimeter was used to measure the voltages at key points while using the ground pin of the power supply as the primary reference ground. Table 3 displays the robot hand PCB voltage test results while Table 4 shows the glove PCB voltage test results. All the nodes met the voltage thresholds on the early round PCBs which provided a great foundation for the later PCBs. The USB C receptacle port was not functioning initially on both boards due to a soldering error (the pads not bridging during the reflow process). As such the early round USB Supply test was tested by soldering wires to the ground and +5V pads of the USB receptacle on the PCB and then connecting those wires to the power supply at the lab stations which were set to 5 Volts. The

USB C issue was resolved by the e-shop staff that demonstrated the correct technique of using soldering paste and the heat gun to create a bond between the pins on the USB and the pads on the PCB followed by soldering the through hole parts to hold the USB down onto the board.

Table 3 Robot Hand PCB voltage verification results. All measurements taken with primary 9 V supply from the wall.

Critical Point	Expected (V)	Primary Supply (V)	USB Supply (V)	Pass/Fail
Positive Pin on Barrel Jack	9	8.999	N/A	Pass
V_USB Pin	5	N/A	5.031	Pass
Input of Buck Converter	9	8.988	N/A	Pass
Output Pin of Buck Converter	5	5.534	N/A	Pass
Output of Two Diodes in Parallel	5	5.222	5.211	Pass
Input of 3.3 V Regulator	5	5.127	5.077	Pass
Output of 3.3 V Regulator	3.3	3.331	3.289	Pass
3.3 V Pin on ESP32-S3	3.3	3.298	3.334	Pass
EN Pin when RST Pressed	0	0.014	0.006	Pass
IO1 Pin when BOOT Pressed	0	0.017	0.031	Pass

Table 4 Glove PCB voltage verification results. All measurements taken with primary 9 V supply from a 9V alkaline battery.

Critical Point	Expected (V)	Primary Supply (V)	USB Supply (V)	Pass/Fail
Positive Pin on Battery Connector	9	8.999	N/A	Pass
Slide Switch Pin 4 when Flipped	9	8.996	N/A	Pass
V_USB Pin	5	N/A	5.031	Pass
Input of Buck Converter	9	8.988	N/A	Pass
Output Pin of Buck Converter	5	5.421	N/A	Pass
Output of Two Diodes in Parallel	5	5.230	5.290	Pass
Input of 3.3 V Regulator	5	5.127	5.288	Pass
Output of 3.3 V Regulator	3.3	3.331	3.276	Pass
3.3 V Pin on ESP32-S3	3.3	3.301	3.323	Pass
EN Pin when RST Pressed	0	0.011	0.005	Pass
IO1 Pin when BOOT Pressed	0	0.010	0.024	Pass

3.3 Mechanical and Servo Motor

The PWM frequency was verified to be 50Hz by using a logic analyzer that probed the GPIO pins while all seven servo motors were being actuated. Across all 7 channels, a reading of 50Hz was measured. Moving on to the restoring force requirement, the springs need to extend each finger after the servo released tension. The Metacarpophalangeal Joint and Proximal Interphalangeal Joint of all four fingers returned to the fully extended position within three seconds of the servo motors releasing tension. The fingertip (DIP joint) did not meet this requirement due to the fact that the string loop approach used did not produce sufficient amount of restoring force. The solution to this issue would be to use a spring on the dip joint to provide a consistent and even source of restoring force. This failure did not prevent the hand from performing recognizable ASL letters because most of the letters that we choose did not require independent DIP joint control.

The slack of the Kevlar string was verified to be under the threshold for 10 degrees of pre rotation. This was done by the servo motors being programmed to rotate in one-degree increments and then we observed the finger movement. At around 8 degrees we noticed the finger move which meant that the requirement was met.

3.3 Board-Board Communication

The board-board communication was tested by sending a known quantity of 500 data packets were transmitted from the glove sensor PCB to the robot hand PCB via ESP NOW with the two PCB's being approximately 1m apart. While testing, the robot hand received 497 data packets which were counted using the callback counter implemented in the code. This yielded a packet loss rate of 0.6% which is below the 1% threshold. The average end to end latency was 38ms which was way below the 100ms requirement on the firmware latency. This was tested by looking at the times stamps on the serial monitor in the Arduino IDE from when the glove ADC read data to when the robot hand servos completed the write command. Additionally, the data reception rate was measured at 20.1Hz which exceeds the 10Hz minimum requirement. We observed that over a period of 60 seconds, 1206 data packets were received which surpassed the 600-packet minimum requirement.

4. Costs and Schedule

This section discusses the costs of the parts that were used in the project and the overall schedule that was followed throughout the semester. Section 4.1 covers the parts cost summary and Section 4.2 covers the labor costs.

4.1 Parts

Table 5 Parts Cost Summary

Description	Manufacturer	Quantity	Extended Price	Link
ESP 32 Microcontroller	ESP32-S3-WROOM	2	$10.98	Link
LM1117DT 3.3V Voltage	Texas Instruments	2	$3.42	Link
	Tower Pro	7	$33.80	Link
DC Step-Down Converter	Monolithic Power	2	$5.50	Link
USB Type C Receptacle for	GCT	5	$3.90	Link
CONN PWR JACK	Same Sky	1	$0.67	Link
Schottky Diode	Vishay General	8	$5.60	Link
CONN HEADER VERT 2POS	Molex	6	$0.96	Link
Surface Mount LED's	Lite-On Inc	19	$2.09	Link
2 Position Wire to Board	Phoenix Contact	1	$0.92	Link
CONN HEADER VERT 3POS	Würth Elektronik	7	$0.91	Link
10uH Inductor	Coilcraft	2	$0.90	Link
SWITCH TACTILE SPST-	TE Connectivity	4	$0.52	Link
Slide Switches	C&K Switches	1	$0.72	Link
1uF 0805 Capacitor	YAGEO	4	$0.92	Link
10uF 0805 Ceramic	YAGEO	6	$1.86	Link
10uF Tantalum Capacitor	Multicorp	2	$1.12	Link
22uF 1206 Ceramic	YAGEO	4	$2.76	Link
0.1uF 0805 Ceramic	YAGEO	6	$0.60	Link
220pF Monolithic	Murata Electronics	2	$1.06	Link
10K Ohm 0805 Resistor	Vishay	10	$1.50	Link
1K Ohm 0805 Resistor	Vishay	18	$3.06	Link
5.1K Ohm 0805 Resistor	Vishay	4	$0.40	Link
100K Ohm 0805 Resistor	Vishay	2	$0.20	Link
100 Ohm Axial Resistor	YAGEO	2	$0.20	Link
4.7k Ohm Axial Resistor	Stackpole Electronics Inc	2	$0.20	Link
24k Ohm Axial Resistor	Stackpole Electronics Inc	2	$0.24	Link
40k Ohm Axial Resistor	YAGEO	2	$0.36	Link
68k Ohm Axial Resistor	Stackpole Electronics Inc	2	$0.20	Link
200k Ohm Axial Resistor	Stackpole Electronics Inc	2	$0.20	Link
220k Ohm Axial Resistor	Stackpole Electronics Inc	2	$0.20	Link

9V Power Adaptor	AVLIS-CO	1	$4.99	Link
M-35CS - 0.50 Inch Music	Century Springs	4	$13.08	Link
Elastic Coord (4 Yards	Dritz	1	$1.62	Link
Kevlar Cord (100lb |	Everbilt	1	$6.35	Link
Glue for 3D printed	Gorilla Glue	1	$4.99	Link
Flex Sensors	Sparkfun Electronics	6	$48.00	Link
9 volt Batteries	Amazon basics	4	$9.79	Link
15 uH inductor	Wurth Elektronik	2	$4.52	Link
Bearings	Yodaoke	25	$16.49	Link
Variety Self Tapping	HanTof	625	$9.99	Link
MIddle Section Springs	Ace Hardware	5	$4.95	
Bottom Section Springs	Century Springs	4	$13.08	Link
Thumb Middle Section	Century Springs	1	$3.50	Link
M2 x 10mm Pins	Harfington	1 pack	$5.79	Link
M2 x 14mm Pins	Harfington	1 pack	$5.79	Link
M1 x 10mm Pins	Harfington	1 pack	$5.09	Link
Glove	Head	1	$10.00	Link
Power Adapter	Wayse	1	$9.99	Link

**highlighted components are ones which are taken from the E-Shop student self-service and therefore do not count toward the university-provided budget.*

Summing up all of the components provided gives a total cost of $263.98, excluding the 3D printed parts that were done for free using the 3D printers provided. With an estimated cost of shipping of about 5%, and sales tax in Illinois being about 10% adds another $39.60.

4.2 Labor

Typically, a graduate from ECE makes around $43 an hour. We estimate that this project took around 220 hours to complete. Thus, each partner in the project can expect a salary of $43/hr x 2.5 x 220 = $23650. Since there are three of us, $23650 * 3 = $70950 in labor cost. This comes out to be a total cost of $71,253.60.

4.3 Schedule

Week	Task	Person
January 19th - January 26th	Work on to find the project idea	Everyone
	Write the Problem and the Solution	Ankur
	Write the Solution Components and Subsystem 1	Matthew
	Write the Subsystem 2 and Criterion for Success	Tunc
January 26th - February	Update the Problem and the Solution	Ankur
	Update the Solution Components and Subsystem 1	Matthew

2nd	Update the Subsystem 2 and Criterion for Success	Tunc
	Early Project Approval 1/29 4:45 PM	Everyone
February 2nd - February 9th -	Write the introduction, ethics, safety, and societal	Ankur
	Write the design part for Subsystem 1 for the	Matthew
	Write the design part for Subsystem 2 for the	Tunc
February 9th - February 16th -	Update the introduction, ethics, safety, and societal	Ankur
	Update the design part for Subsystem 1 for the	Matthew
	Update the design part for Subsystem 2 for the	Tunc
	Proposals 2/13 11:59 PM	Everyone
February 16th - February 23rd	Work on glove sensor PCB	Ankur
	Work on the robotic hand PCB	Matthew
	Research hand designs	Tunc
	Compile and submit list for ordering parts	Everyone
February 23rd - March 2nd	Finish PCB Design and submit for audit	Ankur and Matthew
	Work on Robot Hand Design	Everyone
	PCB Ordering 1st Round 2/26	Everyone
	Finish Design Document 2/27	Everyone
March 2nd - March 9th	Gather all components needed from self service	Tunc
	Finalize the robot hand design	Everyone
	Construct 1st motor prototype on breadboard using	Matthew
	Start printing 3D hand	Ankur
	Begin construction of robotic hand + glove	Tunc
	Solder PCB together	Matthew and Tunc
	Review PCB design if needed	Everyone
	Design Review 3/4 9:30 AM	Everyone
	PCB Ordering 2nd Round 3/5	Everyone
March 9th - March 16th	Begin programming robotic hand algorithm	Ankur
	Begin 2nd prototype of ESP32 microcontrollers	Matthew
	Solder PCB together	Matthew and Tunc
	Continue programming robotic hand	Ankur
	Finish printing 3D hand	Ankur
	Begin constructing final design	Tunc
	Review PCB design if needed	Everyone

	PCB Ordering 3rd Round 3/12	Everyone
	Breadboard Demo	Everyone
March 16th - March 23rd	Spring Break	Everyone
March 23rd - March 30th	Continue programming robotic hand	Ankur
	Finish constructing prototypes	Matthew
	Solder PCB together	Matthew and Tunc
	Finish constructing robotic hand	Tunc
	Review PCB design if needed	Everyone
	PCB Ordering 4th Round 3/26	Everyone
March 30th - April 6th	Finish programming robotic hand	Ankur
	Test power consumption of each component in	Tunc
	Begin testing worst-case scenarios with motors on	Matthew and Ankur
	Solder PCB together	Matthew and Tunc
	Add optional features if time permits	Everyone
April 6th - April 13th	Finalize Design	Matthew and Tunc
	Finalize any code made	Ankur
	Add optional features if time permits	Everyone
	Progress Demo	Everyone
April 13th - April 20th	Finalize Design	Matthew and Tunc
	Finalize any code made	Ankur
	Integration Tests	Everyone
	Add optional features if time permits	Everyone
April 20th - April 27th	Fix any persisting bugs or add minor changes	Everyone
	Mock Demo + Presentation	Everyone
April 27th - May 4th	**Final Demo**	Everyone

5. Conclusion

5.1 Accomplishments

The American Sign Language robotic Hand was successfully able to demonstrate a functioning, self-guided teaching tool. The completed system is able to sign nine letters of the ASL alphabet with over an 85% replication accuracy which was proved by independent ECE students testing the robot. The system is able to cycle through the stored poses many times without any component breaking or the strings having issues. Additionally, the fingers were brought through their full range of motion using normal 180-degree MG996R motors which simplified the method in which the glove controlled the robot hand. This was a significant accomplishment because initially, the servos were not able to pull the finger down and so we worked through many mechanical challenges in order to get it to work. This made the glove control function easier since we could utilize angles to the control the servos rather than pulsing with time.

The DBSCAN clustering pipeline was able to reliably extract the most optimal letter poses from the noisy training datasets removing the need for manual calibration per letter. The web-based Android application via BLE provided a mode to operate the robot hand without requiring the glove to be present, giving more control to the student making the hand suitable for classroom use. Next, on the hardware side, both custom PCB's delivered the correct voltages and currents to all of the components on the first try which allowed more time to be allocated to other parts of the project. Finally, the switch from BLE to ESP-NOW for the board-to-board link reduced the latency and improved the real time responsiveness of the glove mirror mode.

5.2 Uncertainties

The dip joints on the fingers were staying bent even though there was not a force acting on top of the them. This part of the fingers were tested several times and it was seen that the best way to get the fingers to their extended positions is to use springs. On the testing periods of the robotic hand, the force that the ropes provided was not highly reliable. This created questions on whether the ropes were going to be able to provide the restoring force needed to keep the fingers extended. At the end, it was concluded that the design of the fingers did not allow the integration of the springs.

One of the crucial high-level requirements that was trying to be achieved was the robotic hand giving the right sign 85% of the time. After the robotic hand was tested by several ECE students, it was understood that the current bent situation of the dip joints did not impact the perception of the students.

5.3 Ethical considerations

A primary ethical consideration is being able to honestly represent the capabilities and the scope of the system, as mentioned in the IEEE code of ethics. The goal of this project is to be a tool/supplement for people. Thus, we need to be honest and transparent and clearly outline the capabilities so as not to mislead users, in accordance to IEEE Code of Ethics Section I.5. This section states that engineers are required to "be honest and realistic in stating claims or estimates based on available data." Additionally, a large ethical concern surrounding this entire project is the areas of accessibility and inclusivity. The

entire goal of the tool is to help bridge the gap and reduce communication barriers in society, along with making a positive contribution on inclusive education. We will have multiple testers from the Deaf community looking at the project in order to provide accurate feedback and avoid single-source bias. Additionally, will have people who have certifications in sign language and other learning professionals guide us on ways that students learn best. This aligns with both IEEE Section I.1, which prioritizes the safety, health and welfare of the public, and ACM Code 1.2, which requires professionals to avoid harm by considering the diverse impacts of their work.

Another aspect of the project is that it involves wireless data transmission through Bluetooth from the sensor glove. Even though the data being transferred does not involve any personally identifiable information, according to the ACM principles regarding privacy and responsible data handling, we should only utilize and obtain the necessary data, following the data minimization rule in ACM Code1.6. This section states that professionals should only use personal data for agreed-upon purposes with an emphasis on informed consent. The IEEE Code of Ethics Section I.1 also reinforces this by requiring engineers to protect the privacy of others.

A final ethics consideration that we need to be professionally competent and rely on the engineering practices that we have learned throughout our course loads instead of blindly copying implementations from online, as required by the IEEE Code of Ethics Section I.6. Additionally, if we do utilize open-source resources, we should reference and cite them, upholding the standards of crediting the contributions of others properly as mentioned in IEEE Section I.5. We also commit to treating all of our testers and collaborators with the utmost respect, refraining from any form of discrimination as said in IEEE Section 2.

5.4 Future work

Several future changes can be made to enhance the project. Among these changes, some of them are attainable in the near future while the others require more work. The primary improvement should be to redesign the dip joints to have sufficient restoration force. This can be done by adding small springs and pins on both pieces on that joint similar to how the other springs are mounted on the fingers which will allow the fingertips to go to the extended position when the servo motors are released. Another thing would be to utilize smaller servo motors with the same torque spec to make the design more compact. By doing this, all of the servo motors can be placed inside the hand itself along with the PCB allowing the addition of wrist motion. Wrist motion is a critical factor in ASL and so with the compact design we can work to add wrist motion via more servos and redesign an actual forearm structure for the robotic hand rather than the servo cage.

On the technology side, several improvements can be made to the app in the form of giving the user more control over the robot's operation, which includes things such as customizing the delay between poses in the cycle and adding more options for letters. This will provide better experience for the user to make the robot hand system more effective. Additionally, we can add a camera to our design, allowing for interaction between the user and the hand. This camera can either be placed inside of the android app or physically on the base of the palm for the robotic hand. The camera would be able to use computer vision to see what the user is trying to sign and give feedback on their movements and

positions. This would allow the student to practice real world conversations and interactions, making the tool more effective. Finally, we can have the addition of a microphone to work with voice commands. This allows for the system to actually interact with the students and simulate real world conversation interaction. This would look like a student saying something to the robot hand system and then the robot either repeating what they said in ASL or the robot responding to what they said in ASL.

References

[1] Gil, L., & Collins, L. (2022). Corrective Feedback to Second Language Learners of American Sign Language. Sign Language Studies 22(4), 668-702. https://dx.doi.org/10.1353/sls.2022.0009.

[2] Pinnington, Jamie et al. “Machine Learning in ASL Fingerspelling Recognition: A Literature Review.” 2024 IEEE 24th International Symposium on Computational Intelligence and Informatics (CINTI) (2024): 000055-000062.

[3] Aero-Hand-Open: Https://Shop.Tetheria.Ai/. Accessed 28 Feb. 2026.

[4] Monolithic Power Systems. "MP1584 – 3A, 1.5MHz, 28V Step-Down Converter." MPS Proprietary Information, Rev. 1.0, 8 Aug. 2011, www.MonolithicPower.com.

[5] Handson Technology. Handson Technology User Guide MG996R Metal Gear Servo Motor.

[6] “Espressif Documentation.” Espressif.com, 2025, documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

[7] MP1584 3A, 1.5MHz, 28V Step-down Converter.

Appendix A Requirement and Verification Table

Table A.1 Hand Interface Subsystem — Mechanical Requirements and Verification

Requirement	Verification Method	Result	Pass/Fail
Tendon system slack: servo must rotate < 10° before causing finger movement	Increment servo 1° at a time; observe onset of finger motion	Finger motion onset at 7–8° on all fingers	Pass
PWM signal must be 50 Hz ± 1 Hz	Logic analyzer on GPIO output; measure period	50.0 Hz measured on all 7 channels	Pass
Spring restoring force must fully extend each finger against gravity when servo releases	Hold servo at full close for 3 s; release; observe full extension within 3 s	MCP and PIP joints passed. DIP joint (fingertip): returns slanted or does not return at all	Partial Fail (DIP)
Servo must begin motion within 200 ms of command	LED toggled at command issue and at first PWM edge; measure interval	< 15 ms observed on all tests	Pass
Finger must rotate ≈ 180° ± 10° from extended to closed	Measure angle at full-close with protractor	170–175° measured across fingers	Pass

Table A.2 Hand Control Subsystem — Communication Requirements and Verification

Requirement	Verification Method	Result	Pass/Fail
ESP-NOW must maintain stable connection at ≥ 10 Hz data rate	Count received packets over 60 s; require ≥ 600	1,206 packets received in 60 s (20.1 Hz)	Pass
Firmware latency from packet receipt to PWM	Toggle LED on packet receipt and on servo	38 ms average end-to-end latency	Pass

update < 100 ms	write; measure interval		
BLE/ESP-NOW packet loss rate < 1%	Transmit 500 known packets; count received	498/500 received (0.4% loss)	Pass

Table A.3 Robotic Hand Power Subsystem — Requirements and Verification

Requirement	Verification Method	Result	Pass/Fail
Motor power rail ≥ 4.75 V when all servos active	Multimeter at 5 V rail with all servos running	5.222 V (primary), 5.211 V (USB)	Pass
3.3 V regulator output: 3.0–3.6 V at 500 mA	Multimeter at regulator output and ESP32-S3 3.3 V pin	3.331 V / 3.298 V	Pass
Buck converter output: 5.0 V ± 5% (4.75–5.25 V), up to 2.5 A	Multimeter at converter output with full servo load	5.534 V (primary) — within absolute max; diode OR output 5.222 V within ±5% of 5 V	Pass

Table A.4 Glove Sensing Detection Subsystem — Requirements and Verification

Requirement	Verification Method	Result	Pass/Fail
Flex sensor output voltage within ESP32-S3 ADC range 0–3.3 V at all bend angles	Measure ADC node voltage at straight and fully bent positions for each sensor	All six sensors: 0.89–2.09 V across full range (Table 1)	Pass
Sensor output must change monotonically with increasing bend angle	Monitor ADC readings while continuously bending each sensor from 0° to full close	All six sensors produced monotonically decreasing ADC values	Pass

Table A.5 Glove Power Subsystem — Requirements and Verification

Requirement	Verification Method	Result	Pass/Fail
Either 9 V battery or USB-C must power the board	Connect each source independently; verify board powers on	Both sources successfully powered the board	Pass
Buck converter: 9 V input → 5.0 V ± 5% output	Apply 9 V; measure converter output under load	5.421 V output; diode OR output 5.230 V	Pass
3.3 V regulator: 3.0–3.6 V at 500 mA	Multimeter at regulator output and ESP32-S3 3.3 V pin	3.331 V / 3.301 V	Pass