REMOTELY ADJUSTABLE CAST

By

Alice Getmanchuk,

Jack Burns,

Saloni Garg

Final Report for ECE 445, Senior Design, Fall 2022

Professor: Viktor Gruev

TA: Stasiu Chyczewski

7 December 2022

Project No. 10

Abstract

Every type of cast has its own limitations, The Remotely Adjustable Cast maximizes benefit while minimizing these limitations. It is designed to be clean, mobile, and auto-adjusting to keep you on the road to recovery. Our product is an innovation on the AirCast boot that is traditionally worn on broken limbs. Our innovation involves strap adjustment with one push of a button on our web application and assistance with correctly pumping the air cells. Additionally, all of our upgrades can be powered easily by a portable battery pack, so putting on or taking off the boot is easy and portable. The cast design was a success, The Remotely Adjustable Cast resolved the problems associated with traditional casts on the market to provide the patient with consistent doctor-prescribed healing. It is our hope that after further research about optimally using it, the Remotely Adjustable Cast leads to better patient outcomes.

Contents

1. Introduction

1.1 Problem

Several types of casts can be prescribed to patients who have broken their limbs including plaster, splint, fiberglass, and AirCasts [2]. While the general purpose of these varying types of casts remains the same – to be an assistive device in the healing process for broken limbs – they each come with different benefits and drawbacks. While plaster, splint, and fiberglass casts are sturdier on the limb, they are also bulkier and may inhibit mobility [9]. Additionally, if the inside of these casts is not cared for properly, they may develop mold since they are irremovable in nature. AirCasts deal with the issues of mobility and removability but come with the issue of improper replacement of strap tension and air cell pressure when the cast is taken off then put back on. AirCasts in general may require more visits to the doctor for progress monitoring, and improper adjustment by the patient has the potential of leading to a longer healing process.

1.2 Solution

To address the problems explained above, we designed a remotely adjustable AirCast. This solution involves providing patients with the ability to control the strap tension adjustment, using motors, and air cell pressure values, using force resistive sensors, through a simple user interface. By automating the AirCast replacement process, users are still able to reap the benefits of the AirCast in terms of removability and mobility, while ensuring that cast components are properly adjusted, such that doctor visits may be limited. The intention of this solution is to allow for doctors to input the proper strap tension and air cell pressure values into a user interface based on their own discretion. These values can then be stored, and the user is able to take the AirCast on/off using the user interface as well. Not only does this solution allow for users to keep their broken limbs clean and as mobile as possible but improves the overall experience between doctors and patients.

1.3 High-Level Requirements

1. The cast's straps are adjusted/tightened per doctor's settings without manual adjustment.
2. The doctor's cast adjustments for pressure and tightness can be stored.
3. All necessary components for auto-adjustment of the cast fit on the cast without extreme addition to the original weight of the cast.

1.4 Block Diagram

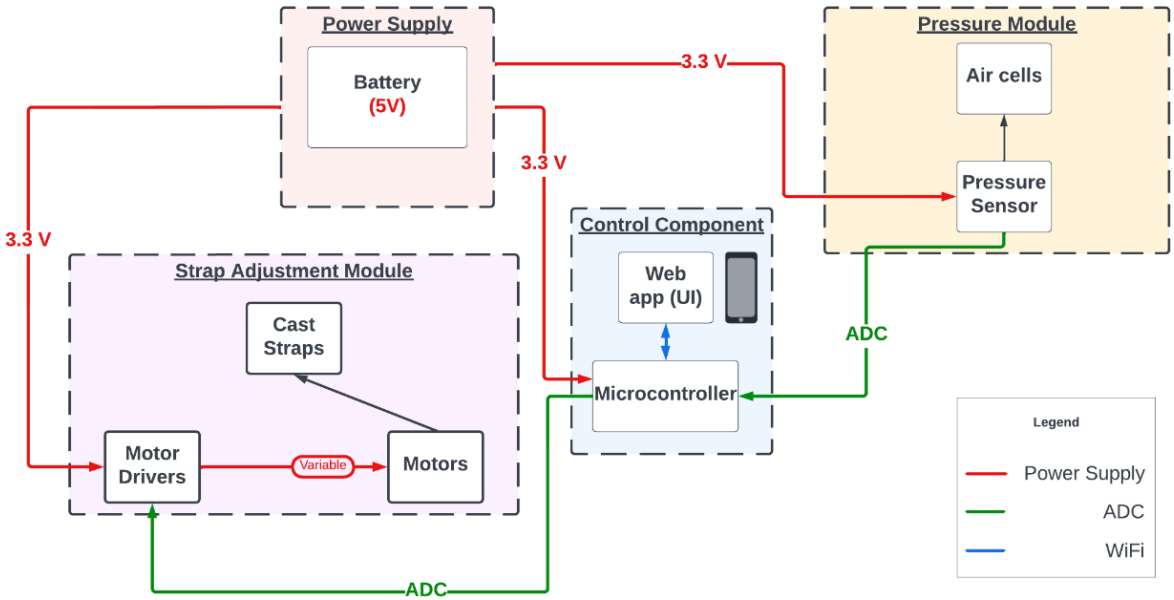

Figure 1. Remotely Adjustable Cast Block Diagram

Figure 1 is the block diagram of the various subsystems necessary for the remotely adjustable AirCast. The diagram displays the components included in each individual subsystems, as well as how components communicate with one another across and within different subsystems. Finally, the block diagram depicts how much power is being provided components of each subsystem from the original power supply unit.

2 Design

This section will go over the design procedures and details for each module of this project, including the control, pressure, strap adjustment, and power modules.

2.1 Design Procedure

The design procedure for each module will be an overview of how and why we designed each module as we did.

2.1.1 Control Module Procedure

The control module was designed with the intention of being able to adjust the boot mechanically and to communicate with the patient. We wanted to be able to trigger the boot triggering via Bluetooth initially, to be able to control the motors, and to be able to read air cell pressure pins. We ended up choosing the ESP32 WROOM32D to fit all our needs. With this microcontroller, we can utilize the ADC (analog-to-digital converter) pins in order to read and send information to the adjustment modules. We can also use the Bluetooth capabilities of the microcontroller to connect to a web application to control the status of the boot as well. Our overall goal in mind with designing this module was to try to make it as simple as possible so that it is easier for our patient. If the control component is too complex, we risk making our project too complicated (and therefore not helpful) for our users.

While more technical specifications of the control module are covered in section 2.2.1, our control module was designed with the intention of being able to interface with all other modules in our project simultaneously. For example, it is powered by the power module (cut down from 5V to 3.3V to not fry the microcontroller). It reads the pressure values constantly when the boot is powered to allow for adjustments to be made. Also, the control module instantiates the strap adjustment module which also being mounted on the boot. This module was designed to accomplish all our high-level requirements and bring together all of the functionality of the other modules.

2.1.2 Pressure Module Procedure

Our goal with the pressure module was to read the pressure of the air cells. This way we know the pressure being exerted on the patient's ankle which is critical for the boot to fit snuggly, aid in rehabilitation, and to not allow the patient to hurt themselves further. Our initial design was to somehow fit a pressure sensor *within* the air cells in order to read the pressure. This would be the most accurate way to get the pressure cell measurements from air pressure within the cell and from the external pressure exerted on the air cell by the patient's ankle. However, puncturing these air cells was out of the question. We only have 2 air cells to work with, and if we were to improperly seal or puncture the air cells, we risk losing the ability to use the air cells altogether and they are an *essential* part of the AirCast treatment. We also considered going a step further and installing pumps on the air cell ports to automatically inflate/deflate the air cells, however that was beyond the time span of the project.

We completed our design for the pressure adjustment module by settling for a non-invasive way to measure pressure. Since these air cells were removable and sat on the walls around the patient's ankle, we decided to install force sensing resistors (FSRs) between the air cell and the wall of the boot. Based on the pressure exerted on the FSR, we are able to convert its resistance to pressure since we know the

active area of the FSR, the range of Newton reading for the pad, and the linear relationship between resistance and force for this FSR. The control module is responsible for this conversion and for displaying the value of the cells on the user interface which will tell the user whether to inflate/deflate each cell more. The pressure adjustment module overall consists of the FSRs on the boot transmitting data to the control module.

2.1.3 Strap Adjustment Module Procedure

The strap adjustment module was designed to allow for patients to utilize the user interface to tighten/loosen the straps on the AirCast. The patient can click on a switch on the user interface to tighten/loosen the straps, after which the user interface will communicate with the control unit to run the motors in the necessary direction and speed to complete the action it is being asked to complete. To create this module, we had to begin by working with UIUC ECE's machine shop to have stepper motors mounted onto the AirCast itself, with a pulley through which the cast strap could be threaded.

After working with the machine shop on where to place these motors and how the strap needed to be threaded to properly run the module, we had to work on controlling the motors such that the tension they were applying on the straps as the program was running, was a value we could precisely measure and display on the user interface. This was done by finding the relationship between the motor's pulling torque based on the RPM it was running at and correlating these factors with the strap length and pulling angle to calculate the tension. The details of these calculations will be discussed further in sections 2.3.3 and 3.3.

After understanding how to control the motors such that they were performing the necessary action of tightening/loosening the straps of the AirCast, and applying the necessary tension, we had to complete the final step of providing users with the ability to tighten or loosen the straps via the user interface. This was done by implementing communication between the motors and ESP32 microcontroller via ADC pins, and between the ESP32 Microcontroller and user interface via Wi-Fi. We were able to program a switch into the user interface which communicated to the ESP32 about which ADC pins we wanted to control. This way, we were able to specify when we intended to control each motor individually and turn it off/on accordingly.

2.1.4 Power Module Procedure

The power module has one function only, and that is to power all the other modules. One of the appeals of AirCast boots is how mobile they are, and we wanted to maintain that by using a portable power supply. If the patient had to plug their boot into a wall outlet each time they needed to take off or put on the boot, that would be majorly inconvenient for them. Therefore, we decided to have the power module be a portable battery pack (like a phone power bank) that could power all other modules and also be mounted on the boot or on a belt. More technical specifications are provided in section 2.2.4.

2.2 Design Details

The design details for each module will be more technical specifications and calculations we did to design each module.

2.2.1 Control Module Details

Multiple things were considered when designing the control module: interacting with motors, interacting with pressure module, and interacting with the user interface. When interacting with the motors, we need to be able to (1) control the tension of the straps based on prescribed tensions and (2) control which motor we are tightening. For the first motor control requirement, we can control the tension that we are setting the motors to by modifying the speed at which we are running the stepper motors (more in section 3.3.1) so all we needed was our control module to have a microcontroller with ADC pins to send signals to the motors. For the second motor requirement, we need to have enough ADC pins so that we can control each motor separately (therefore 8 pins total for motor interactions). For the pressure module interactions, we needed to have ADC pins to read the resistance of the FSRs when pressure is exerted so that means we need at least 10 usable ADC pins to interact with the boot adjustment modules. The ESP32 WROOM32D was the choice we made.

For the user interface, we initially wanted to have Bluetooth capabilities to interact with a React Web App with Bluetooth in-browser functionality. In the end of our research, we decided to use Wi-Fi to interact with our web app. Now our requirements changed to be able to host a web server via our microcontroller to create a web application that can be updated with our sensor values.

The last thing we needed to do was construct a programming circuit for our ESP32 WROOM32D. This was an essential part of the control module that we initially neglected. Using the development kit bypassed this requirement for testing, but for a production grade product and being able to mount the modules on the boot we included a programming circuit to our PCB so that we can upload code onto the microcontroller via micro-USB B.

2.2.2 Pressure Module Details

As mentioned in section 2.1.2, we want to be able to measure the pressure exerted on the patient's ankle while wearing the boot after tightening the straps and pumping the air cells. Although we are not medical professionals and cannot claim that our remotely adjustable boot is a medical device yet (will require testing), we can try to make this boot as safe as possible. Therefore, we conducted research into the safe tightness around a human limb. Based on a study we found regarding external pressure and blood flow [5], the external pressure exerted on a limb should not exceed around 20 mmHg relative to the environmental pressure. This can be seen in Figure 2 below from the previously mentioned study.

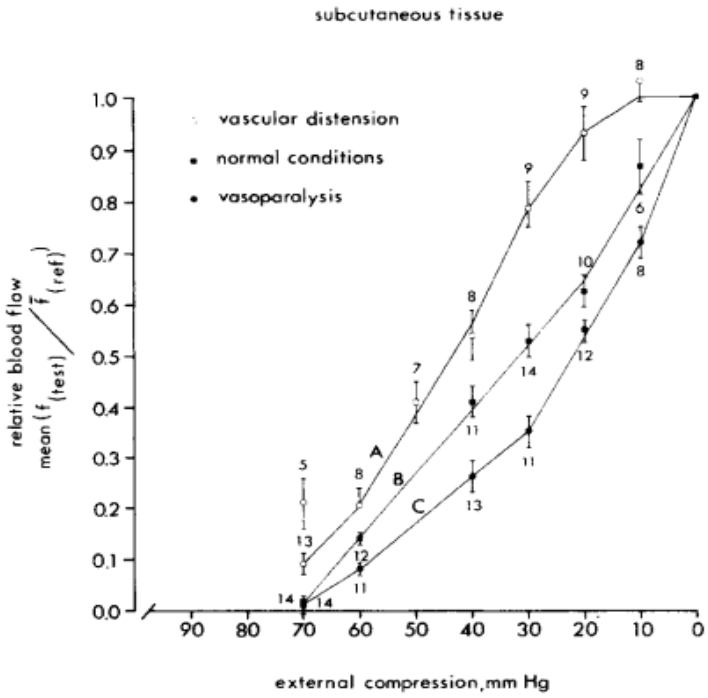

Figure 2. Relative Blood Flow vs. External Compression [5]

Based on this find, we must ensure that our pressure sensing module must be able to detect pressure of each air cell up to at least 20 mmHg reliably. We originally wanted to use small barometers inside of the air cells, but we landed on using force sensing resistors (FSRs) to calculate the pressure of each air cell as to not modify the air cells (mentioned more in 2.1.2).

Since we know the active area of our FSR (38.1^2 mm^2), and that the maximum Newtons sensed by this FSR (10 N) we can find that the max pressure that can be measured is 51.671mmHg.

$$\frac{10(N)}{38.1^2(mm^2)} * \frac{1(mmHg)}{0.000133322\left(\frac{N}{mm^2}\right)} = 51.671\ mmHg \quad (1)$$

Given that the force on the FSR is linearly related to the resistance read, we can map the resistance obtained from the FSR which is in a range of 0 to 4095 to a measurement in mmHg from 0 to 51.671 mmHg. Given the tolerances of the FSR we found that the readings we were getting would be accurate ±1.55 mmHg. This fits our constraint of reading safe pressure measurements on the ankle of the patient so that we do not over tighten the boot.

2.2.3 Strap Adjustment Module Details

Several considerations were taken into account when developing this module throughout the overall procedure. In the original design of this module, we planned to include a load cell or force sensor which would be placed underneath each of the straps on the AirCast, to measure the tension being applied by the motors as it tightened the straps. However, as we began the research and development phase of the project, we found that this tension could be calculated without the force sensors, as there was research

available correlating the motor's RPM to its pulling torque. Using this relationship, the length of the strap, and the angle the strap was being pulled at, we were able to use Equation 2, as below, to calculate the tension being applied to the straps, without the need for a force sensor or load cell.

$$\tau = F * r * sin\theta \quad (2)$$

Additionally, consistent communication with the machine shop was necessary for this portion of the project, as the stepper motors being used for strap adjustment needed to be mounted onto the AirCast directly. This meant speaking with the machine shop about where the motors needed to be placed on the AirCast, ensuring that the straps could pull at the desired angle, and that the pressure, power, or control modules would not be disrupted by the addition of the motors on the cast. After communicating with the members of the machine shop about these details, we found that the straps of the AirCast needed to be replaced altogether, to straps which could be threaded through the pulleys on the stepper motors properly.

Additionally, making sure that the ADC pins being used to connect the ESP32 to the stepper motors were usable was a vital part of the process. To confirm they were, we used the ESP32 datasheet [3] as well an ESP32 pinout reference [4], to choose the ADC pins to use and tested the signals going in and out of the motor drivers using an oscilloscope and multimeter. For this module, the L293D motor drivers were used between the stepper motors and ESP32. In the original design, we intended to use DMV8833 motor drivers, but after having issues putting the PCB together – we had to quickly switch to a stepper motor driver which was still available to us through the ECE445 lab and could be breadboarded. The signals tested to ensure the ADC pins of the ESP32 were usable, and that the motor drivers were receiving the correct amount of power were those going in/out of the motor driver, as well as into the ESP32 ADC pins.

Furthermore, the strap adjustment module required communication between the ESP32 microcontroller, user interface, and stepper motors. The implementation of this involved working with the Arduino IDE to interface between these different components. Several unipolar and bipolar stepper motor methods were available for use through the Arduino IDE within the Stepper class [11]. This made controlling the motors through the ESP32 simple, with our primary focus being on ensuring the torque and speed values correlated with the desired pulling tension force by the motors. In the final program used, the motors were controlled by setting the speed based on the torque correlation and making the motor step through 3 full revolutions at that speed to tighten the straps and move the same number of revolutions in the opposite direction to loosen the straps. While the direct implementation was simple, this module required several rounds of verification, as will be discussed further in section 3.3.

2.2.4 Power Module Details

As mentioned previously, the goal of the power module is to be able to power all other modules with a single portable battery pack. Therefore, the constraints of this power pack are that the voltage must be 3.3V or higher to power the ESP32 properly (we will cut down the voltage to 3.3V) and the power rating must be high enough for all modules (meaning we have to have enough current to power the motors, pressure module, ESP32, etc.). Based on the datasheets for the FSRs and the NEMA-23 motors, we

estimated the total current usage to be around 2A. Therefore, a traditional phone power bank which is 5V and 2A should be powerful enough to power the whole project. We also plan to include a linear voltage regulator to cut down the voltage from 5V to 3.3V so that we do not fry the microcontroller.

3. Design Verification

3.1 Control Module

Verification of the control module has a few different requirements. Verifying that the user interface is communicating with the ESP32, the ESP32 is correctly reading the analog inputs that are being communicated from the pressure module, and the ESP32 is communicating with the strap adjustment module properly. The user interface verification will be described below while the communication verifications will be described in the respective subsystem design verification sections to avoid repetition.

3.1.1 User Interface Control

The user interface is a local host server, hosted on a Wi-Fi that the ESP32 connects to. By connecting to the same Wi-Fi that the microcontroller is connected to, and accessing the server that the ESP32 is hosting, we verify that the control component can be accessed by the user. By creating a switch on the user interface that controlled an LED on the ESP32 development board we were able to verify that the switches on the user interface operated correctly and would control the motors correctly.

3.2 Pressure Module

Verification of the pressure module requires that both force sensing resistors (FSR) operate efficiently and that they are communicating properly with the ESP32. The force sensing resistors are used by the pressure module to determine the pressure that the air cells are applying to the leg of the user. Each FSR is connected to a voltage divider. The voltage divider is connected to an ADC pin of the microcontroller. Since we know the voltage before the voltage divider, we can easily determine the resistance of the FSR by reading the voltage at the division point with the ESP32 thus finding the voltage drop across the resistor, and since we know the value of the other resistor in the voltage divider, we can then determine the resistance value of the FSR. We can use the user interface to display the calculated resistance values of the FSRs. By using a multimeter to measure the resistance across the FSRs and looking at the UI displayed values, we were able to verify that the pressure module was capable of accurately calculating the resistance that the FSR was providing, and that the control module was capable of communicating successfully with the pressure module. Based on the research we found [5] we need the pressure module to apply pressure within 10-20mmHg of additional pressure. As above we know that the resistance values being measured by the ESP32 are accurate to what is being measured by the multimeter. Given the tolerances from the datasheet, the measured resistances are accurate ±1.55 mmHg allowing us to confidently land within the 10-20mmHg range.

3.3 Strap Adjustment Module

Verification of the strap adjustment module can be divided into three subcomponents, including stepper motors, motor drivers, and communication between the user interface, ESP32, and other strap adjustment module components. The verification for each will be discussed below.

3.3.1 Stepper Motor Verification

The stepper motors used for the strap adjustment module, the NEMA-23 bipolar motors in this case, included measuring the tension force being applied to the straps by the motors and ensuring enough power was being provided to the motors. The first requirement of this module was being able to calculate the tension of the straps such that the motor would only run until the holding tension value was +/- 3N of the doctor's prescribed strap tension value (Appendix A, Table 5). This was to be verified originally by reading the tension applied by the straps through a force sensor which would be placed underneath each strap. However, after finding research correlating the speed of a NEMA-23 stepper motor and its pulling torque as in Figure 3 below, we were able to eliminate the need for a force sensor in this verification process.

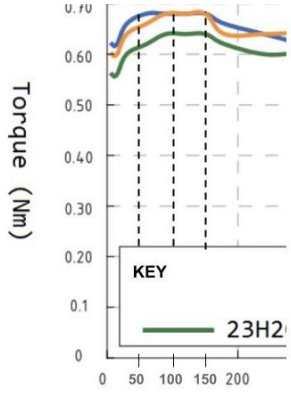

Figure 3: NEMA-23 Speed (RPM) vs Torque (Nm)

By combining information from Figure 3 and Equation 2, we calculated the tension force being applied by the motor on the straps, to verify that the applied force was within +/- 3N of the doctor's prescribed value. Furthermore, verifying that enough power was provided to the motors was calculated by probing the output of the motor drivers using a multimeter, and confirming the values against the NEMA-23 datasheet [1].

3.3.2 Motor Driver Verification

Verifying the motor driver portion of the strap adjustment module included hooking all components up to the driver correctly based on the datasheet and ensuring that power going in/out of the driver was within the range of what the motor driver could handle, while still providing the motors with enough power to run properly. Once the connections to the motor driver were verified using the L293D datasheet [10], voltages going in/out of the driver were assessed through multimeter probing – in which the voltage going into the driver was 5V directly from the power source, and the voltage going out to

the stepper motors was 1.5V per coil of the motor, aligning with the necessary power consumption for proper running of the motors.

3.3.3 Communication Verification

The final portion of the strap adjustment module to be verified was communication between the ESP32 and stepper motors. The requirement outlined at the start of the design process for this was to ensure the motor could be toggled on/off through a signal from the microcontroller. This was verified by programming the ESP32 microcontroller such that a toggle switch on the user interface would cause the speed of the motor to change from 0 to a value > 0 (Appendix A, Table 5). We were able to confirm this verification was met by demonstrating the motor tightening/loosening the straps based on the toggle of two separate switches for each action on the user interface, therefore verifying that the motor was moving at a speed greater than 0 RPM when either switch was toggled on.

3.4 Power Module

Verification of the power module was done by checking that all the components of device were receiving the correct amount of power. By using a multimeter, we were able to check that the voltage and current values throughout the circuit, and make sure that the measured values were as expected. We were able to verify that the power bank was supplying 5 volts and that the linear regulator that we chose was cutting that voltage down to around 3.3 volts. The ESP32 was receiving 3.3 volts and was grounded correctly. We were also able to verify that the motors had enough current to successfully operate. While the PCB was never fully working, the power circuitry on the board was confirmed to work properly, by probing different parts of the board we were able to verify that components were receiving the correct voltages.

4. Costs

4.1 Parts

Table 1: Parts Costs

Part	Manufacturer	Quantity	Link	Cost ($)
AirCast boot	DJO Global	1	Link	$13.37 – just shipping, preowned
ESP32 MCU Module	Adafruit	1	Link	$8.95
NEMA-23 Stepper Motor	Adafruit	2	Link	$49.90 - both
SEN-09376 ROHS Pressure Pad Sensor	Sparkfun	2	Link	$25.00 - both
USB Battery Pack	KMASHI	1	Link	$15.99
Linear Voltage Regulator - LM3940IT-3.3	Digi-Key	1	Link	$2.63
Power jack	Digi-Key	1	Link	$0.69
0.2 Ohm Resistor	Digi-Key	2	Link	$4.38 - both
ESP32 Dev Kit	Amazon	4	Link	$44.00 - all
Micro-USB B port	Digi-Key	2	Link	$1.40 - both
CP2102 – programmer	Digi-Key	1	Link	$5.06
Motor driver – L293D	Adafruit	2	Link	$17.90 - both
Total				**$189.27**

4.2 Labor

On average, University of Illinois at Urbana-Champaign Engineering graduates make around $87,000 a year [7]. This translates to $41.83 per hour for labor of one of our engineers. While our initial estimate was that this project would take around 200 hours collectively to complete, we ended up spending around 22 hours a week each in the ECE 445 Laboratory to do work for 10 weeks which is a grand total of 660 hours on the project by our group members. The duration of this project was longer than 10 weeks, however this number should account for some slower weeks and some more intense weeks. Using a 2.5 overhead factor for this project, we can calculate the total cost of labor for accomplishing our project:

$$\frac{\$41.83}{hour * person} * \frac{22\ hours}{week} * 10\ weeks * 2.5\ overhead * 3\ persons = \$\mathbf{69,019.50}$$

4.3 Total Cost of Project

The overall cost of completing of our project is the sum of the parts cost and the labor cost:

$$\$189.27 + \$69{,}019.50 = \$\mathbf{69,208.77}$$

4.4 Team Schedule

Table 2: Team Schedule Breakdown

Week	Task	Person
September 25 - 30	Talk with ECE Machine Shop about motors for strap adjustment and placement	Everyone
	Begin PCB design (list of components needed on board)	Alice + Saloni
	Sensor data ↔ microcontroller transmission design	Jack
	Complete Design Document	Everyone
October 3 - 7	Continue PCB design (& PCB Board Review)	Alice + Saloni
	Design Review with Instructor & TAs	Everyone
	Start designing strap adjustment module with motors and sensor data readings	Jack
	Soldering Assignment	Individual - Everyone
October 10 - 14	Place PCBway Orders (Need to pass audit by 10/11)	Everyone
	Teamwork Evaluation I	Everyone
	Visit Machine Shop (Revisions)	Everyone
October 17 - 28	Place Second PCB Order (with programming circuit)	Everyone
	Continue strap adjustment module with motors + Sensor data testing	Alice + Saloni

	Begin pressure sensor module sensor testing	Jack
October 31 - November 4	Create Web Server on ESP32 (WiFi not Bluetooth)	Alice
	Work on PCB and other modules	Everyone
	Individual Progress Reports	Individual - Everyone
November 7 - 11	Work on PCB board	Alice + Saloni
	Finalize strap adjustment and pressure subsystems	Jack
November 14 - 18	Finalize PCB board and all subsystems	Everyone
	Mock Demo to TA	Everyone
November 21 - 25	FALL BREAK	N/A
November 28 - December 2	Final Demo to Instructor and TAs	Everyone
December 5 - 9	Final Presentation	Everyone
	Complete Final Papers	Everyone
	Complete Lab checkout + Lab Notebook	Everyone
	Final teamwork evaluation	Everyone

5. Conclusion

The Remotely Adjustable cast was a successful design allowing users to remove the cast without compromising the consistency of their treatment. Since the design was a success, more research on how this technology affects patient recovery can be done, and the design could be further developed to be more portable and lighter for users before being released to the public.

5.1 Accomplishments

The main accomplishments are that the remotely adjustable cast fulfilled all of the high-level requirements and all of the subsystems operated as we intended them to. The cast was able to store the doctor prescribed tension and pressure values, and the strap adjustment module tightened the straps to the prescribed tension with only the click of a button on the user interface. While the motors added a larger than expected weight, the addition of all of the components required for auto adjusting the cast did not add excessive weight to the cast. The cast worked as we imagined it would. By simply clicking a button on the user interface the straps were tightened or loosened, and the pressure applied by the air cells was measured and displayed to the user. There was text on the user interface that informed the user that the air cells were inflated to the proper values. Finally, the tolerances for the tension and pressure measurements were within an acceptable range, so our measurements are accurate. Overall, the project was a success.

5.2 Uncertainties

Since we were unable to obtain small weights to test the force sensing resistors, we were not able to verify that the part was measuring force as it was designed too. We used our hands to test the sensitivity and deemed that it was accurate in determining light to hard presses and relied on the data sheet and the provided part tolerances. Another uncertainty was with our PCB and why a successfully programmed ESP32 that seemed to be soldered and powered correctly was not connecting to Wi-Fi. We programmed an ESP32 on a devkit, desoldered it from the devkit and soldered it to the PCB and measured that the ESP32 was receiving 3.3 volts and that it was grounded correctly but for some reason it was not connecting to Wi-Fi.

5.3 Ethical considerations

In terms of ethical considerations of our product, there are many. Since we are upgrading a medical device, we must consider the safety aspects of our product to the user, especially. Section I.1 of the IEEE Code of Ethics [6] says that we must “hold paramount the safety, health, and welfare of the public.” The goal of our remotely adjustable cast is to help with the rehabilitation of patients’ limbs. It is critical to not harm the user more than they already are. The main ethical concern of our project is malfunction. If the strap tightening module were to malfunction and accidentally tighten the straps too much, it could cut off the blood circulation to the foot which could extend the recovery time for the patient even more. The same goes for not tightening the straps enough and accidentally causing the patient to roll an ankle. In order to mitigate this risk, we disclose any and all possible risks to the user. We also have carefully chosen the motors so that we *cannot* provide too much tension to the bootstraps with the motor specifications.

Another ethical concern is that of data privacy and hackers. Since the user interface is storing patient medical data it is important to make sure that only the patient and doctors can access that data. Since our design has the user interface hosted on a local server connected to WiFi it is important that the user is using a password protected WiFi so that hackers can't connect to it and loosen the boot or access the medical data. A password protected personal mobile hotspot is perfect because the user can make sure that only allowed people have access to the WiFi and this is the most likely case since the user may want to take off or adjust the cast at a place that is not their home.

Additionally, while we are trying to improve upon the AirCast boot to make it an even better experience to the user, we must make sure that these added benefits are worth the added cost. Upgrading the AirCast boot to have remotely adjustable functionality comes at an increased price, however we do not want to increase the cost too much to the patient if they were prescribed it. Thankfully, if this boot were to go to production it would cost significantly less than $69,000 (or even $189) to produce each boot. The added benefits of the cast that we are providing must be substantial enough from testing to warrant the price increase that it would take for the patient.

5.4 Future work

Given the opportunity to continue working with this product, some changes would be made to ensure a cleaner better product. The PCB would be re-organized (for example, the programing circuit would be moved to the edge of the board to be more easily reached) and would be adjusted so that it works properly. A belt attachment would be made to house the PCB and power bank, so they don't have to be mounted to the cast. The cast would also have clamps attached to the side that clamp down on the straps allowing the user to turn off the motors when wearing the cast instead of relying on the holding torque of the motors. This would increase the life of the battery pack and the life of the motors. The cast would also utilize smaller motors that were out of stock since these motors would weigh less and still provide the necessary torque. If we could control the manufacturing of the air cells, we would switch back to the original design of inserting a barometer in the air cells to get a more consistent reading of the pressure. We would also like to partner with medical professionals to allow for testing of our cast to see how it affects treatment. If professions could administer tests to see what the ideal strap tension and air cell pressure is it could improve treatment for people. After this more tests would need to be done to see whether or not this cast provides a distinguishable enough increase in treatment. If it was found that it did provide a distinguishable increase in treatment, we would get final approval and move to production.

References

[1] A. Industries, “Stepper motor - NEMA-23 size with 9mm GT2 pulley,” *Adafruit Industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/5117. [Accessed: 07-Dec-2022].

[2] “Air Cast Vs. Plaster”. *Healthfully*. [Online]. Available: https://healthfully.com/air-cast-vs-plaster-6618746.html (August 24, 2022).

[3] “ESP32 Series” *Espressif Systems*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: 07-Dec-2022].

[4] “GPIO & RTC GPIO,” *Espressif Systems*. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html. [Accessed: 07-Dec-2022].

[5] H.V. Nielsen. “External pressure – blood flow relations during limb compression in man.” *National Library of Medicine*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/6659990/ (September 15, 2022).

[6] “IEEE Code of Ethics.” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (September 7, 2022).

[7] “Illini Success All Campus Report.” *University of Illinois*. [Online]. Available: https://uofi.app.box.com/s/aoply09y5kf6i36es8v3bl758n2lfl08 (December 5, 2022).

[8] “PART 814 - PREMARKET APPROVAL OF MEDICAL DEVICES”. *Code of Federal Regulations*. [Online]. Available: https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-814 (September 7, 2022).

[9] “Plaster or Fiberglass? A Guide to Casts”. *Healthline*. [Online]. Available: https://www.healthline.com/health/types-of-casts (August 24, 2022).

[10] “Quadruple half-H drivers (rev. C) - Adafruit Industries,” *Adafruit*. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/l293d.pdf. [Accessed: 07-Dec-2022].

[11] “Stepper,” *Stepper - Arduino Reference*. [Online]. Available: https://www.arduino.cc/reference/en/libraries/stepper/. [Accessed: 07-Dec-2022].

Appendix A Requirement and Verification Table

A. Control Component

The microcontroller chosen, an ESP32, will communicate with the Bluetooth/WiFi chip, force sensor for the strap adjustment module, and pressure sensor for the pressure module via UART interfaces. Additionally, the microcontroller should send signals to the motor for the strap adjustment module via I2C bus and receive sensor readings from the force and pressure sensors via UART as well.

Table 3: Control Component Requirements and Verifications

Requirement	Verification
Microcontroller implements control system for the motor of the strap adjustment module by taking inputs from force sensors and outputting motor speed. Microcontroller also translates pressure data to the web interface telling the user if they need to pump the air cell more.	If the Bluetooth/WiFi functionality of the microcontroller is enabled, the proper sensor readings from the pressure sensor and force sensors will be displayed on the web application after being read by the microcontroller. The measured force and pressure readings displayed on the app will match doctor prescription.
Microcontroller must be able to interface with a web application via Bluetooth.	Controls enabled on the developed web application can properly change functionality of the microcontroller, and therefore of the strap adjustment and pressure modules as intended.

B. Pressure Module

The pressure module will sense the air pressure inside of the air cells via a SEN-09376 barometric pressure pad sensor between the air cell and the wall of the boot. The force exerted on the pressure pad gives us a sense of how inflated our air cells are. The pressure on the pressure pad will be communicated to the microcontroller. If the stretch goal is hit it will also automatically fill the air cells up until it has reached the prescribed pressure (controlled by microcontroller).

Table 4: Pressure Component Requirements and Verifications

Requirement	Verification
Pressure pad sensor must be able to indicate to the user when they have inflated/deflated the air cells to the intended pressure, while	When the user views the web application, they should be able to view the correct pressure value of the air cells such that as they inflate/deflate the air cells using

staying within 10-20 mmHg of the local environment pressure value such that the user's blood circulation is not cut off.	the included pump, the reported value in the front-end interface displays the value changing accordingly, and a warning is shown to users if the pressure value reaches 20 mmHg above the local environmental pressure value. The local environmental pressure value will be measured by a separate pressure sensor located directly mounted on PCB.

C. Strap Adjustment Module

The strap adjustment module should be able to utilize a force sensor to find the tension of the straps on the cast and communicate this tension reading with the microcontroller. The microcontroller should in turn communicate back with the motor in this module to properly adjust the straps based on the force read, and strap tightening necessary based on the stored strap adjustment value.

Table 5: Strap Adjustment Component Requirements and Verifications

Requirement	**Verification**
The tension of the straps must be able to be calculated such that the motor runs until the calculated tension value is within the prescribed value ± 3N (estimated maximum value will be ~26N)	When the intended torque is applied to the strap of the boot via the stepper motor at a given angle, the microcontroller receives the tension read by the force sensor and stops the motor when the prescribed tension is reached, while displaying a reading that is accurate to this tension value. The maximum possible value displayed should be no more than 26N according to the maximum torque provided by the motor (~0.8 N*m) at the angle the motor will be pulling the strap at (12°).
Motor must be able to be toggled on/off by receiving a signal from the I2C bus from the microcontroller.	Utilizing the Arduino IDE compatible with the ESP32 microcontroller, the microcontroller is programmed to control the motor, such that a specific command leads to the motor speed changing from 0 to a value > 0.

D. Power Subsystem

The power supply chosen must be able to power the microcontroller as well as the pressure and strap adjustment modules. Additionally, the power supply should be easily rechargeable by the user and be placed in a safe location on the boot where physical damage cannot come easily.

Table 6: Power Component Requirements and Verifications

Requirement	Verification
Power must not exceed 3.6V ± 0.3V when feeding into the microcontroller	Measure voltage going into the microcontroller at different power supply charges to make sure this is always true
Must be able to power all chips on the board and the adjustment modules	Measure voltage input to all parts that receive power from the power supply & make sure all receive enough power to function based on data sheets