ACTIVE POSTURAL CORRECTION VEST DESIGN DOCUMENT

By

Jordyn Andrews

Aparna Srinivasan

Sophia Sulkar

Project Proposal for ECE 445, Senior Design, Spring 2026

TA: Frey Zhao

26 February 2026

Project No. 65

1. Introduction

1.1 Problem

Poor posture is a widespread and persistent problem, particularly among people who spend long periods sitting at desks for work or school. Hours of slouching and improper spinal alignment can gradually lead to musculoskeletal imbalances, chronic neck and back pain, shoulder strain, and even reduced breathing efficiency. Because these effects develop over time, poor posture is often ignored until it causes significant discomfort or long-term health complications.

Current solutions do not fully address the root of the problem. Traditional posture braces passively hold the body in place, but they often do all the work for the user, which prevents the user from actively engaging and strengthening the muscles needed to maintain proper posture independently. On the other hand, posture-monitoring devices that only send alerts or reminders rely entirely on the user to consciously correct themselves each time. This can be easy to ignore and does not provide physical assistance in forming better habits. Existing products either over-correct by removing user effort or under-support by offering no physical correction at all.

The core problem, therefore, is the lack of an active posture-correction solution that both assists the user physically and encourages their own muscular engagement.

1.2 Solution

We propose an Active Postural Correction Vest. Unlike passive braces, this system uses an active electromechanical feedback loop to physically retrain the user's posture, then letting go of the tension so that good posture is maintained by the user, not just the device itself.

The device itself consists of a wearable vest equipped with stretch sensors which attach to elastics. These sensors continuously monitor how much the elastics are extended. When the system detects a "slouch" state (shown by the stretch sensor reading shifting away from the calibrated threshold), the central PCB triggers a high-torque servo motor mounted on the back plate. The servo reels in a cabling system made of elastic connected to the shoulder straps, physically pulling the user's shoulders back into a proper position. Once the sensors detect that the user has returned to the correct posture, the servo releases tension, allowing for natural movement and self-maintained posture until the next slouch event. In terms of safety precautions, we plan to create an assistive device that does not use a lot of force, so it cannot cause any damage. We also are going to have an emergency stop button as well as an auto shut off when the resistance level reaches a level that is too high. We also will filter out noise by adding a timer that only activates the motors if the person is sitting in a slouched position for a prolonged time

The Bluetooth app will consist of a calibration setting, a posture log, and a mode toggling capability. For the calibration setting, each user can calibrate the sensors on the vest to their own posture and slouch positioning to start. In the posture log, users can see what their posture activity looks like each day, and they can see how the device is helping them improve their posture. The mode toggling capability of the app will be between the active mode, where the device will release the tension after correcting the user's posture, and brace mode, where the device will keep its tension after correcting the user's posture.

1.3 Visual Aid

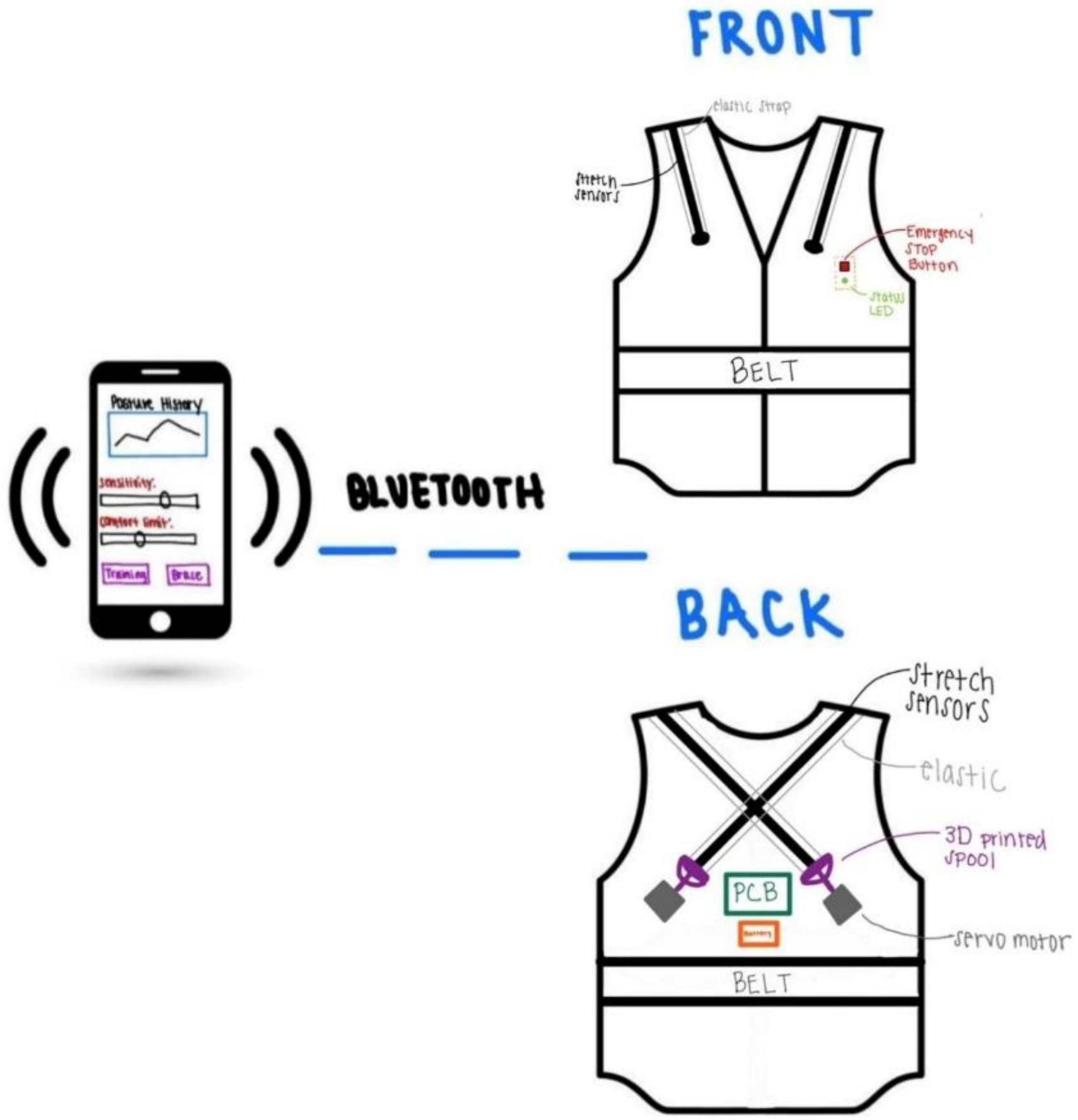

Figure 1: *Simple sketch of the physical design of Posture Correction Vest*

Figure 1 provides a high-level view of how the Active Postural Correction Vest is worn and how its main components interact during use. The system is split into (1) a user-facing interface for setup and feedback, (2) a front-facing safety/indicator interface, and (3) a back-mounted sensing + actuation assembly that performs the actual posture correction.

As shown by the Bluetooth-connected phone interface, the user will interact with the vest through a mobile app that supports calibration and monitoring. The app displays posture history and allows the user to adjust settings such as sensitivity/threshold behavior and select operating modes (e.g., "training/active" vs. "brace/hold" behavior). This provides a way to personalize the device without adding complexity to the vest itself.

The front view of the vest illustrates the wearable form factor and safety feedback. Adjustable straps and a belt provide fit and stability, while a status LED gives quick system feedback (powered/connected/active states). An emergency stop button is placed on the front for immediate access. When the button is pressed, it disables the actuation system to ensure the user can instantly stop correction force at any time.

The back view of the vest highlights the core assembly responsible for detecting and correcting posture. Stretch sensors integrated with the elastic straps measure strap extension and provide a real-time indication of the user's posture relative to a calibrated "good posture" baseline. When a sustained slouch condition is detected, the central PCB commands the servo motor(s) to reel in a cable/elastic path using a 3D-printed spool, gently pulling the shoulders back toward proper alignment. Once the sensors indicate the user has returned to the target posture, the system releases tension (in active mode). This allows for natural movement and encourages the user to maintain posture until the next slouch event.

1.4 High-Level Requirements

- The system must provide active postural restraining by applying corrective tension for 10 seconds until the user returns to a calibrated upright position, where the tension will be released after 10 seconds of consistent proper posture.
- The system must be able to distinguish between poor posture and natural movement by utilizing a time delay of 30 seconds after sensing a slouch event so that it is only triggered by a prolonged period of slouching.
- The vest must support daily tracking of user behavior with a wireless interface and a mobile app which will be calibrated for the user and allow them to visualize their postural habits and to adjust their preferred mode (brace or active).

2 Design

2.1 Physical Design

For our vest, we will be making a tight-fitting vest with an adjustable belt so that it is snug to the user's body in order to get the most accurate readings possible. The entire system consists of the vest, a PCB, a battery pack with a BMS, 2 servo motors, a 3D printed spool, stretch sensors, elastic, an emergency stop button, and a green status LED.

On the front of the vest, the elastics will be sewn into the vest, which will act as the "anchor point" of the elastic straps. Attached to these elastic straps, we have our stretch sensors, which are made of conductive rubber cord. Like the elastics, the stretch sensors will also be anchored here. There is also a panel on the front of the vest that consists of the emergency stop button and the status LED, which is lit when the system is on. We also will have an adjustable belt, which can be adjusted using Velcro in the front of the vest, as modeled below.

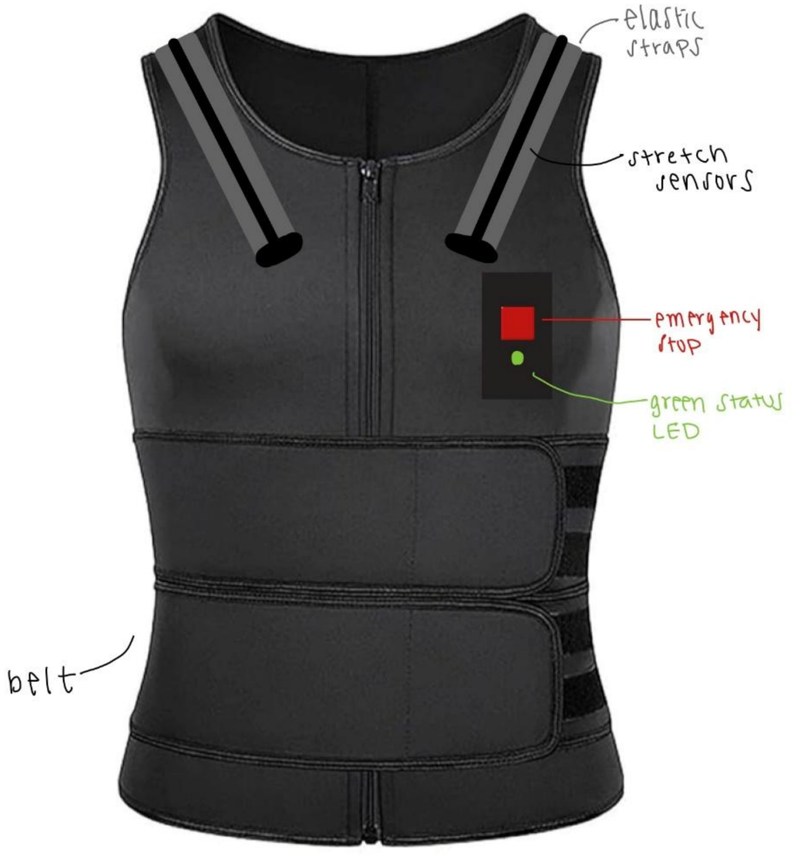

Figure 2: *Model of the Front of the Active Postural Correction Vest*

On the back of our vest, we have the mechanical portion of this vest. The elastic will attach to the 3D printed spools, which will then be controlled by the servo motors, allowing the elastic to be “reeled in” during active correction. Our PCB will also be here, as this is where many of our connections electrical connections will be. Near the PCB, we have our battery pack, consisting of the 7.4V battery and the external BMS chip. This portion of the system is modeled below.

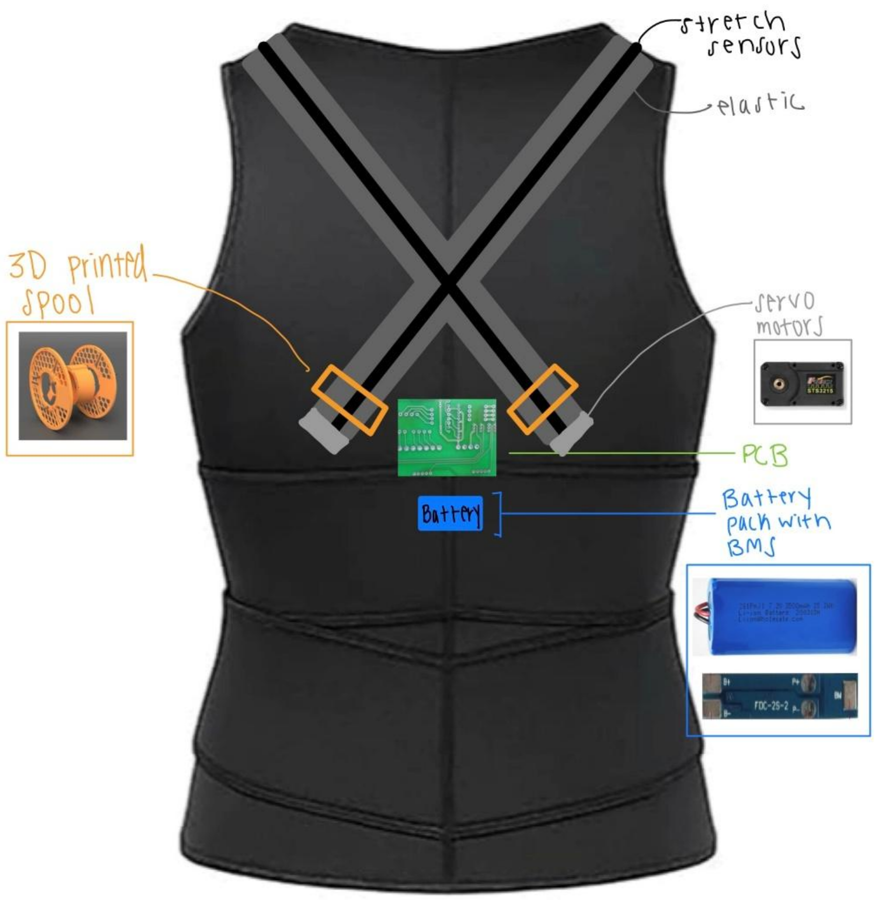

Figure 3: *Model of the back of the Active Postural Correction Vest*

2.2 Block Diagram

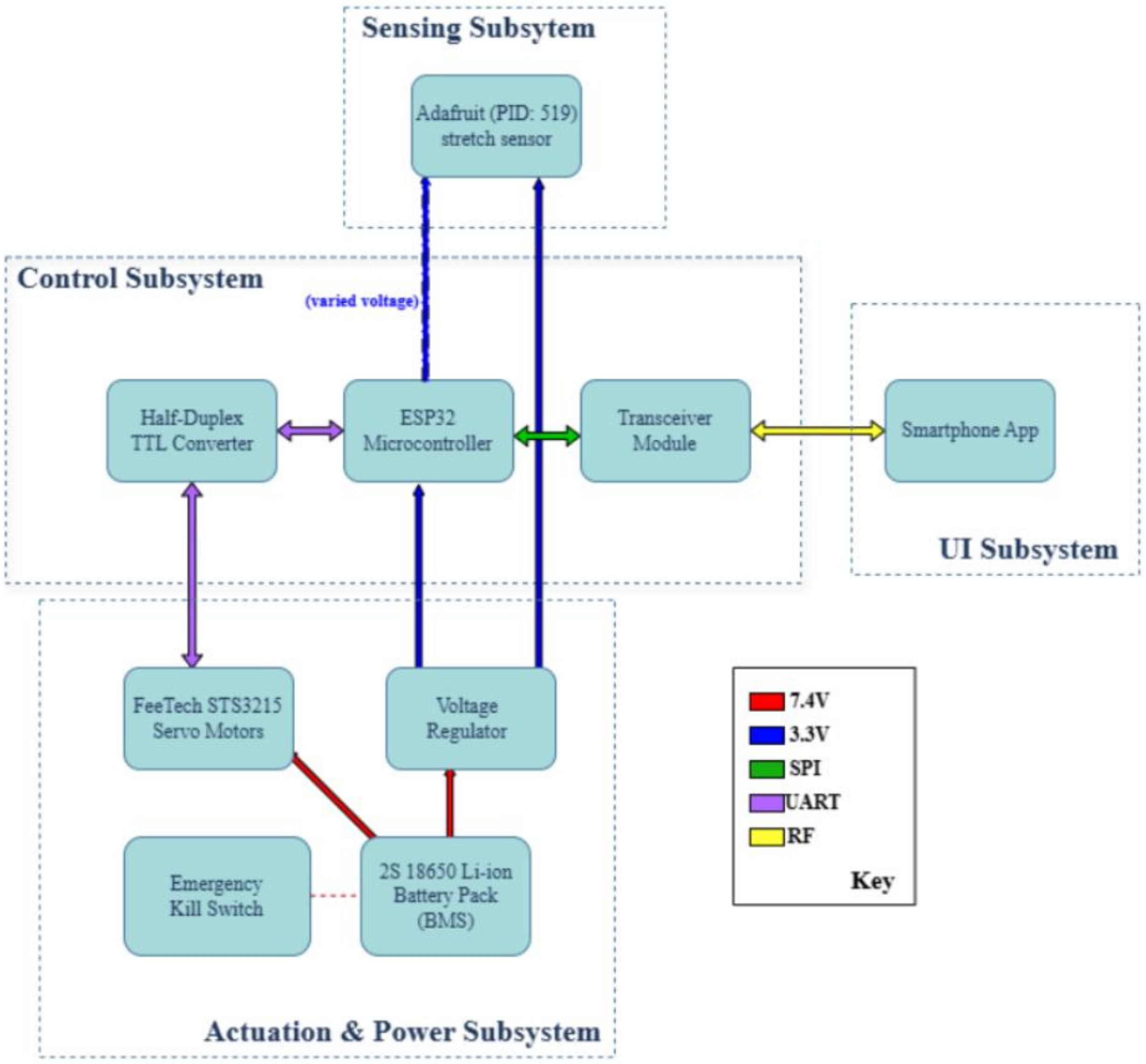

Figure 4: *Block Diagram for Hardware/Electrical Design*

2.3 Subsystem Overview

1. Actuation and Power Subsystem

The actuation and power subsystem provides all the electrical power and drives the physical correction capabilities of the vest. This subsystem consists of the motors and kill switch along with the battery pack and voltage regulator. The battery pack provides the main DC source. The hardware emergency kill switch is placed on the motor power path, so that motor power can be cut instantly if needed. The voltage regulator generates a stable 3.3V for the ESP32. This subsystem connects “upward” to the Control and Sensing subsystems via the 3.3V rail.

2. Control Subsystem

The control subsystem executes posture detection as well as safety decisions. The ESP32 reads stretch sensor voltages, applies logic to confirm sustained slouching, and issues motor commands when correction is needed. It also utilizes the smart serial servos for torque feedback and enforces limits by stopping or reversing motion when the load exceeds a comfort threshold. This subsystem connects to the Sensing subsystem via ADC inputs from the stretch sensors. It connects to the Actuation subsystem via a half-duplex UART servo bus. Lastly, it connects to the Wireless/UI subsystem via Bluetooth.

3. Sensing Subsystem

The sensing subsystem provides posture measurement signals. Conductive rubber stretch cords are mounted in the strap path so that strap extension changes their resistance. Each sensor is conditioned into an analog voltage through a voltage-divider circuit and read by the ESP32. In addition, servo load is treated as a safety measurement that can replace pressure sensors. This subsystem connects to the Control Subsystem through ADC (stretch).

4. Wireless/UI Subsystem

The Wireless/UI Subsystem provides user interaction, calibration, and logging. A smartphone app exchanges BLE packets with the ESP32 to run calibration, set detection sensitivity and comfort limits, and display posture history and events. This subsystem connects only to the Control subsystem (wireless data) and applies those parameters to sensor interpretation and actuation behavior.

2.4 Subsystem Requirements

1. Actuation and Power Subsystem

The Actuation and Power Subsystem enables the active postural restraining with the motor and battery system to apply the corrective tension for 10 seconds, then release after another 10 seconds, and ensures user safety with the emergency kill-switch. This block consists of a 2S 18650 Li-ion Battery Pack which provides 7.4V to the rest of the physical system. Along with this, we will have a 2S Li-ion Protection Board (BMS) which will act as protection for the battery pack. This is a safety regulation which will be between the battery and the rest of the system to prevent battery damage, short circuits, and fires. This battery pack is connected to the two FeeTech STS3215 Servo Motors which each require 7.4V, which are also connected to the control subsystem. The Battery Pack is also connected to the voltage regulator which is used to step down the 7.4V supply from the battery to 3.3V to connect to the ESP32 Microcontroller in the Control Subsystem and the Adafruit stretch sensors in the Sensing Subsystem. For safety purposes, we have a connection between an Emergency Kill Switch which will shut down all the subsystems in case of emergency.

Requirements	Verification
• When the Actuation and Power Subsystem receives a signal from the Emergency Stop button on the vest, the system must immediately disconnect the 7.4V battery source from all components (Voltage Regulator, Servo Motors, MCU) within 500 ms of activation in order to fully shut the system off.	• Power on the system and make sure that ESP32 and Servos are energized. • Press the Emergency Stop. • Use a digital multimeter (DMM) to measure the voltage at the input of the voltage regulator and the power pins of the servo motors. • Confirm that the voltage that is measured is 0V within the 500ms timeframe.
• The battery pack must provide a nominal 7.4V (with range from 6.0V to 8.4V) to the Servo Motors. This will enable the motors to apply corrective tension for 10s ± 5s, then release the tension for a period of 10s ± 5s.	• We must connect the battery pack and start the posture correction software up. • We will use a stopwatch to measure the duration of the active tension after a slouch event. We must confirm that this lasts in the range of 9.5s-10.5s. • Repeat this process for the release duration. Again, it should last in the range of 9.5-10.5s. • During the active tension event, we will use a digital multimeter (DMM)

	to make sure that the battery voltage remains above 6V when under load.
• The Voltage Regulator must be able to step the 7.4V battery input down to a stable output of 3.3V ± 0.5V to power the ESP32 Microcontroller and stretch sensors so they do not lose power or get driven with too much voltage.	• We must connect the battery to the input of the Voltage Regulator. • We will use a digital multimeter (DMM) to measure the output voltage at the VCC pin of the ESP32 Microcontroller and at the power rail of the stretch sensors. This reading should be in the range of 3.0V-3.5V. • This must be observed for ≥60 seconds to ensure that the system is stable and that there are no brownouts or spikes in voltage.

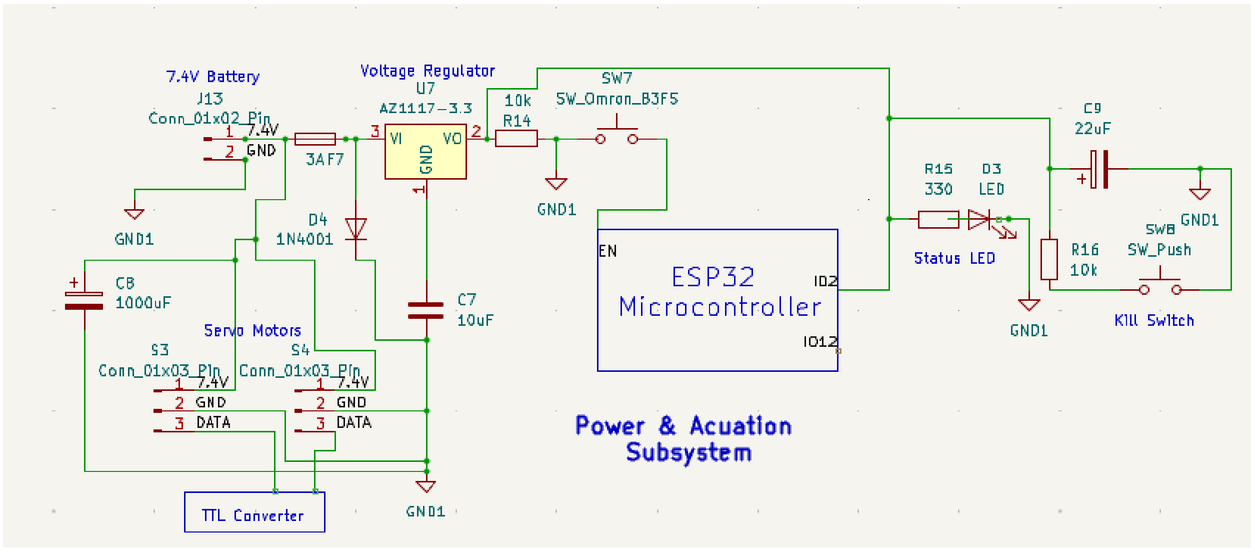

Figure 5: *Actuation and Power Subsystem Schematic*

2. Control Subsystem

The Control Subsystem implements logic for the 30 second delay time when detecting a slouch event, interprets sensor data, commands motor actuation, and enforces the limits on safety for the vest. The Control Subsystem includes the ESP32 Microcontroller, which requires 3.3V from the voltage regulator in the Actuation and Power Subsystem. This microcontroller has a built-in Transceiver module which will communicate with the microcontroller via SPI. This Transceiver Module will interface with the UI subsystem via RF, which will allow for Bluetooth connection to the mobile app. The microcontroller also must interface with a Half-Duplex TTL converter in order to communicate with the Servo motors. This TTL converter is necessary because the ESP32 uses a standard 2-wire UART protocol with TX and RX buffers, while the STS3215 servo motors that we are using require a 1-wire half-duplex data line. Essentially, the UART communication between the ESP32 and the TTL converter is full-duplex, while the UART communication between the STS3215 servo motors and the TTL converters is half-duplex, which is why conversion is necessary here. Lastly, the ESP32 interfaces with the Adafruit stretch sensors in the Sensing Subsystem via a varying voltage (which will be further explained in the Sensing Subsystem).

Requirements	Verification
• The ESP32 Microcontroller must have a non-blocking 30 second timer. After detecting a slouch event, the controller must wait for the full duration of 30 seconds before applying corrective tension. The tension should only be applied if the slouching threshold persists after the delay, so if the user corrects their posture before the 30 second timer is up, the timer must reset to 0.	• We must power up the system and connect the ESP32 Microcontroller to a PC using a USB connection • We will monitor the Serial Monitor in the Arduino IDE and will set it to 115200 baud. • We will write some code to monitor what happens when we apply a load to the stretch sensor. We will add print statements to log the passing seconds after a slouch event to ensure that the timer begins and ends when it is supposed to. • We then will apply a load to the stretch sensor, which will act as our slouch event. We will run multiple trials, which will include: • Period of “good posture”. The timer should not go off • Short period of “slouching” (<30s). The timer should go off, then stop when the slouching

	period stops within a threshold of 1s. • Full "slouch" event. The timer should count all the way from 0s-30s.
• The ESP32 Microcontroller must be able to continuously operate at 3.3V ± 0.1V. It must be able to continuously sample data from the Sensing Subsystem and manage the Bluetooth aspect, maintaining a code execution loop frequency of ≥ 50Hz.	• We will use a Digital Multimeter (DMM) to measure the voltage between the 3V3 and GND pins on the ESP32 during a full actuation cycle. We must confirm that the voltage is consistently 3.2V-3.4V. • To verify the loop frequency, we will use the difference between the current time and last execution time, which will be the period. We will then calculate frequency using f=1/T, and verify that this value is ≥ 50Hz

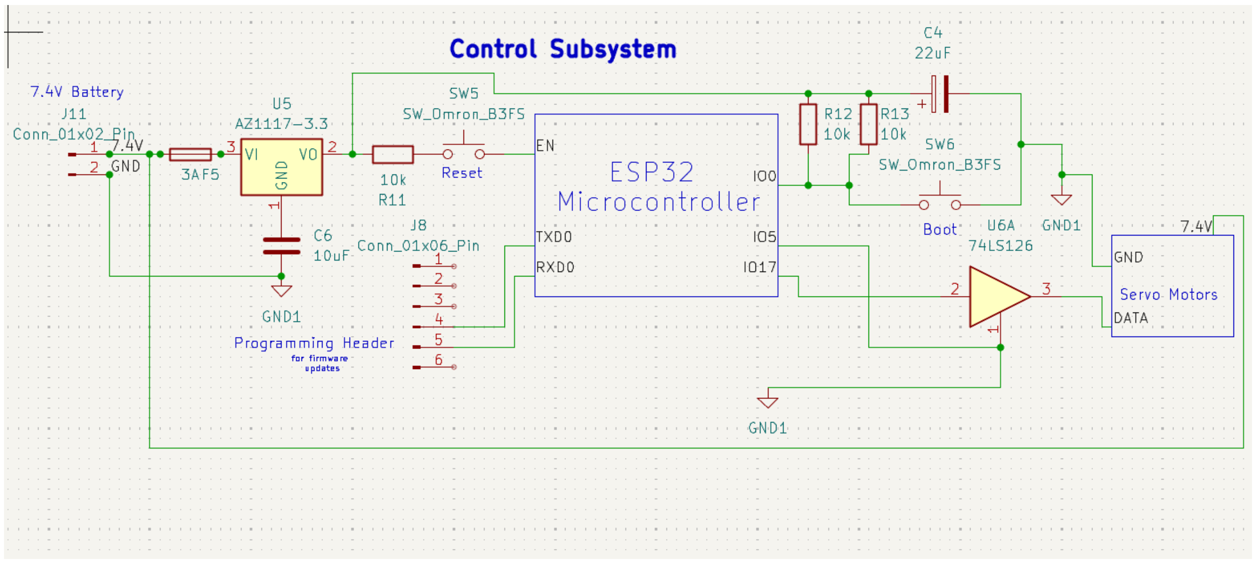

Figure 6: *Control Subsystem Schematic*

3. Sensing Subsystem

The Sensing Subsystem will distinguish a slouch event versus normal movement and will provide the signal that is measured for the delay of 30 seconds before a slouch event is acted upon. This subsystem will use a conductive rubber cord (Adafruit PID:519) which is supplied with 3.3V from the voltage regulator in the Actuation and Power Subsystem. The circuit for these stretch sensors is formed as a voltage divider using a fixed pull-down resistor. These stretch sensors communicate with the ESP32 by outputting a varying analog voltage between 0V to 3.3V directly to the microcontroller's ADC pins.

Requirements	Verification
• The Sensing Subsystem will use a voltage divider circuit in order to convert the variable resistance from the stretch sensors into an analog voltage ranging from 0V-3.3V. This signal will have to be linear enough to be able to distinguish good versus slouched posture.	• We will use a Digital Multimeter (DMM) to measure the voltage at the ESP32 ADC pin while the user is in an upright position with good posture. This voltage will be recorded as V_{good}. • We will then record a V_{slouch} value after having the user in a controlled slouch position (for initial testing, we will start with a more intense slouched position, then we will tune for lesser slouch positions). • We will then confirm that $\|V_{slouch}-V_{good}\|$ is $\geq 0.5V$, as this value should confirm that the signal from the stretch sensors is strong enough to overcome noise.
• The ESP32 must be able to digitize the analog signal from the stretch cord accurately. It is vital that the raw ADC values are visible via the serial output of the Control System to ensure that the physical connection is secure enough for data to be received.	• First we will plug the ESP32 into a laptop via USB. • Now open the Arduino IDE to the serial monitor, and set the baud rate to 115200. • When observing the ADC values, we should see numbers ranging from 0-4095. • Now, we will simulate a wiring failure by disconnecting the sensor wire. When doing this, we should see a value that is either floating or ground (0 or 4095) • We will then reconnect the sensor and confirm that the values return to the 0-4095 range again.
• The conductive rubber cord must maintain a stable resistance such that the "neutral" ADC value does not drift by more than 10% over a period of 5 minutes. This will ensure that the cord is in a good condition and provides accurate data.	• The vest will be placed on the user, who must hold a good posture and stay still for 5 minutes. • We will then log the ADC values to the serial monitor on the Arduino IDE, and observe them as they come in.

	• After the 5 minutes are up, we will calculate the variance, which should be within a 10% range of the initial reading, or the cord should be replaced.

A conductive rubber stretch sensor does not directly measure tension.
It measures change in resistance as it stretches, so to get accurate vest tension you need a calibration curve that maps sensor resistance/stretch to actual strap tension. These sensors are also not perfectly linear and can vary from batch to batch, so calibration on our actual vest is important.

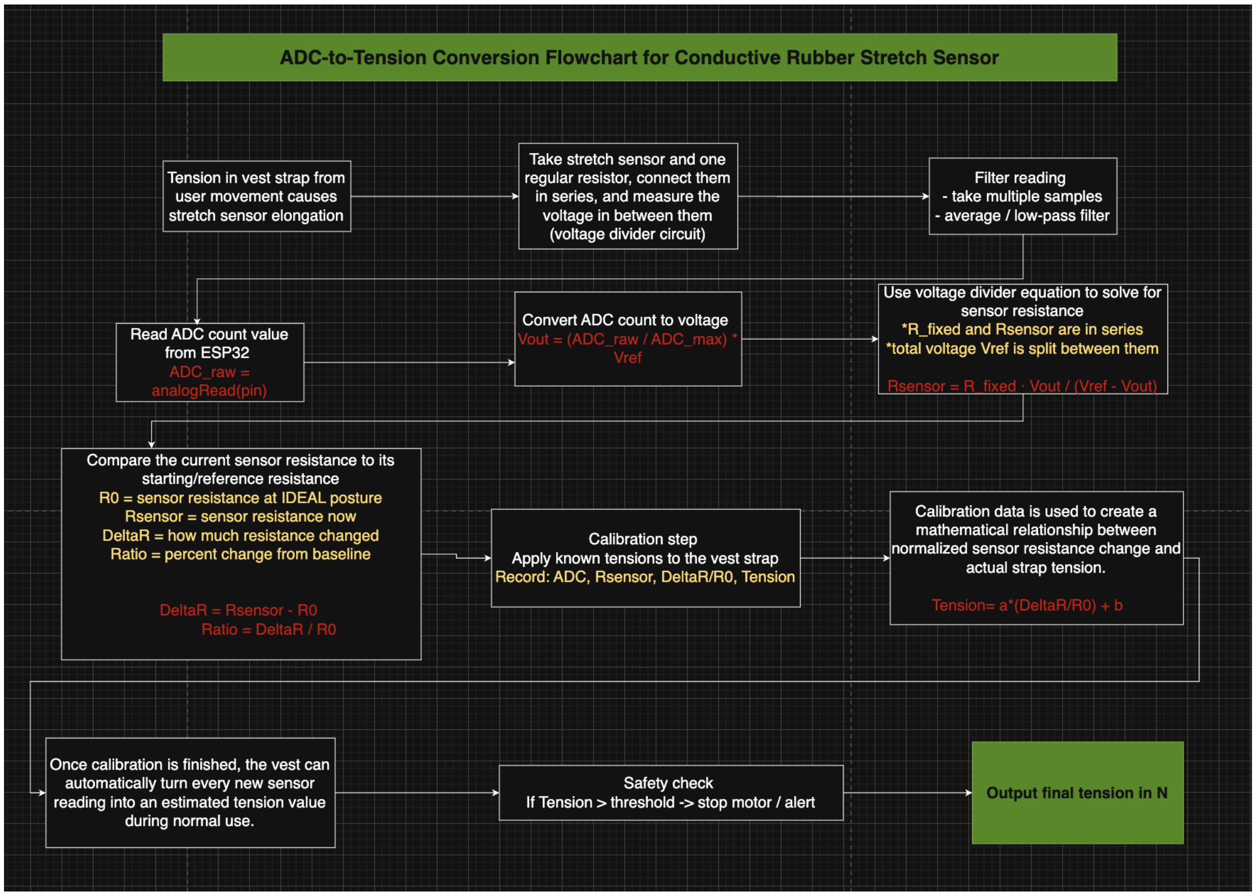
ADC-to-Tension Conversion Flowchart for Conductive Rubber Stretch Sensor
Tension in vest strap from user movement causes stretch sensor elongation
Take stretch sensor and one regular resistor, connect them in series, and measure the voltage in between them (voltage divider circuit)
Filter reading - take multiple samples - average / low-pass filter
Read ADC count value from ESP32 ADC_raw = analogRead(pin)
Convert ADC count to voltage Vout = (ADC_raw / ADC_max) * Vref
Use voltage divider equation to solve for sensor resistance *R_fixed and Rsensor are in series *total voltage Vref is split between them Rsensor = R_fixed · Vout / (Vref - Vout)
Compare the current sensor resistance to its starting/reference resistance R0 = sensor resistance at IDEAL posture Rsensor = sensor resistance now DeltaR = how much resistance changed Ratio = percent change from baseline DeltaR = Rsensor - R0 Ratio = DeltaR / R0
Calibration step Apply known tensions to the vest strap Record: ADC, Rsensor, DeltaR/R0, Tension
Calibration data is used to create a mathematical relationship between normalized sensor resistance change and actual strap tension. Tension= a*(DeltaR/R0) + b
Once calibration is finished, the vest can automatically turn every new sensor reading into an estimated tension value during normal use.
Safety check If Tension > threshold -> stop motor / alert
Output final tension in N

Figure 7: *ADC to Tension Conversion Flowchart*

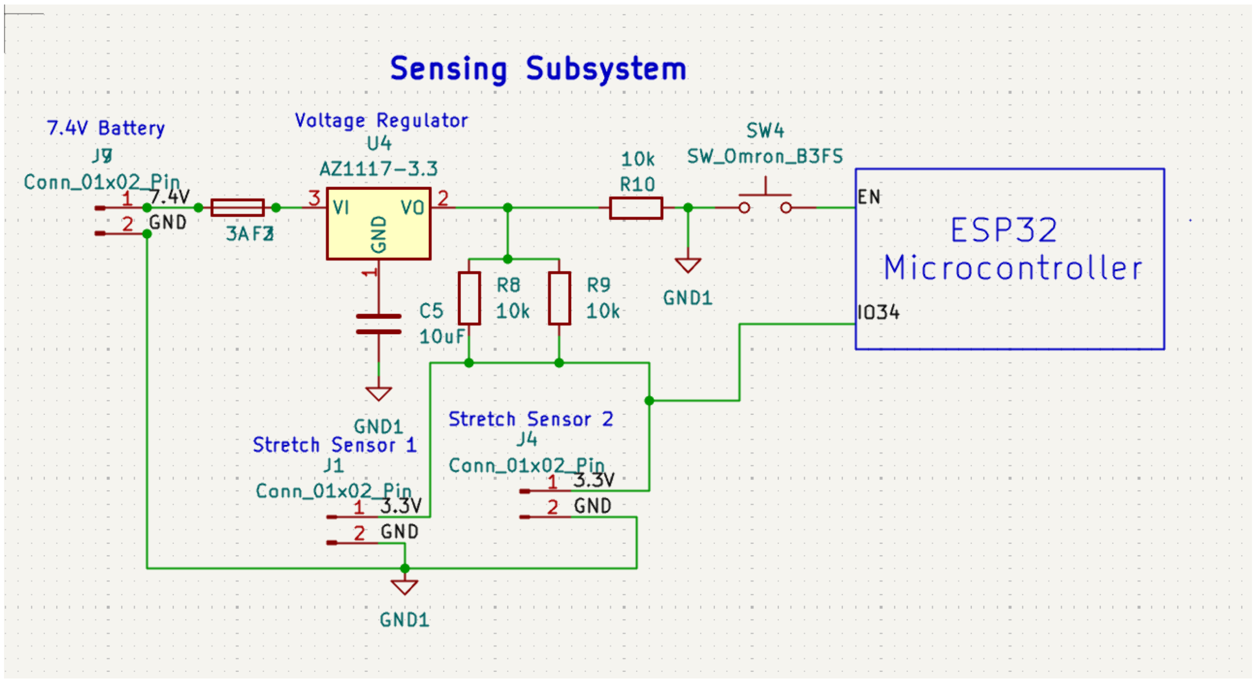

Figure 8: *Sensing Subsystem Schematic*

4. Wireless/UI Subsystem

The User Interface Subsystem overall makes long-term tracking and user-adjustable utilities available. The smartphone app will communicate with the Transceiver Module in the ESP32 in the Control Subsystem via RF, or Bluetooth, and will have a Connection Status to show the state of the Bluetooth connection. It will get data from the microcontroller for long-term posture tracking and will also allow for a brace setting to be set, which will be communicated from the application to the microcontroller.

Requirements	Verification
• The ESP32 must maintain a stable Bluetooth connection with the smartphone at a range of up to at least 2 meters. This connection must be able to support the bidirectional transfer of posture data and calibration settings with a packet loss rate of ≤5%.	• We will first power up the vest and open up the smartphone app. • We then need to plug the ESP32 into a laptop via USB and open the Arduino IDE's Serial Monitor. • We will then move the smartphone away from the vest (or the user wearing it).

	• We then will use the Serial Monitor to make sure that the phone is sending calibration data to the ESP32 by putting a print statement associated with that action. After the data is sent from the app, the message should appear on the console.
• The UI must allow the user to easily navigate to and toggle between “Brace Mode”, where motors will maintain a pre-defined tension, ignoring timer logic; and “Active Correction Mode”, where motors will trigger after 30s of slouch detected. The ESP32 must acknowledge the changing of modes via the Serial output within 500ms.	• First, we must connect the Bluetooth app to the vest and open up the Serial Monitor on the Arduino IDE, which will be getting data from the ESP32 via USB. • To test active correction, we will select this mode on the app, check the serial monitor for a message, and slouch for <30s. The motors should not move. Then the user should slouch for >30s, and the motors should move, which will confirm that the timer logic is active. • To test brace mode, we will first select the brace mode on the app and check for a message on the serial monitor. The motors should also move to the pre-defined tension value.

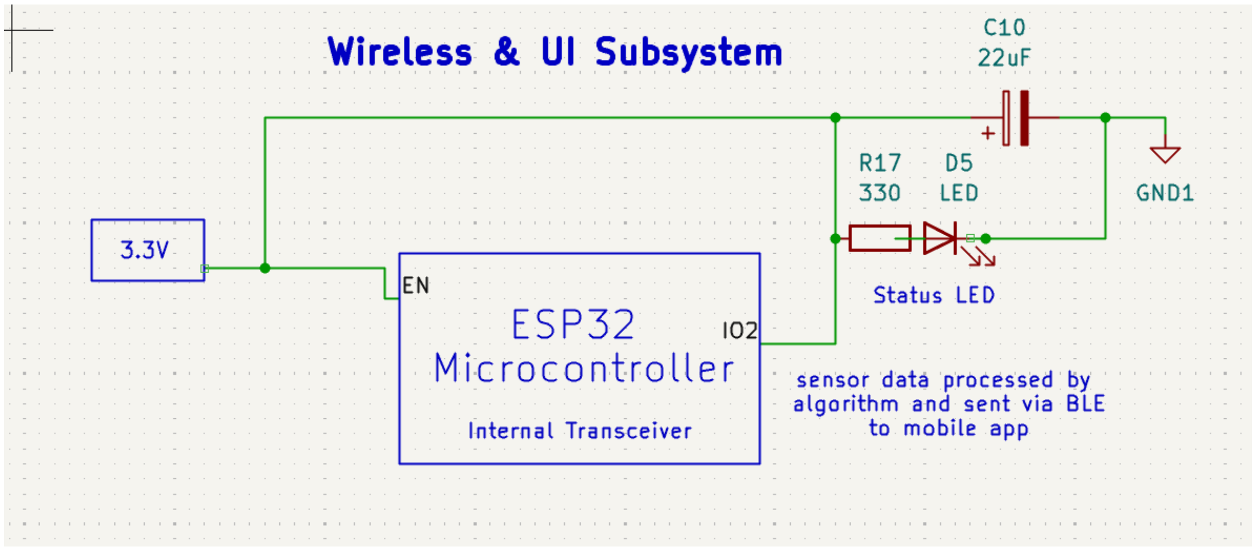

Figure 9: *Wireless/UI Subsystem Schematic*

2.5 Tolerance Analysis

A key project risk is the servo-driven spool mechanism exerting too much force and applying excessive strap tension. This could cause discomfort or potentially unsafe pressure on the user's shoulders/chest. Since the servo can generate large torque, small changes in spool radius or friction can amplify strap force. If the strap force is too high, the design would fail the requirement that the device remain safe/comfortable and stop actuation when resistance becomes too high.

The linear pulling force exerted on the user's straps is approximately:

$$F = \frac{\tau}{r}$$

where:

- $F = strap\ force\ (N)$
- $\tau = servo\ output\ torque\ (Nm)$
- $r = spool\ radius\ (m)$

From the STS3215 servo datasheet the servo's stall torque is specified as $19\ kgf \cdot cm$. Converting to SI units gives $\tau \approx 1.86\ Nm$

If we designed a 3D-printed spool with a radius of 2 cm:

$$F = \frac{\tau}{r} = \frac{1.86\ Nm}{0.02\ m} \approx 93\ N$$

A linear pull of approximately **93 N** could be too much force for a posture-correction device and could cause significant discomfort or strap failure. A typical posture brace relies on **gentle resistance**, often on the order of **20–50 N**, to cue the user to engage their own muscles rather than forcing the posture correction. To address this, our force tolerance must be enforced in software. The system will be constrained to operate within a target range of **20–40 N** of strap force. The ESP32 will monitor the servo's **UART-reported load/torque estimate** and trigger an automatic shutoff if the estimated strap force exceeds **50 N, providing** a safety margin below the actuator's maximum capability. Overall, this analysis shows the STS3215 is more than capable mechanically, and the primary design risk is limiting transmitted torque/force through software safety constraints to maintain a **50 N** maximum tolerance.

3. Cost and Schedule

3.1 Cost Analysis

The total costs are displayed in the figure below and they come to a total of $161.82.

5% shipping costs and 10% sales tax adds another $24.27.

We will expect a salary of $45/hr * 2.5 * 100 = $11,250 per team member

If we multiply this by the number of team members, which is three, it comes to a total labor cost of $33,750

Adding our equipment cost with our labor cost, our grand total comes to $33,936.09.

Description	Manufacturer	Quantity	Price	Link
ESP32-C3-WROOM-02-N4	Espressif Systems	1	$3.81 (from ESHOP)	ESP32
ESP32-DevKitC V4	Espressif Systems	1	$10.00 (from 2070)	ESP 32 Dev Kit
Conductive Rubber Cord Stretch Sensor	Adafruit	1	$19.95 (order placed)	Stretch Sensor

STS3215 19KG 7.4V Digital Servo with Magnetic Encoder & TTL Bus Control 360°	FeeTech	2	$40.88/ea (order placed)	STS3215 Servo Motor
2400mAh 2S 7.4V RX LiPo Battery Pack with JST-SYP Plug	Gens Ace	1	$27.89	7.4V Battery Pack
Voltage Regulator - AZ1117CD-3.3TRG1 (TO252-2)	Diodes Incorporated	1	$0.27 (from ESHOP)	Voltage Regulator
Capacitor - 0.1µF / 25V (0805)	Murata Electronics	5	$0.35/ea (from ESHOP)	0.1 uF Capacitor
Capacitor - 10µF / 20% / 10V (0603)	Murata Electronics	5	$0.21/ea (from ESHOP)	10 uF Capacitor
Switch - Tactile	Same Sky (Formerly CUI Devices)	3	$0.10/ea (from ESHOP)	Tactile Switch
Resistor - 10KΩ 1%(1/8W) (0805)	YAGEO	10	$0.10/ea (from ESHOP)	10k Resistor

BMS CHIP	DIANN	1 (pack of 10)	$7.99	BMS Chip
6 position MTA100 Header	ECE supply center	3 (from ECE supply center)	$0.34/ea	6 Position MTA100 Header- ECE Supply Center
22AWG Solid Yellow Hook-Up Wire 25ft	ECE supply center	1 (from ECE supply center)	$5.03	22AWG Wire- ECE Supply Center

3.2 Schedule

Week	Task
February 23rd - February 27th	- Finish KiCad Schematic & PCB layout Draft (Aparna) - Order E-shop parts (Sophia) - Fix high level requirements (Jordyn) - Finish Design document (Everyone)
March 2nd- March 6th	- Start board assembly (Everyone) - Work on design review presentation (Everyone) - Confirm all ordered parts and revise materials if needed (Sophia)
March 9th – March 13th	- Finish Board Assembly (Everyone) - Finalize PCB Design and pass audit (Everyone) - Begin initial electrical testing of power and sensor connections (Everyone)
March 23rd – March 27th	- Print first version of 3D-printed prototypes (Sophia) - Assemble physical vest and sew components together (Jordyn & Aparna) - Check fit and placement of electronics on vest (Everyone)
March 30th – April 3rd	- Finalize 3D prints (Everyone) - Continue mechanical design by integrating stretch sensors into elastic straps (Everyone) - Secure wiring and component placement on vest (Everyone)
April 6th – April 10th	- Test ESP32, ADC, and stretch sensor readings (Everyone) - Verify PCB functionality and debug hardware issues (Everyone) - Begin servo communication setup (Everyone)
April 13th – April 17th	- Connect and test servo actuation (Everyone) - Integrate servo with ESP32 control logic (Everyone) - Begin safety testing for force/tension limits (Everyone)
April 20th – April 24th	- Complete full system integration (Everyone) - Run full demo rehearsals (Everyone) - Finalize presentation and speaking roles (Everyone)

	- Perform final testing and debugging (Everyone)
April 27th – May 1st	Final Demo
May 4th – May 7th	Final Paper

4. Discussion of Societal Impact, Engineering Standards, Ethics, and Safety Considerations

4.1 Ethics

As engineers with a goal to produce a positive impact, we want to approach this from a professional standpoint as this can affect everyday lives. Under the IEEE Code of Ethics**,** our first obligation is to protect and uphold public health and safety (IEEE 1.1). In terms of our posture vest, this involves acknowledging that we are placing a motorized system onto a human body. To avoid ethical breaches, we are prioritizing a human-centered design, as this device is meant to aid, and is not a replacement for muscular effort. It is not meant to be a device used by workers whose job requires an unrestricted range-of-motion, or by those that have serious musculoskeletal injuries that require the intervention of medical help. We have a professional duty to be honest and realistic in the claims we make, which is why we will specify the audience that should and should not use this device (IEEE 1.5). It would be unethical to market this product as a medical solution for scoliosis or chronic injury, and instead, we will clearly define it as a training tool for behavioral correction. Additionally, since the vest utilizes a microcontroller for data logging, we are bound to handle user data with integrity (IEEE 1.1) and ensure that sensitive information about an individual's physical health remains private and secure by locking that information to the user only (AKA moderators have no access). Since we're putting a motorized system directly on a human body, we have a huge responsibility to make sure it's safe and reliable. We added a 74LS126 half-duplex buffer to handle the data between the ESP32 and the FeeTech servos specifically to keep the system stable. This follows the IEEE 1.5 code of ethics because it's a deliberate design choice to prevent electrical noise from the motors from messing with our sensor data or causing the software to glitch. By isolating the sensitive logic signals from the heavy power side of the motors, we're prioritizing a design that won't do anything unpredictable while someone is wearing it.

4.2 Safety

Regarding safety and regulations, the project sits at the intersection of wearable technology and assistive devices. While the FDA's General Wellness Policy provides a pathway for low-risk devices like ours to bypass the rigorous 510(k) clearance required for Class II medical devices, we still aim to align with IEC 60601-1 standards for basic safety and essential performance. This includes rigorous testing for electromagnetic compatibility to ensure we don't interfere with other vital electronics. On a local level, we are adhering to the University of Illinois Division of Research Safety (DRS) protocols, specifically concerning the thermal management of Li-Ion batteries and the structural integrity of our vest and other components. To mitigate the risk of mechanical over-extension, we are integrating physical limiters that prevent the servo motor from ever pulling beyond a safe anatomical range, regardless of what the software instructs. In terms of the specific engineering safety rules we are following for the battery, we are building in a hardware-based Emergency Kill Switch. Instead of just relying on software, this switch uses a hardware interrupt to bypass the main code entirely. If something goes wrong, the user can hit the button to immediately shut down the 74LS126 buffer, which kills the servo motor torque instantly, regardless of what the microcontroller or sensors are doing. We will also be setting up safety guidelines for testing that will be documented in the lab notebook, as well as providing a thorough safety user manual for the final Active Postural Correction Vest.

4.3 Societal Impact

The impact of this vest extends beyond the individual user. Socially and globally, it addresses the problems that many white-collar and office workers have with posture and "Tech Neck", potentially lowering the long-term economic burden of musculoskeletal injury and strain. By choosing materials like breathable mesh and natural fibers, we aim to both fulfill our environmental responsibility to minimize e-waste and provide a comfortable solution for all users. Another way we are making sure that the building of the vest is environmentally responsible is that we designed a modular power setup using a 2S 10A BMS and XT30 connectors so that if a part breaks, you can just swap it out instead of throwing the whole PCB and vest away. This helps cut down on e-waste. We also plan to keep our slouch-detection algorithms open-source so this can be a low-cost and accessible tool for any who deals with chronic posture-related pain or "tech-neck."

4.4 Detailed Safety Procedures

We will set up a few layers of protection to keep ourselves safe while building this and to keep the user safe during the final demonstration. To prevent battery shorts, we are utilizing a 3A Blade Fuse right at the power input that acts as a physical backup if the electronic BMS ever fails. We also added a 1000µF capacitor to keep the 7.4V power rail steady. This prevents the ESP32 from browning out and resetting, which is an important safety concern because a reset could leave the servo motors stuck in a dangerous position. Before we test this out on a person, we're doing our testing on a mannequin-like object to make sure the voltage dividers are mapping the stretch sensor data correctly to safe motor movements.

References

[1] IEEE, “P7-8 - IEEE Code of Ethics,” IEEE, web page. Available at: https://www.ieee.org/about/corporate/governance/p7-8. Accessed Feb. 10, 2026

[2] U.S. Food and Drug Administration, “General Wellness: Policy for Low Risk Devices - Guidance for Industry and Food and Drug Administration Staff,” FDA-2014-N-1039, Sept. 2019. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-wellness-policy-low-risk-devices. [Accessed: Feb. 10, 2026].

[3] International Electrotechnical Commission, *Medical electrical equipment - Part 1: General requirements for basic safety and essential performance*, IEC 60601-1:2005+AMD1:2012+AMD2:2020, 2020.

[4] Division of Research Safety, “Battery Safety,” University of Illinois Urbana-Champaign, 2024. [Online]. Available: https://drs.illinois.edu/. [Accessed: Feb. 10, 2026].

[5] RobotShop, "FeeTech 7.4V 19kg Serial Bus Servo w/ Current Feedback," *RobotShop*. [Online]. Available: https://www.robotshop.com/products/feetech-74v-19kg-serial-bus-servo-w-current-feedback. [Accessed: Feb. 13, 2026].

[6] Adafruit Industries, "Conductive Rubber Cord," *Adafruit*. [Online]. Available: https://www.adafruit.com/product/519. [Accessed: Feb. 13, 2026].

[7] Shenzhen Feetech RC Model Co., Ltd., “STS3215 Product Specification,” Ed. A/1, Jun. 23, 2023. [Online]. Available: https://files.seeedstudio.com/products/Feetech/108090023_STS3215-C001_Datasheet.pdf. [Accessed: Feb. 24, 2026].