Microcontroller Communication

A Senior Project Report

presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Engineering

By Sereen Benchohra

June 2022

Abstract

The Senior Project is focused on Microcontroller Communication in particular the communication between the Raspberry Pi and the Arduino. The microcontroller communication is used as part of a bigger Project to make a Lower-Limb Exoskeleton Assist Project, or LLEAP. The LLEAP uses two microcontrollers in order for the system to work, the Arduino in which it handles the Sensor Data from the User and User Interface , it processes and translates the data into motor commands. The Arduino then synchronizes, preprocesses, and packages and sends them to the Raspberry Pi, which reads in the data and computes and trajectory through ROS. Once the trajectory has been completed , the Raspberry Pi then sends the trajectory data to the Arduino, where the Arduino processes and translates for motor movement.

Table of Contents

Introduction

Client

There are two main clients for this project, the direct client, the club , the Lower-Limb Exoskeleton Assist Project, or LLEAP, who requested the microcontroller communication bridge , to help function the Exoskeleton which is needed for the client of the exoskeleton, Tamar De Leon (The Second Client), and the ultimate client that would benefit from this project. Tamar is a local student at Cuesta College in San Luis Obispo studying business. He was born with quadriplegic cerebral palsy which prevents him from using his legs. The goal is eventually for the exoskeleton to be working for the ultimate client.

Stakeholders

The primary stakeholders that will be affected and have vital interest in the project's design decision , is the LLEAP club , since the microcontroller communication is vital in having the overall exoskeleton working and integrating all the components together. As a result, it is important that throughout the process , project and design designs were based off of current design implementations of the LLEAP project, for example, since the main read and write functions in the ROS in the Raspberry PI (where it computes the trajectory) is in C++, the serial communication had to be in C++ , likewise other design aspects like the type of microcontrollers used had to be considered since it had to be in conjunction with the hardware used in exoskeleton (the Raspberry Pi and Arduino).

Framed Insights and Opportunities

Since the primary client and stakeholder is the LLEAP club. As a result there isn’t a difference in needs between the stakeholders and clients . The primary discussion between LLEAP and I was on how to get the Microcontrollers to communicate and how to integrate it into the overall LLEAP design.

Project Goals and Objectives

The goal for this quarter is to find the optimal Interprocess Communication between the Microcontrollers, find which data format and protocols, and measure and account for the timing to process for the motor to process the command. In addition, familiarize yourself with ROS in order to figure out how to integrate the read and write commands. Once completed, the goal is then to implement the read and write commands and effectively connect the two microcontrollers and have the Arduino and Raspberry Pi send data back and forth.

Project Deliverables

The deliverables that will be produced in this project is the message protocol format for sending and receiving data. The type of protocol itself used in the communication, in which the Universal Synchronous/Asynchronous Receiver/Transmitter. In addition, other deliverables is the schematic setup, the code in both the Raspberry Pi and the Arduino that sets up the communication between the two microcontrollers.

Project Outcomes

If the project were to exist, it would allow more complex systems that have multiple different functionalities that would send vital data to each other and create more smooth overall functionality and speeds. In addition, it would allow for more Robotic opportunities , and Autonomous Systems.

Background

Thanks to the common availability of Arduinos and Raspberry Pi's , there is an abundance of literature when it comes to how to deal with Raspberry Pis and Arduinos, their limitations and what they can do. In addition, since Microcontrollers are a key component in integrating the exoskeleton and communicating the key data , it is important that there are no major errors with the system when communicating bad data since it could prevent the user's mobility and cause an accident. If the user is in an area like an intersection and the microcontrollers send old data like move forward, the consequences can be severe. In addition, there is a whole literature on the ethics of creating prosthetic limbs. Many ethical issues regarding prosthetics usually are about how the functioning of a prosthesis for the remainder of someone's life cannot be predicted reliably on the basis of a couple of clinical trials with human subjects or a few tests with animals. There is a real risk, therefore, that people will be fitted with prostheses or implants that malfunction, have harmful side-effects. Ideally, prostheses would be tested over many years, decades even, and involve a large number of human subjects. But such extensive clinical trials and experimental uses are often considered too lengthy and costly and raise ethical issues by making guinea pigs out of human beings. Tests on animals often cannot serve as a substitute, and also raise ethical issues of their own[1][2][3]. In addition, how each. In addition, potential costs, and disparities on who can afford the prosthetics are another area that is a problem.

Formal Project Definition

Customer Requirements

The customer requirements for the project were :

- Find a communication protocol that is optimal for communicate between the two microcontrollers
- Once a communication format was found, create protocol format for sending data
- In addition have the microcontrollers communicate with each other
- Finally research on how to integrate the communication into the larger LLEAP project

Engineering Requirements

Spec Number	Parameter Description	Requirements or Target with Units	Tolerance	Risk	Compliance
1	Power	3.3 Volts	Max	L	S/A
2	Board Connection	Wired	Max	L	T
3	Raspberry Pi connection	Wireless	Low	H	I
4	Coding Language	C++	Max	M	T
5	Protocol	USART	Low	M	T
6	Design Protocol Format	8-bit packets data format	Max	M	A

7	Integration into larger LLEAP	Part of ROS	Max	H	I

Table 1: Engineering Requirements Table

The engineering decisions were made based on a combination of   customer requirements and what was inferred from the customer requirements. In Table X, is where the Engineering Requirements for the Microcontroller Communication should be doing. Further discussion on how much of the requirements was actually implemented will be discussed in the System Testing and Analysis.

Customer

Since this project was created specifically for the LLEAP club, the Customer would be LLEAP and they would specifically use the Microcontroller Communication Bridge. The Customer value is to aid people improve their lives and mobility. They specifically want to help their own client, Tamar, be able to walk and improve his quality of life.

Use Cases and/or User Stories

Since the project is created for the LLEAP club, its use case would be utilized to communicate sensor data, trajectory data between the software architecture utilizing

ROS , and sensor data from the Arduino (Figure 1).

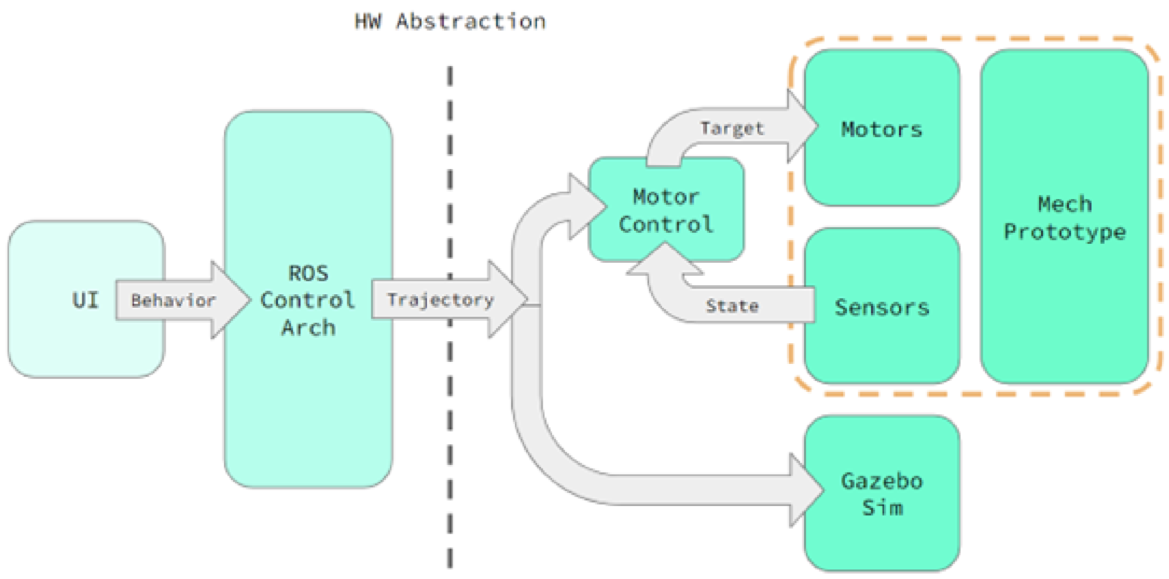


Figure 1: General LEAP architecture

Design

The design decisions were based directly on the Customer and Engineering Requirements. The project’s goal was to transmit the ROS trajectory to the Arduino (Figure 1) , and have the Arduino process and translate the data into motor commands , once finished , the Arduino checks for sensing data to send to the Raspberry Pi (Figure 2) .

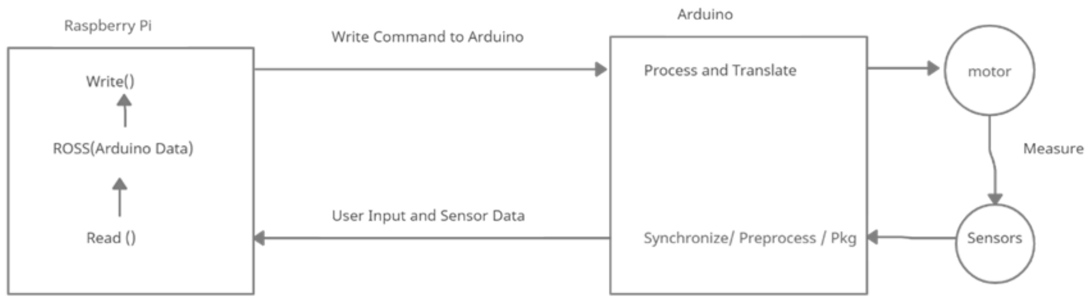


Figure 2: Communication between the Raspberry Pi and Arduino

Most of the design was based on the Engineering Requirements The first specification of the power was inferred based on the power requirements was based on the given microcontrollers used for the exoskeleton, the Arduino and Raspberry Pi. Since the power in the Raspberry Pi cannot exceed 3.3 V , the specification and connection between the the Arduino and Raspberry cannot exceed 3.3 V , which is fairly a low risk since the decision to use a USB connection to connect the two microcontrollers , handles the power conversion between them.

The second specification was based on the overall design of the LLEAP's exoskeleton design in which the wiring is in a closed system , so as a result of how the system was already designed, having the two microcontrollers connected wirely seemed like a logical conclusion. In addition, it seemed to be the most secure way, in order to prevent outside interference with data sending(see Figure 3.

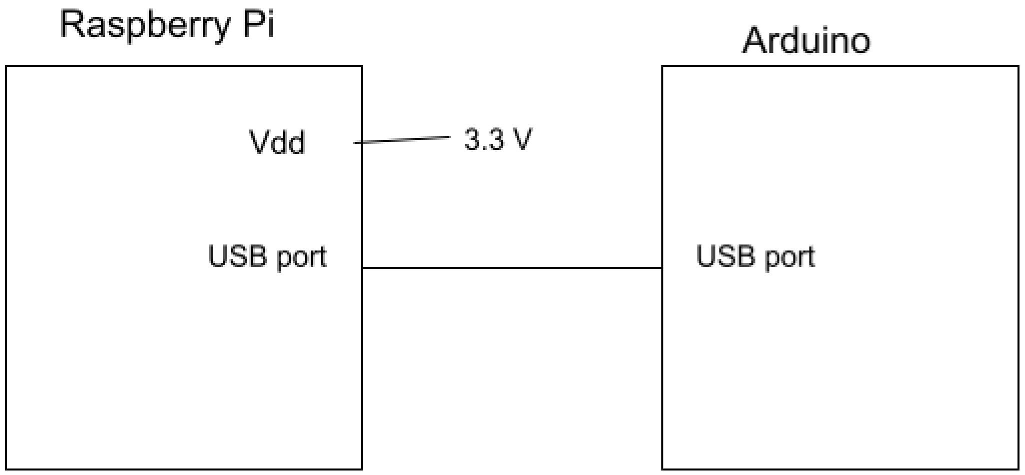


Figure 3: Schematic of the Microcontroller Bridge

The third specification was one of the hardest to reach, ideally since the use of the Raspberry Pi is headless (means no monitor and keyboard has to be access via terminal), it needs an internet connection to access it, and after two weeks the decision was to change it to an Ethernet connection, given the time constraint and move onto working on the actual communication between the Raspberry Pi and Arduino communication.

The fourth specification was created as a result of precedent for the seventh specification, where the communication protocols were to eventually have to be written in C++, in order to integrate into ROS .

The fifth specification was finding the optimal protocol that would suit the Customer's needs. The USART protocol seemed like the optimal choice given it can work Synchronously and Asynchronously. For the exoskeleton , new data should be sent when the sensors are being used and hence it is not based on a clock, but rather if there is new data. In addition, the USART protocol is limited only to two microcontrollers which is optimal for this system since only two microcontrollers will ever be needed, and prevent any additional components being added , making the system safer and preventing any additional hardware trojans. Finally, the USART is supported by the USB, so wiring is more direct.

The sixth specification was based on how data packets should be sent. The data design was based on the communication protocol that was used , the USART. The intended goal of the design document (see Appendices for USART data format) , according to the Customer , was to be tentative and a work in progress since much of the exoskeleton has not been implemented, the design protocol format used did not have much direct application.

The seventh and final specification from the Engineering Requirement that was attempted was integrating the reading and writing protocols into the ROS module and software architecture part of the greater LLEAP project.

System Testing and Analysis

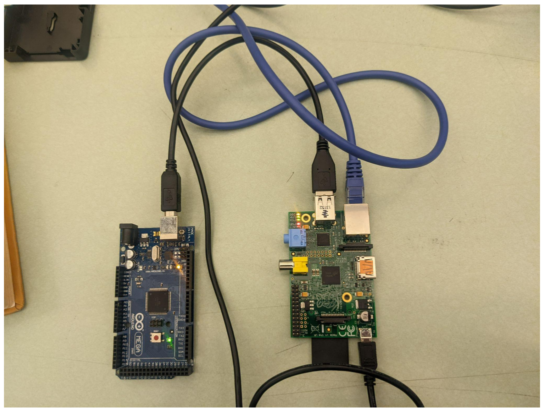
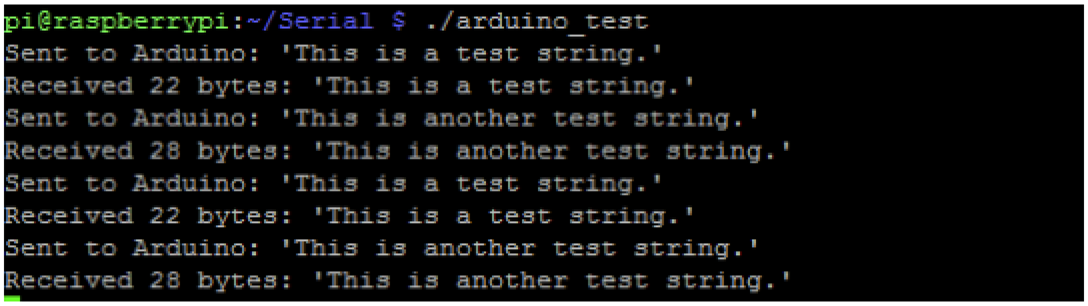


Figure 4: Microcontroller Communication Physical Setup

When testing out the communication between the Arduino and Raspberry Pi , the code was written in Python to first test that the data is able to be sent between the two microcontrollers. However in order to actually integrate into the ROS, the communication protocol had to be written in C++ (see Figure 5), so a considerable amount of time was spent changing the communication between the microcontrollers into RS232 utilizing C++ (Figure 6 and 7).



```
pi@raspberrypi:~/Serial $ ./arduino_test
Sent to Arduino: 'This is a test string.'
Received 22 bytes: 'This is a test string.'
Sent to Arduino: 'This is another test string.'
Received 28 bytes: 'This is another test string.'
Sent to Arduino: 'This is a test string.'
Received 22 bytes: 'This is a test string.'
Sent to Arduino: 'This is another test string.'
Received 28 bytes: 'This is another test string.'
```

Figure 5: Implementation of the communication between the Raspberry Pi and Arduino

The seventh specification was high risk , since of the many issues encountered, much of the LLEAP project was not fully functional or was in the planning phase. As a result , it was difficult to integrate and test with real time data. However, with research , integrating the protocol is highly possible, into the ROS system. But due to the fact the software architecture is in the planning phase, the best route is to relay and delegate to the team specializing in ROS on how to integrate the communication protocols into the ROS read and write commands.

Conclusion and Future Work

All in all, the project was able to successfully implement communication between the Raspberry Pi and Arduino. Code in the Raspberry Pi was written in C++ , which would allow actual integration into the exoskeleton project (and other potential exoskeleton projects) in the near future when the other components of the exoskeleton have been implemented, and gives the club a strong foundation to utilize. In addition, finding the best communication protocol for the exoskeleton and the tentative data format for future reference or at least a starting point on how to format and parse data communication within the exoskeleton.

Reflection

There are a couple of lessons that I learned from this Senior Project. The lessons that I learned from this Senior Project is that not everything works out as intended , and as a result you have to either make do with the limited constraints that you have and make alternatives and workarounds to get my part of the project to work since it is part of a bigger project and the rest of the project is still in planning phases. As an engineer , I learned how to deal with setbacks and with non-ideal circumstances.

Bibliography

[1]Brey, P. (2000). 'Technology as Extension of Human Faculties,' In C. Mitcham, ed., Metaphysics, Epistemology, and Technology. Research in Philosophy and Technology, vol 19. London: Elsevier/JAI Press.

[2]Gray, C. H. (ed.) (1995). The Cyborg Handbook. New York and London: Routledge.

[3]Moor, J. (2004). “Should We Become Cyborgs?” In R. Cavalier, ed., The Impact of the Internet on Our Moral Lives. Albany, NY: State University of New York Press.

Appendices

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "rs232.h"

#define BUF_SIZE 128

int main()
{
  int i=0;
  int cport_nr=24; /* /dev/ttyACM0 */
  int bdrate=57600; /* 9600 baud */

  char mode[]={'8','N','1',0}; // 8 data bits, no parity, 1 stop bit
  char str_send[2][BUF_SIZE]; // send data buffer
  unsigned char str_recv[BUF_SIZE]; // recv data buffer
  strcpy(str_send[0], "This is a test string.");
  strcpy(str_send[1], "This is another test string.");

  if(RS232_OpenComport(cport_nr, bdrate, mode))
  {
    printf("Can not open comport\n");
    return(0);
  }

  usleep(2000000);  /* waits 2000ms for stable condition */
  while(1)
  {
    RS232_cputs(cport_nr, str_send[i]); // sends string on serial
        printf("Sent to Arduino: '%s'\n", str_send[i]);
        usleep(1000000);  /* waits for reply 1000ms */
        int n = RS232_PollComport(cport_nr, str_recv, (int)BUF_SIZE);
        if(n > 0){
      str_recv[n] = 0;   /* always put a "null" at the end of a string! */
      printf("Received %i bytes: '%s'\n", n, (char *)str_recv);
       }
       i++;
    i %= 2;
    usleep(5000000);  /* sleep for 5 Second */

  }
  return(0);
}
```

Figure 6: C++ code implementation of Message Protocol in Raspberry Pi for sending and receiving message from Arduino

```
void setup() {
  Serial.begin(57600); // opens serial port, sets data rate to 57600 baud
}

void loop() {
  while (Serial.available() > 0) { // if any data available
    char incomingByte = Serial.read(); // read byte
    Serial.write(incomingByte); // send it back
  }
}
```

Figure 7: Code implementation in Arduino for receiving and sending messages to Raspberry Pi

USART protocol Data format design for sending and receiving data

Reads the general info layout and type of the data and then actually reads the byte of data

1) Info Category
   (Arduino -> RPI)

Read from Arduino

BITS 2:0 **LGM**: Leg General Movement

reserved	reserved	reserved	reserved	reserved	LGM [2:0]		
					rw	rw	rw
7	6	5	4	3	2	1	0

BITS 2:0 **LGM**: Leg General Movement

These bits configure the general movement for the exoskeleton

000: STEP_RIGHT
001: STEP_LEFT
010: SIT
011: STAND
100: STOP
101: Saved for future potential movement types
110: Saved for future potential movement types
111: Saved for future potential movement types

BIT 7:6 **DS**: Data State

00: Info category
01: Sensor Data
10: Motor Command
11: - Future space

BIT 5: **RF** : Ready Flag

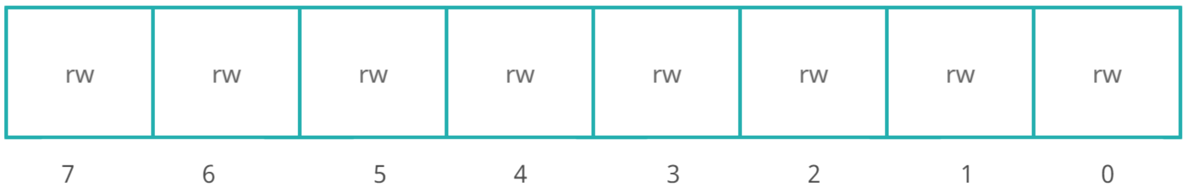
This bit indicates that the Arduino has new sensor data to be read, the data is only read if the RF bit is 1. Once the data is read, the RF is cleared, the RF flag is set only when new sensor data is inputted.

0: Ready Flag is off (no new sensor data)
1: Ready flag is set (new sensor data is inputted

BIT 4: SRDF: Start reading Data Ready Flag

This indicates to the start to read data
0: Data not ready to start
1: Data start ready to read

2) Sensor Data



READ end - From RPI

Reads in address, (if I2C) -> checks if it is LEFT or RIGHT foot (one bit) -> have 6 motion commands(have a 3 -bit register, might block some bits or only care for certain bits)

1) Ideally Read in a Packet (8 bit) from Arduino.

2)

// there are 3 bits left to use for any general motion command (Bits 5-7)

Reads in sensor Data from Arduino

2)

— — — — — — — — ——

```
#define L/R BIT 7
#define LEFT 0b10000000 (0x80) // bit 7 turned on
#define RIGHT 0b00000000 // bit 7 turned off
```

1) Info category (Ard->RPI, 4- 6 bits)
    a) High level cmd
        - step left
        - Step right
        - Sit
        - Stand
        - … stop?

2) Sensor Data (ard->RPI)
    a) What sensor?

- i) Encoders:
  - (1) Hip
  - (2) Knee
- ii) Force
  - (1) Foot
  - (2) Thigh
  - (3) hamstring
  - (4) Shin
  - (5) calf
- iii) IMUs(?)
  - (1) Upper leg
  - (2) Lower leg
  - (3) Torso

b) What info from each ?
- (1) Angle?
- (2) Tics?
- (3) Speed?
- (4) Accel?
- (5) force(v)?
- (6) What level of precision ? how many bits ?

3) Motor Cmd (RPi -> Ard)
   a)

1) Ask what the code is on the high level (see picture ref)
2)

3) Motor Command RPI->Arduino
   Writes to Arduino

Have a Write Command

3) Motor Command (Raspberry Pi -> Arduino)

rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0

Bit 7:6 Data Type

Bit 5: Left or Right Leg

0: Left
1 : Right Leg

Bit 4:3 Joints

Bit 2:1 : Not used for now (future use)

Bit 0 : Start Bit

One this packet is sent then motor command data is sent until stop bit

Senior Project Analysis

Summary of Functional Requirements

What the Senior Project does is that it communicates between the two microcontrollers, more specifically the Arduino and the Raspberry Pi. The Raspberry Pi sets up a UART communication protocol that was primarily used through the Universal Serial Bus Port (USB) that connects both the Arduino and Raspberry Pi directly. The Raspberry Pi sends data, the Arduino acknowledges the data sent by the Raspberry Pi , and sends its own message to the Raspberry Pi, where the Raspberry Pi reads the data from the Arduino and prints it to the terminal. The program runs in an infinite loop until it is killed by the user.

Primary Constraints

There were many challenges and difficulties and a considerable number of constraints associated with this Senior Project. One of the major constraints of the senior project was that the project was part of a bigger project for a club, called Lower-Limb Exoskeleton Assist Project (LLEAP). Since the overall LLEAP project is not fully functional, and multiple parts of the LLEAP are highly integrated with each other , the microcontrollers were not able to utilize sensor information planned from the Arduino since the sensing Team was still in the early stages of the project. Likewise, the software architecture where ROS is implemented, retrieves the sensing data and computes the trajectory and sends the trajectory into the Arduino , so that the Arduino would drive the motors to move the legs.

The crucial sensory data from the Arduino , and Trajectory data from the ROS were not available since both parts of the projects are in the early stages , hence as a result data testing was merely theoretical.

One challenge faced in implementing the microcontroller communication is setting up the internet connection needed to access the Raspberry Pi , since the goal was to use the Raspberry Pi without using the Monitor and Keyboard, hence an internet connection is necessary to access the Raspberry Pi. Initially a wireless connection was attempted to set up the Raspberry Pi to access the terminal , but given the time constraints and productivity, the internet connection was changed to an ethernet connection.

Economic

Since the project is only dealing with microcontroller communication , the actual set is fairly small . The only real costs are the Raspberry Pi, the ethernet cable , and the Arduino. Which is shown in table 1.

The actual cost of the project totalled to $0 due to the equipment already owned by the club.

	Cost
Arduino mega 2560	$41.24
Ethernet Cable	$5
Raspberry Pi	~$34 - 64
Total	$80.24 - $110.24

Table 2 . Bill of Materials

The planned development time for the project was calculated to be about 12-13 weeks however due to setbacks the actual development time it took to complete the project took roughly 15 weeks approximately. The first 7 weeks were in the planning and research stages of the project and the remaining 8 weeks were the implementation of the microcontroller communication bridge.

If manufactured on a commercial basis:

If the microcontroller communication bridge were to be manufactured on a commercial basis the primary customers would be robotics companies that need multiple communication between microcontrollers . The estimated manufacturing cost would be $150 and the purchase price would be $200 (a $50 profit for each unit)

Environmental

Describe any environmental impact associated with manufacturing or use.

The potential environmental impact of this device would be since it utilizes microcontrollers , it has to use limited resources like water a lot to make and cool and the metals used in the microcontroller have to be mined (which causes pollution)

Sustainability

Since it is an Embedded System and requires the use of Microcontrollers , it requires a good amount of water to create the chip. In addition, it needs to run whenever the exoskeleton is powered on , so it requires a significant amount of energy.

Ethical , Health and Safety

Since Microcontrollers is a key component in integrating the exoskeleton and communicating the key data , it is important that there are no major errors with the system when communicating bad data since it could prevent the user's mobility and cause an accident. If the user is in an area like an intersection and the microcontrollers send old data like move forward, the consequences can be severe.

Social and Political

Like many other electronics , the social and political concerns are tied to the manufacturing of the microcontrollers. The chips are typically made from Taiwan , a major powerhouse for semiconductor chips and pretty much the main global supplier. Hence whenever there is a drought in Taiwan , the supply chain comes to a halt, or whenever there is tension with China , that also affects the supply chain.

Development

The techniques that I have learned in this project are how to utilize Arduino and Raspberry Pi , since this is the first time that I have used these devices. In addition , I learned more in depth of serial communication , and how to utilize communication between the two Microcontrollers in both python and C++. Likewise , some soft skills I learned were how to have alternative plans and how to deal with setbacks if the intended plan did not work out.