Jeff Suitor

IBEHS 3P04: Final Report

400138679

Abstract

We propose a smart remote physical therapy device for gait analysis intended for children with cerebral palsy. This project addresses the unique gap in the smart rehabilitation market and the failure of many of devices to address the psychology and motivational aspects needed for younger patients to have a successful and enjoyable rehabilitation experience. Our device is uniquely suited to those suffering from cerebral palsy which is a customer base with unique constraints that other smart rehabilitation products fail to address.

Introduction

We set out to design a remote gait laboratory that monitors improvements in walking gait patterns after tendon or muscle lengthening surgery for children with Cerebral Palsy (CP). Our design specifically had to address equinus gait in these children after surgery and throughout their rehabilitation process at home. CP is characterized as a motor disability in young children commonly associated with the equinus gait or “toe-walking” pattern. Yearly routine surgeries are often performed in order to release the tension in the Achilles tendon via the orthopaedic tendon or muscle lengthening surgery. Ideally, this surgery will promote heel strike during gait while reducing the tendency to plantarflex the foot. Gait analysis is a huge component of the rehabilitation process of regaining strength in their muscles as well as relearning how to walk and adjust to the new anatomical changes of their tendons.

The typical gait laboratories where this rehabilitation would take place are costly to setup and utilize while also not being mobile. This means that patients are limited as to where they can live because they must have access to a facility. It also means that for any progress to be monitored sessions must be booked and conducted in person. This means that stakeholders are left wondering about the child’s progress for potentially weeks or months at a time between sessions. Having access to a mobile and remote gait analysis laboratory would allow stakeholders to more accurately track progress with the constraints imposed by a typical gait analysis laboratory.

A potential solution would be the use of “Smart insoles” which are currently available to remotely monitor gait and pressure activity in the foot. These systems typically consist of pressure sensors, an accelerometer and a gyroscope. [1] The array of pressure sensors is utilized to quantify a high-resolution pressure map of the patient’s foot during locomotion. The sensors can monitor the individual’s gait cycle and the time spent in each phase. [2] The accelerometer and gyroscope collect position and movement measurements. The data is then transferred to software on a smartphone through Bluetooth to communicate important gait features to the user. The smartphone can calibrate the raw sensor data, remove noise, and then provide a meaningful analysis to the patient. [1]

However, these systems fail to address important aspects of child psychology that are critical with young patients. Motivational rehabilitation interventions are proven to lead to better outcomes for children with cerebral palsy [3] and Appendix A contains a table that demonstrates that when elements of fun and interest were incorporated there was an observed improvement in results. Additionally, statistically significant improvement of results was evident for motivational interventions in terms of ankle dorsiflexion rep completion time and gait biofeedback (i.e. physiological awareness which can be harnessed into the process of relearning how to walk).

Background Theory

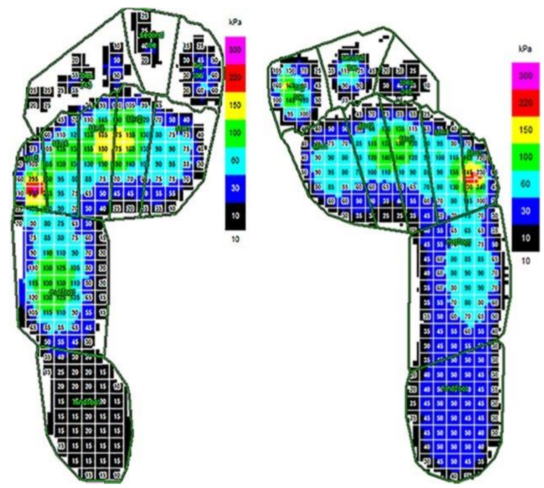

Figure 1: Pedobarograph displaying the plantar pressures during clinical gait analysis for spastic cerebral palsy [4]

Our solution aims to measure the progress post-surgery to determine the effect and level of impact that the surgery has on gait pattern for children with CP. We plan to track the progression of rehab via collection of pedobarographic data and degree of joint angles to provide constructive feedback and updates for both the patient and other important stakeholders regarding the extent of plantarflexion during gait analysis [4]. As demonstrated in Figure 1, plantar surface pressure can be displayed to relay information about contact of the feet with the ground specifically during the stance phase of gait [4]. Using force sensitive sensors, we will be able to retrieve data required to determine and measure the progressive decrease in pressure at the distal end of the foot indicating a reduction in “toe walking” and an increase in pressure at the proximal end indicating the presence of heel strike. In addition, we are hoping to track and measure the joint angle at the ankle, to measure its deviation from angles larger than 90 degrees, which is indicative of plantarflexion [4].

The gastrocnemius is a muscle found in what is typically referred to as the calf region and is the plantar flexor of the foot. The contraction of the gastrocnemius can be measured using electromyography (EMG), which measures the muscle response or electrical activity in response to a nerve’s stimulation of a muscle. [5]

Methods

Design Criteria

We decided to be able to optimally create a solution we first needed to create a very clear set of design criteria that had to be met in any design that we moved forward with. The primary design objective was accurately quantifying plantarflexion and dorsiflexion throughout the child's gait. Measuring and communicating information on ankle flexion and heel strike will allow physiotherapists and parents to monitor the progression or regression of equinus gait. Another criterion was improvement in the child's quality of life and mental state during the rehabilitation process. Children with CP require extensive daily stretching and physiotherapy appointments to prevent gastrocnemius spasticity after surgery [6]. Positive psychology concepts should be implemented within the design to motivate the child during the rehabilitation process. The third design requirement was to improve the accessibility of the patient's data and progress for important stakeholders. Physicians, physiotherapists, and parents are integral to the rehabilitation process and must be informed of the child's progress.

Technical Considerations

While designing the prototype it was important to consider our end user. We needed to ensure our solution was low voltage and low risk of shock as well as resistant to liquid. We also prioritized a design that was robust to wear and tear as we expected this device to become a regular feature within the lives of our end users. We also choose to design efficient non blocking code to ensure that our live data feed was correct and appropriately time stamped. Another technical consideration we had was how to mount a power source and not impede gait with the extra weight. Therefore, we chose to use a long USB cable attached to a power bank that could be either carried in the hand or placed in a pocket. This transferred the significant weight of the power bank off the foot which likely would have impacted gait especially after extended use. We chose to select a croc from a technical standpoint because it was low cost, is easy to take on and off, and it offered unique wire routing options. We also needed an instantaneous way to inform important stake holders about the child's progress which was done by the app but also a way to inform the child of their progress which is performed by the LED lights.

Clinical Considerations

From a clinical perspective we knew we needed to be able to gather data about the heel and toe strike pattern as well as measure the strength of the gastrocnemius contraction should the heel strike be unavailable. We also knew it was important to use motivational rehabilitation concepts in our design. This informed our choice of a croc, which is often considered to be a fun childish shoe. We need to ensure that we were challenging the child to improve, this was planned to be done by setting a variable threshold to trigger the various led lights. During one session, a child may have only slight heel contact, which would result in a green LED whereas in later sessions heel contact would need to be longer and more consistent to trigger the green LED light. We also wanted to reserve the use of the red LED light as much as possible as they may have a negative connotation and therefore, we would have focused on primarily showing the green and blue LEDs.

Design Selection and Initial Prototype

After deciding on our desired problem, we looked at all the presented solutions and decided to combine the solutions presented into the initial design in figure 2. This initial design consisted of the FSR's in the sole of the shoe as well as the LED indicators on the front. There was also the inclusion of a speaker module and potential for a deep learning pose model that we would have used to estimate foot positions.

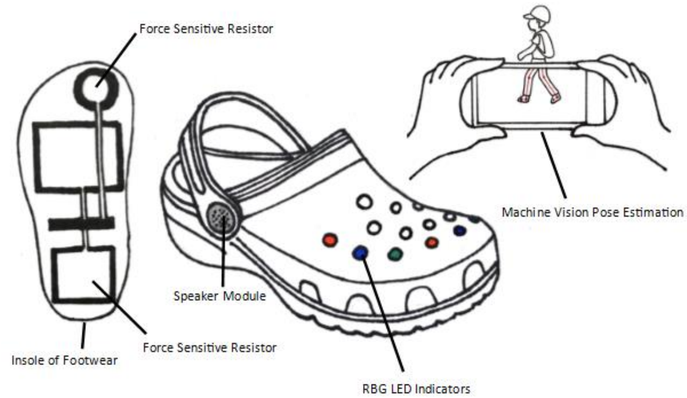

Figure 2: Initial Design

Design Iteration

We iterated on our initial design in several respects to approach our current design. Firstly, we chose to remove the deep learning pose model due to time constraints and complexity, but this is a feature we believe can still be implemented in the future. We also decided to create a mobile application to be able to instantly display the data for important stakeholders. The addition of the mobile application allowed us to remove the speaker module from the design as the speaker was now available due to the mobile device, however, we chose not to use auditory cues to ensure we did not confuse the child by potentially giving positive feedback audio when say a physiotherapist was pushing them to work harder.

We also chose to change the sensors we had incorporated. Originally, we had thought about using more than two FSRs but due to project constraints, we were limited to two FSRs. Additionally, we chose to include an EMG as one of our sensors that way we could still gauge gait improvement even if there was no heel contact such as in a severe equinus gait. We also ended up substituting all the RGB LEDs for a single LED and pompoms due to project constraints

We also added an additionally plastic waterproof cover to help prevent water damage.

These changes resulted in a simpler and more robust design that is better suited to account for a wide range of severity of equinus gait due to the addition of the EMG and is able to better relay information to the important stakeholders via the creation of the mobile application.

Analysis, Calibration, Validation

Before analyzing the data, we tuned the EMG gain to ensure that we were not saturating the sensor. Once that was done, we collected a variety of gait data, including normal gait, mild equinus gait where the heel still contacts the ground regularly, and severe equinus gait where the heel rarely or never touches the ground. This gait data was processed by applying a savgol filter to the heel and toe FSRs and the EMG to remove noise from the signal. We then took the mean of the toe and heel FSRs and thresholded the signals to create a signal that looks like a step function. When you add these two signals together you get a gait pattern. The details of this gait pattern will be explained later in this report.

For the EMG after applying a savgol filter we adaptively thresholder using the mean again and overlay the EMG pattern with the toe FSR pattern you can consistently see that the EMG leads the toe FSR slightly indicating that there is an identifiable correlation. We then normalize the signal between the 0 to 2 range to match the gait pattern. We would then take the area under the curve if there is no heel contact and compare that area under the curve value to get the relative improvement compared to previous steps based on if the area under the curve value is lower.

For our analysis plan the only calibration that needs to be performed is ensuring that the EMG gain is not too high. However, after that calibration done it is not likely to be needed to be performed again. However, it should be noted that we discard the first 2 seconds of values from the connection because sometimes upon connecting the values from the Bluetooth signal will be scrambled and meaningless and just lead to confusing results.

We validated our analysis and calibration results by using the real-world gait data that we gathered from our team members who would either have a normal gait or mimic various severities of equinus gait.

Results

Mechanical Design

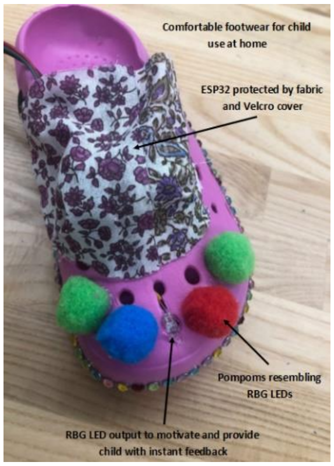

Figure 3: GastroCroc Top View

Here we can see the top view of our solution: the GastroCroc. We can see the fabric cover, which was later replaced with a plastic cover to add water resistance. We can see the single RGB LED that is used for instantaneous feedback for the child and the pompoms that act as stand ins for other LEDs.

Figure 4: GastroCroc Rear View

In the rear view of the GastroCroc you can see that there are two force sensitive resistors located at both the ball and the heel of the foot.

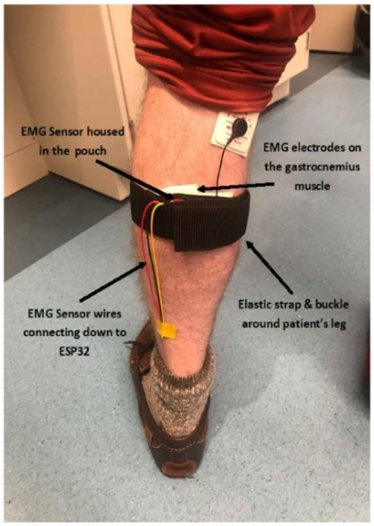

Figure 5: EMG Strap

The EMG is composed of a Velcro strap with a pocket sewn to hold the EMG. This pocket has holes that way pads can be easily attached and not have the strap interfere. The EMG electrodes are positioned on the gastrocnemius muscle which one of the primary muscles in plantarflexion of the foot.

Electrical Design

The electrical design is relatively simple with each sensor appropriately connected to the ESP32 microcontroller. The holes in the croc were used to help keep the wiring simple and clean. It should be noted, however, that the FSRs used can tear and damage the contacts so it is important to reinforce the solder connection with hot glue or tape to prevent tearing at the FSR contacts.

Software Design

The software design for this device is composed of two separate components. The first of which is the data collection and analysis. The data from the ESP32 was sent a 15-byte packet that is composed of a number separated by a pipe character. Since the largest number is 4096 the longest possible byte length for one element in the packet is 5. If the number is only 1-3 digits a 0 must be added to the end of the packet to ensure it remains the same length. This packet was composed as follows:

Toe_fsr	Heel_fsr	EMG	Variable length array of 0s

Figure 6: Bluetooth Packet Structure

The ESP32 also receives a message containing a character string of either “red”, “blue”, or “green” and converts that message to upper case to ensure that it is readable. In the future this string would be split into an RGB code to change the output of the RGB LED to whatever colored is desired but in the current implementation it only takes the 3 colour word strings.

Once the packet is composed it is sent to the Bluetooth buffer, which then needs a client to access it. This is done by our python program, 3P04_Data_Gather.py which connects to a given device by name and will record the data for an assigned period and place the values into a specified spreadsheet. This allows us to quickly gather information on the gait patterns for equinus gait and standard gait which are used to help analyze the data.

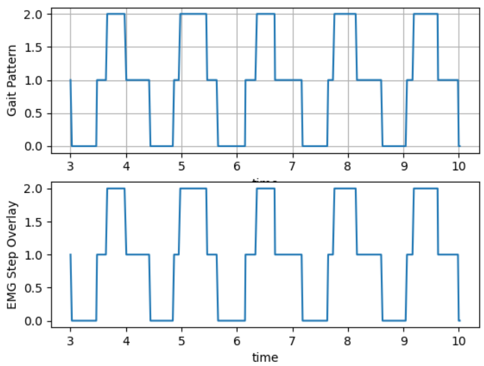

Figure 7: A typical gait pattern. There is no EMG overlay because each step can be analyzed without needing the EMG.

The data is analyzed in the 3P04_Data_Processor.py. Our FSR analysis follows our analysis plan. The gait pattern that is formed can be analyzed as follows: the pattern is 0 when there is no contact, then the first instance where it is 1 is where the toe contacts, it is 2 when both the toe and heel contact, and when it returns to 1 that is when only the heel is contacting. In a severe equinus pattern there may be no point where the value is 2 which indicates that the heel fails to touch.

The EMG analysis follows the plan outlined earlier in this report. While monitoring the gait pattern if we see that the gait pattern remains at 1 too long, we can assume that there is no heel contact occurring due to the nature of CP and muscle tightening. We then would then take the area under the curve for the period in which the gait pattern is 1 and that would constitute a step. This would then be compared to previous steps to see if there is a relatively improvement due to a reduction in the area under the curve number.

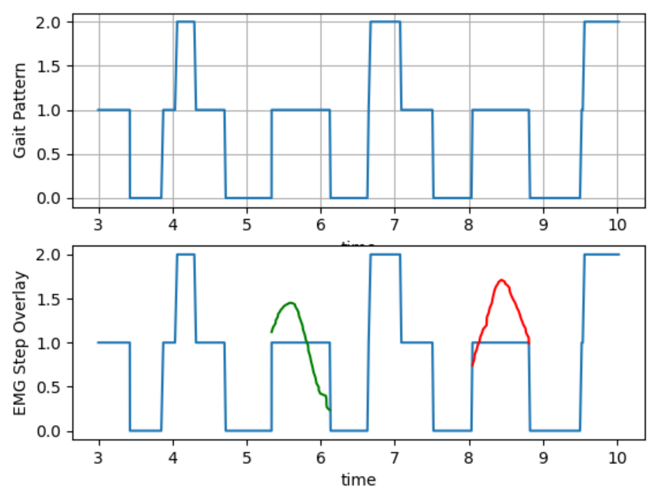

Figure 7: A severe equinus gait pattern. The lines on the bottom indicate when the EMG was used to determine which light color to show. The green line represents a step where the contraction of the gastrocnemius was not as severe as the red line as determined by the area under the cruve.

The second aspect of the software design was the mobile application for the important stakeholders. We successfully completed a mockup of the design using Adobe XD. This allowed us to export the artboards as React Native components, which we could then pair with a copy of the python code transcribed in JavaScript, allowing our app to work natively on mobile and be easily transferrable to a web application.

Figure 6: App Mockup Select Boards

Protocol

The protocol for this device is to first have the child place their foot in the shoe. Then you put the pads on the EMG strap and place the strap against the back of the calf (gastrocnemius) and then the ground pad off the side away from the EMG. Next, tighten the strap and plug in the device to the battery. Afterwards open the application and begin a new session. The phone would connect to the ESP32, beginning the training process. You would then have the child walk around as normally as possible to gain more information about their current gait state. You would then do the prescribed stretching / physical therapy routine. The reason for doing this routine after is to make sure that the reading is that of a typical day without loosening due to stretching.

Discussion

Future Changes and Limitations

For future revisions of our design from a mechanical view we would like to change from a soft waterproof cover to a hard-waterproof cover to better protect the electronics inside. We would also like to move to a system where the FSRs are integrated into the shoe and less susceptible to wear. In the future, we would like to use a custom PCB as well as more accurate and precise electrical components to improve our system durability and data accuracy. If given more time, we would have liked to complete the mobile application and been able to complete our entire project. It is just a matter of transferring the code over to JavaScript and linking together the artboards and adding the graphing React Native components. Furthermore, we have liked be able to communicate and process the data live with the ESP32 and use a full complement of LEDs.

We were limited by the sensors we had and therefore had to use relative measures of improvement as opposed to absolute measurements. This would have been ideal we could have compared progress accurately against other children with CP to gauge the child's progress in terms of their condition in relation to other cases. We also would have liked to have had more sensors to generate more accurate gait patterns. Instead of 2 FSRs, we would like to have used enough to create a smart insole.

Challenge Success and Design Feasibility

Overall, we were successfully able to address the design challenge and meet our design criteria. Our design was mobile, low cost, provided valuable data about the child's progress in a simple way to important stakeholders and appeared fun and motivating for the child to use. We believe that this design is very feasible and with certain future additions such as creating an insole instead of mounting sensors directly to the shoe, we could create a product that can be used in a variety of footwear, thereby enabling expansion to other potential customer markets. It should be noted that as we increase the accuracy and volume of sensors in the future, costs would also increase. If, however, we implement small scale manufacturing, parts could likely be sourced for cheaper.

Conclusions

Overall, this project was successful in creating a remote gait monitoring solution. This project shows how with the advent of powerful microcomputers and low-cost sensors there is an opportunity to create powerful new devices. And with the ever digitization of services, remote physical therapy solutions provide a key improvement in rehabilitation by providing more consistent information about a patient's progress whether old or young.

References

[1] P. Fv and X. Hgx, “Smart Insole: A Wearable System for Gait Analysis,” vol. 5, no. 18, pp. 0–3, 2012.

[2] H. Cho, “Design and Implementation of a Lightweight Smart Insole for Gait Analysis,” pp. 792-797, 2017.

[3] S. K. Tatla, K. Sauve, N. Virji-Babul, L. Holsti, C. Butler, and Hendrik F Machiel Van Der Loos, “Evidence for outcomes of motivational rehabilitation interventions for children and adolescents with cerebral palsy: an American Academy for Cerebral Palsy and Developmental Medicine systematic review,” Developmental Medicine & Child Neurology, vol. 55, no. 7, pp. 593–601, 2013.

[4] S. Armand, G. Decoulon, and A. Bonnefoy-Mazure, “Gait analysis in children with cerebral palsy,” EFORT Open Reviews, vol. 1, no. 12, pp. 448–460, 2016.

[5] “Electromyography (EMG),” *Electromyography (EMG) | Johns Hopkins Medicine*. [Online]. Available: https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electromyography-emg. [Accessed: 12-Apr-2020].

[6] T. Pin, S. Physiotherapist, P. O. Box, N. Melbourne, and P. E. Database, “Review The effectiveness of passive stretching in children with cerebral palsy,” vol. 48, no. 3, pp. 855–862, 2020.

Appendix A: Motivation Driven Rehabilitation Table

Table III: Evidence table – outcomes of motivational interventions for children and young people with CP (level of evidence I, II, or III)

Outcomes by ICF component of function and disability	Improved results with motivational intervention (statistically significant)	Improved results with motivational intervention (not statistically evaluated)	Worse results with motivational intervention (statistically significant)	Results unchanged or not statistically significant
Body functions				
Motivation				II+Weak[13]
Interest		III+Weak[16]		
Fun		III+Weak[16]		
Ankle DF ROM				III+Weak[16]
Ankle DF # Reps			III+Weak[16]	
Ankle DF Rep Completion Time	III+Weak[16]			
Ankle DF Hold time at End range				III+Weak[16]
Gait biofeedback	II+Weak[13]			II+W[13], II+Weak[14]
Activities & Participation				
Upper limb activities and function				II+Weak[15]

Figure 2: Measured outcome of motivation-driven interventions on children with CP [2]

Appendix B: Code

3P04_Data_Gather.py

```
# This code is designed to gather data from the esp32 and save the data as a
txt file

import bluetooth
import csv
import time

def get_address(target_name): # Find bluetooth device from a name
	while True:
		nearby_devices = bluetooth.discover_devices()
		for bdaddr in nearby_devices:
			if target_name == bluetooth.lookup_name(bdaddr):
				print("Device found")
				return bdaddr

if __name__ == "__main__":
	collection_time = 10
	header = ["time", "toe_fsr", "heel_fsr", "emg"]
	mac_address = get_address("OOF.png")
	# Create the client socket
	client_socket = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
	print("Connecting to device")
	client_socket.connect((mac_address, 1))
	print("Connected to device")

with open("kierra_equinas_severe_3_data.csv", "w+", newline="") as f: # CSV
to write to
	f.truncate(0) # Clear file
	writer = csv.writer(f, delimiter=",")
	writer.writerow(header)
	start_time = time.time()
	session_time = 0
	print("Beginning data collection")

	while session_time < collection_time:
		data = client_socket.recv(15) # Length of incoming bytes is 15
		data_array = data.decode().split("|") # Split along pipe chars
		del data_array[-1] # Delete the last element which is just a
variable length of 0's
		session_time = time.time() - start_time # Update session time
		data_array.insert(0, session_time) # insert the current time
		writer.writerow(data_array) # Write the row of data

print("Session complete")
client_socket.close()
```

3P04_Data_Processor.py

import csv

```
import csv
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter
from scipy.stats import tmean
from sklearn.metrics import auc


def get_data(name, trim_time=3):  # Get the data from the spreadsheet
    with open(name, "r", newline="") as f:
        reader = csv.reader(f, delimiter=",")
        time = []
        emg = []
        toe_fsr = []
        heel_fsr = []
        next(reader)

        for row in reader:
            cur_time = float(row[0])
            if cur_time >= trim_time:  # How many seconds to trim from start
of data
                time.append(cur_time)
                toe_fsr.append(int(row[1]))
                heel_fsr.append(int(row[2]))
                emg.append(int(row[3]))

        return time, toe_fsr, heel_fsr, emg


def area_under_curve(time_stamps, gait, emg, interval):
    start_index = 0
    auc_list = []
    for i in range(len(time_stamps)):

        if gait[i] == 1 and gait[i-1] != 1:  # If the gait is 1 and
previously it was not
            start_index = i # Start recording

        elif gait[i] == 0 and gait[i - 1] == 1: # When the gait is 0 after
previously being 1
            # Check to see if the duration when it was just 1 is longer than
the interval time
            if time_stamps[i] - time_stamps[start_index] > interval:

                # Slice lists based on the start index and ending index
                time_per_step = time_stamps[start_index:i]
                emg_per_step = emg[start_index:i]
                auc_val = auc(time_per_step, emg_per_step)
                auc_list.append([auc_val, time_per_step, emg_per_step])



    return auc_list
```

```
if __name__ == "__main__":
    time, toe_fsr, heel_fsr, emg =
get_data("kierra_equinas_severe_1_data.csv")

    # Apply Savgol filters
    toe_fsr = savgol_filter(toe_fsr, 53, 3)
    heel_fsr = savgol_filter(heel_fsr, 53, 3)
    emg = savgol_filter(emg, 61, 3)

    # Create threshold value
    heel_threshold = tmean(heel_fsr, (1, 4096))
    toe_threshold = tmean(toe_fsr, (1, 4096))

    # Threshold the data
    toe_square_sig = (toe_fsr > toe_threshold)
    heel_square_sig = (heel_fsr > heel_threshold)

    # Create the gait and emg
    gait_pattern = [sum(x) for x in zip(heel_square_sig, toe_square_sig)]
    emg = [x - min(emg) for x in emg]
    emg = [x / max(emg) * 2 for x in emg]

    # Plot Setup
    fig, axs = plt.subplots(2, 1)
    axs[0].plot(time, gait_pattern)
    axs[0].set_xlabel('time')
    axs[0].set_ylabel('Gait Pattern')
    axs[0].grid(True)

    auc_list = area_under_curve(time, gait_pattern, emg, 0.5)
    auc_vals = []
    auc_time = []
    auc_emg = []

    for item in auc_list:
        auc_vals.append(item[0])
        auc_time.append(item[1])
        auc_emg.append(item[2])

    axs[1].plot(time, gait_pattern)
    for i in range(len(auc_time)):
        # For each value where we needed to use the EMG plot the curve
        if auc_vals[i] != max(auc_vals): # Plot a green cruve if not the max
auc
            axs[1].plot(auc_time[i], auc_emg[i], color="green")
            print("GREEN")
        else: # Plot a red curve for highest auc
            axs[1].plot(auc_time[i], auc_emg[i], color="red")
            print("RED")

    axs[1].set_xlabel('time')
    axs[1].set_ylabel('EMG Step Overlay')

    plt.show()
```

3P04_Live_Processor.py (Unfinished)

```
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter
from scipy.stats import tmean
import bluetooth
import csv
import time
from sklearn.metrics import auc
import numpy as np


def bluetooth_connect(target_name):
        while True:
                nearby_devices = bluetooth.discover_devices() # Discover
nearby devices
                mac_address = None
                for mac_address in nearby_devices:
                        if target_name == bluetooth.lookup_name(mac_address): #
If we find desired device name
                                print("Device found")

                                # Create the client socket
                                client_socket =
bluetooth.BluetoothSocket(bluetooth.RFCOMM)
                                print("Connecting to device")
                                client_socket.connect((mac_address, 1))
                                print("Connected to device")
                                return client_socket


if __name__ == "__main__":
        collection_time = 10 # How many seconds to collect data for
        header = ["time", "toe_fsr", "heel_fsr", "emg"] # Spreadsheet headers
        connection = bluetooth_connect("OOF.png")

        with open("test.csv", "w+", newline="") as f:

                # CSV Creation
                f.truncate(0) # Clear file
                writer = csv.writer(f, delimiter=",")
                writer.writerow(header)

                # Setup
                max_length = 100
                start_time = time.time()
                toe_fsr = np.zeros(max_length)
                heel_fsr = np.zeros(max_length)

                # Plot Setup
                session_time = 0
                heel_touch_time = 0
                timestamps = np.zeros(max_length)
                emg = np.zeros(max_length)
                fig, axs = plt.subplots(2, 1)
                axs[0].set_xlabel('Time (s)')
                axs[0].set_ylabel('Gait Pattern')
```

```
        axs[1].set_xlabel('Time (s)')
        axs[1].set_ylabel('Toe Pattern')

        print("Beginning data collection")
        while session_time < collection_time:

                # Get data
                data = connection.recv(15)
                data_array = data.decode().split("|")
                del data_array[-1]

                # Get time
                current_time = time.time()

                # Write CSV
                session_time = current_time - start_time
                data_array.insert(0, session_time)
                writer.writerow(data_array)

                timestamps = np.append(timestamps, [session_time])[-
max_length:]
                toe_fsr = np.append(toe_fsr, [int(data_array[1])])[-
max_length:]
                heel_fsr = np.append(heel_fsr, [int(data_array[2])])[-
max_length:]
                emg = np.append(emg, [int(data_array[3])])[-
max_length:]

                # Apply savgol filtering
                toe_fsr = savgol_filter(toe_fsr, 53, 3)
                heel_fsr = savgol_filter(heel_fsr, 53, 3)
                emg = savgol_filter(emg, 61, 3)

                try:
                        heel_threshold = tmean(heel_fsr, (1, 4096))
                        toe_threshold = tmean(toe_fsr, (1, 4096))
                        emg_threshold = tmean(emg, (1, 4096))
                except ValueError: # Error due to insufficient data
length
                        heel_threshold = tmean(heel_fsr, (0, 4096))
                        toe_threshold = tmean(toe_fsr, (0, 4096))
                        emg_threshold = tmean(emg, (0, 4096))

                # Threshold
                toe_square_sig = (toe_fsr > toe_threshold)
                heel_square_sig = (heel_fsr > heel_threshold)
                emg_square_sig = (emg > emg_threshold)

                gait_pattern = [sum(x) for x in zip(heel_square_sig,
toe_square_sig)]
                toe_pattern = [sum(x) for x in zip(emg_square_sig,
toe_square_sig)]

                # Send LED feedback
                if gait_pattern[-1] == 2:
                        state = "GREEN\n"
                        heel_touch_time = current_time
```

```
                elif gait_pattern[-1] == 1:
                    if current_time - heel_touch_time > 10:
                        if auc(time[-20:], emg_square_sig[-
20:]) < auc(time[-100:], emg_square_sig[-100:])/5:
                            state = "GREEN\n"
                        elif auc(time[-20:], emg_square_sig[-
20:]) < auc(time[-100:], emg_square_sig[-100:])/4.5:
                            state = "BLUE\n"
                        else:
                            state = "RED\n"

                    elif current_time - heel_touch_time > 2:
                        state = "RED\n"
                    else:
                        state = "BLUE\n"
                else:
                    state = "NONE\n"

                connection.send(state)
                print(state)

                # Show the plots
                axs[0].clear()
                axs[1].clear()
                axs[0].plot(timestamps, gait_pattern)
                axs[1].plot(timestamps, toe_pattern)

                plt.draw()
                plt.pause(0.001)

    print("Session complete")
    connection.close()
```

3P04_ESP32.ino

```
#include "BluetoothSerial.h"

BluetoothSerial ESP_BT;

int RED_PIN = 19;
int GREEN_PIN = 22;
int BLUE_PIN = 21;
int TOE_PIN = 35;
int HEEL_PIN = 34;
int EMG_PIN = 37;

String message;
char incoming;

const int array_len = 3;
int output_array[array_len];

void setup() {
  Serial.begin(9600);
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  pinMode(HEEL_PIN, INPUT);
  pinMode(TOE_PIN, INPUT);

  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
  ESP_BT.begin("OOF.png");  // This line is what breaks the force resistors

}

void display_green() {
  // Write only the green pin high
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, LOW);
}

void display_red() {
  // Write only the red pin high
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
}

void display_blue() {
  // Write only the red pin high
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, HIGH);
}
```

```
void display_none() {
  // Write only the red pin high
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
}

String create_packet(int sensor_readings[array_len]) {

  String outgoing_packet = "";

  for (int i = 0; i < array_len; i++) {
    outgoing_packet += sensor_readings[i];
    outgoing_packet += "|";
  }

  for (int i = outgoing_packet.length(); i < (array_len)*5; i++) {
    outgoing_packet += 0;
  }
  return outgoing_packet;
}

void loop() {
  int toe_force = analogRead(TOE_PIN);
  int heel_force = analogRead(HEEL_PIN);
  int emg = analogRead(EMG_PIN);

  output_array[0] = toe_force;
  output_array[1] = heel_force;
  output_array[2] = emg;

  String outgoing_message = create_packet(output_array);

  Serial.println(outgoing_message);
  ESP_BT.print(outgoing_message);

  if (ESP_BT.available() > 0) //Check if we receive anything from Bluetooth
  {
    incoming = ESP_BT.read();

    if (incoming == '\n') {
      message.toUpperCase();
      message.trim();


      if (message == "GREEN") {
        display_green();
      }
      else if (message == "RED") {
        display_red();
      }

      else if (message == "BLUE") {
        display_blue();
      }

      else if (message == "NONE"){
```

```
        display_none();
      }

      Serial.println(message);
      message = "";

    }
    else {
      message += incoming;
    }
  }
}
```

Appendix C: Team Member Roles, Responsibilities, and Personal Contributions

	Roles
Lianna	Administrative tasks (booking rooms, submitting Milestones, etc.) Mechanical Prototype Design
Chelsea	Principle researcher for • Methodology • Anatomy • Social and behavioural application Consulting on all design decisions regarding child psychology and applicability to the end user
Kierra	Principle support role helping with time consuming tasks Principle editor for milestones
Jeff	Software Design • ESP32 Code • Analysis Plan and Design Decisions • Mobile Application Electrical Design and Soldering

	Personally, I found that overseeing all of the software and electrical design meant that I did not contribute on the milestones as much as I probably could have. However, it was a matter of balancing as I was the only one on my team who felt comfortable enough writing the majority of the code and my team was happy to let me focus on the software and electrical design while they focused on the milestones and the research aspects of the project.

Appendix D: Sketches

Initial Design Sketch