R.A.W. Remote Arm Wrestling

Tyler Costa, EE, Steven Lucey, CSE, Zachary Matthews, CSE, and Steven McGrath, EE

*Abstract*—We introduce R.A.W. (Remote Arm Wrestling), a real-time system that emulates arm wrestling across a low latency network. This system provides each user with the same fun and excitement of an in-person arm wrestling match, except the game will be played over a distance between the two users. The robotic arms are powered by a motor, which drives a series of gears. The forearm of the arm will have strain gauges on it to find the force applied by the user. The opposite arm will replicate the force being applied by that user. There is also a phone application that allows a user to make a profile and join/schedule matches.

# INTRODUCTION

T

HE problem that we are developing a solution to, is the stress that is put on long distance friendships. People try to bridge this gap is through phone calls or video chats. These methods are not always that engaging, which leaves people unsatisfied with their experiences. They get frustrated by the distance and no longer keep in touch with that person. The best way we found to overcome this problem, is by making a more engaging experience through competition. We are developing a remote arm wrestling system. In this, people will be able to coordinate matches with friends of far distances and keep in touch. This game might also be able to incentivize people to work out in order to get stronger to beat their friend.

According to Stephen J. Betchen of Psychology Today, there are 4 big problems with long distance relationships that causes them to ultimately fail.[1] The biggest problem that he mentions, is the time and money that goes into maintaining the relationship. He explains that people are always complaining about having to drive and hate spending all that time in their car. He says that even being able to see your friend at the end of it, “chances are pretty good that it will eventually wear on you” [1]. He also highlights that the familiarity between you and that friend will begin to fade as, “Seeing someone on weekends or once a month just won’t cut it” [1]. This is where our game comes into play. It makes it so it feels like you are seeing that person more than just once a month, as you can have physical interaction with them.

Some ways in which people have attempted to bridge this gap, is by scheduling events to do together while on their phone. For example, they might try and watch movies or TV

shows together while on the phone. This is so they can feel like they are actually with each other. People have also further expanded this into using video calls. This makes it so that they can actually see their friend on the other end. Others might try

T.Costa from Kingston, MA (email – tdcosta@umass.edu)

S.Lucey from Easton, MA (email – sjlucey@umass.edu)

Z.Matthews from Peabody, MA (email – zmatthews@umass.edu)

S.McGrath from Stoneham, MA (email – stevenmcgrat@umass.edu)

to engage with each other by playing video games online together. This is more interactive than a phone call alone. As it allows the friends to work together to achieve a common goal. In a recent study by GeekWire they found that 1.2 billion people are playing video games worldwide; 700 million are online. [6]

As technology keeps growing and expanding in capabilities over time people are finding more and more ways to

connect.  With that being said, not one of these include a physical aspect, as engaging as ours. Our solution allows for you to physically engage with your friend so it does not feel like there is as much distance between you.

This problem is of high concern as people depend greatly on the relationships that they have created. The fear of loneliness is prevalent in our society today. People like to be constantly involved in some type of activity, that allows them to stay engaged with others. It has been found in studies that people feel alone the older they get in life. With the creation of this remote arm wrestling system it will be easy to schedule in a quick match with a friend, so that you can still maintain your relationships no matter how busy you become.

A picture containing screenshot

Description generated with very high confidence

Figure 1. Drawing of arm with connections

The system we have decided to make needs to fall within the requirements we have decided to implement. The system will need to have two robotic arms connected over a low latency network. This is so that friends could arm wrestle each other almost like real time. The robotic arms will need to be built sturdy enough to be able to handle the torque being put onto it by both users. The forearm of the arms must have a length of about 0.3 meters; this being the average forearm length of a person. This system will also must to have an application to go along with it. This application will be in charge of allowing the user to make a profile, schedule a match and/or join a match. This application will also allow you to send friend request to your friends, so that you can easily schedule a match with them. Lastly, the application allows you to reserve the match on a nearby arm, which will be located at a fixed geographical location. These arms will most likely be placed at bars or arcades; locations that already stimulate engaging behavior. This system will operate on 11.5 VAC power. This will make it so that battery life is not a worry when using the system. The table below summarizes the system specifications.

TABLE I

Requirements Specifications

|  |  |
| --- | --- |
| Requirement | Specification |
| Forearm length must be the average length of a human foreman | The forearm length must be 0.3m |
| Provide real-time feedback | Feedback will be less than 100 ms |
| Arm must operate like a human arm | Gears and Motor must not have an sudden movements |
| Arm must be able to withstand force applied | Stable and Secure to forces up to a PWM of 255 |
| All part must be enclosed | Clear encasement made so that no gears are exposed |
|  |  |
|  |  |

# Design

## Overview

In order to create and fully simulate the functionality of an arm wrestling match we plan to recreate the force of each user and replicate it as physical torque on the opposite arm. In figure 2 you can see our overall design with each individual subsystem needed as well as the information being transferred inside of the system itself. In order to accurately measure the originating force, we will use a set of strain gauges to measure the stress applied to the forearm and provides a voltage directly relating to the force applied. Next, we will send this voltage differential value through our raspberry pi connection to the opposite arm system. Using this number, the opposite arm will turn the voltage value into a torque through a few simple mathematical calculations. Once the torque number is acquired we will set a duty cycle using our ATmega328 and the PWM function. This duty cycle will relate to an output torque and furthermore through the torque constant it will relate to a torque. In order to ensure this output current/torque is always correct we will also have an inner feedback lack constantly sensing the current being pulled by our motor. If these currents do not match we will have to adjust our duty cycle accordingly to recreate the correct output torque. Essentially our system will be creating a battle of two torques.

We will also incorporate an accompanying application and database to go along with the arm wrestling match. Through the application the user will be able to create an account and connect with your friends. The user will also be able to view your match statistics and overall wins and losses. Lastly you will be able to view current games and corresponding arms in order to set up matches with other users in your tier.

  In designing our system, we considered two options, one being the above theory, and the other the use of haptic technologies. Using haptic technologies would have led to a much more expensive option as there is very little extra design involved and would have been too pricey for this design project.

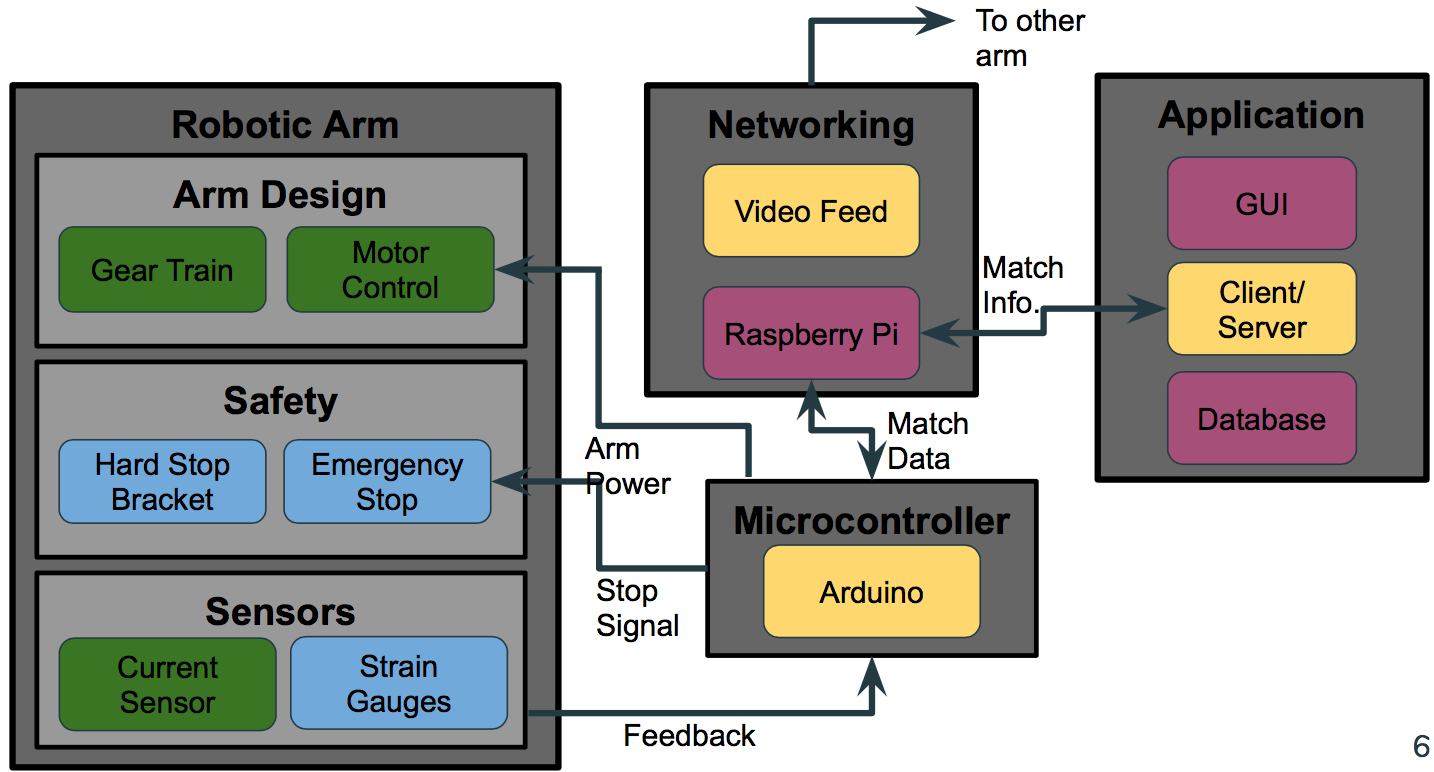


Figure 2. Block Diagram

## Force Sensing and Replication

  Force sensing and replication is an essential component for the success of R.A.W. Strain gauge sensors were selected in order to determine the change in mechanical resistance (tension and compression) on the aluminum cantilever beam which represents the forearm of our robotic arm. We decided to strategically place these strain gauges close to the fulcrum of the beam in a full wheatstone bridge configuration to retrieve the most optimal results [2]. The strain gauges chosen to perform this task were the Bf350 High-precision Pressure Resistive Strain Gauge [3].

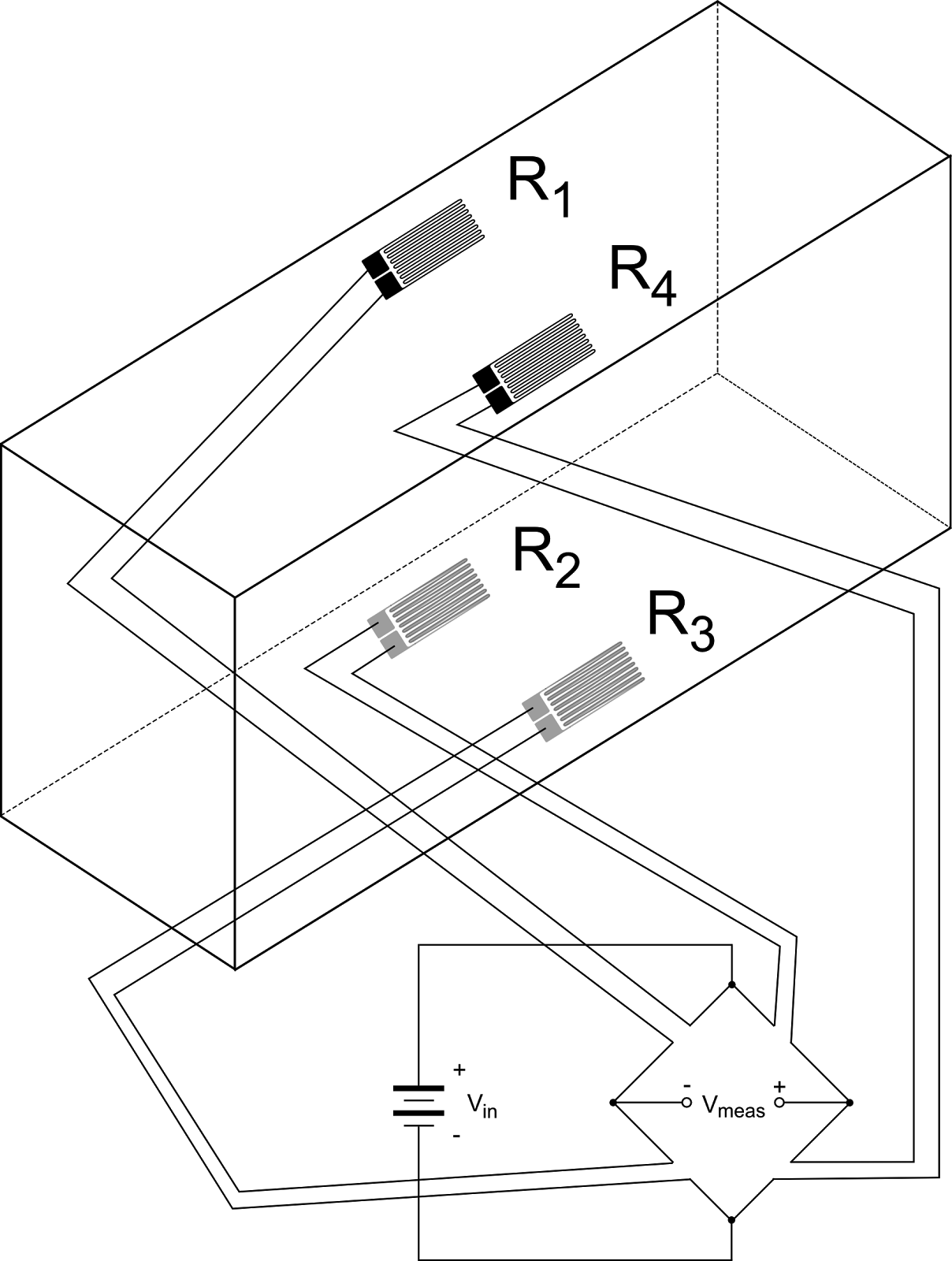
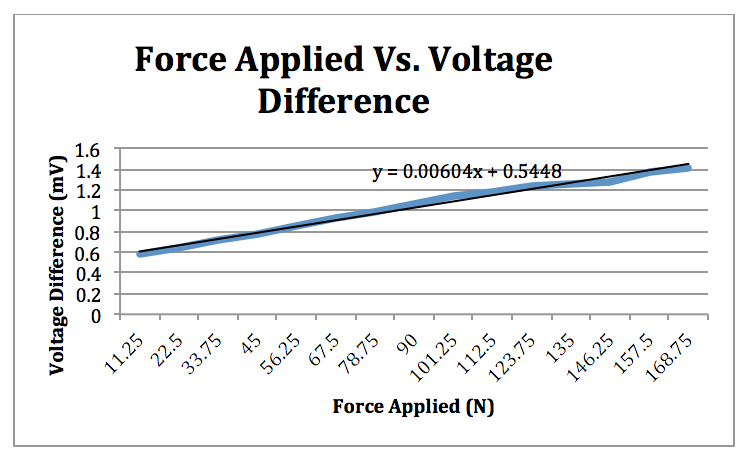


Figure 3. Full Wheatstone Bridge Strain Gauge Schematic

    The test performed to test the strain gauges involved supplying a constant voltage to the strain gauge circuit, clamping the aluminum beam to the table and hanging different amount of weight from the end of the beam. The varying amount of force applied to the beam resulted in change in mechanical resistance towards the beam. With a constant voltage supply and our strain gauges connected to a circuit board in a full wheatstone bridge configuration, we are able to measure the voltage differential at the output of the circuit. A table is displayed below showing the comparison between applied force to the beam and the resulting output voltage differential as well as the corresponding graph displaying the trendline over all of our samples (Graph 1).



Graph 1. Force Applied vs. Voltage Difference in Strain Gauges

    The results of this test provided our group with the anticipated force values, given a voltage differential. This information can be used in order to acquire the torque applied on the beam by simply using Equation 1.

        Torque = Force \* distance         (1)

    Once the value for torque is identified, we can use this value in order to calculate the necessary current to be applied to the motor to successfully replicate the torque applied by the user. We conducted an experiment to manually calculate the torque constant of the motor which can be applied to Equation 2 to calculate the input current for the motor which is later described in the current sensing subsystem.

Input current = measured torque/torque constant            (2)

## C. Safety

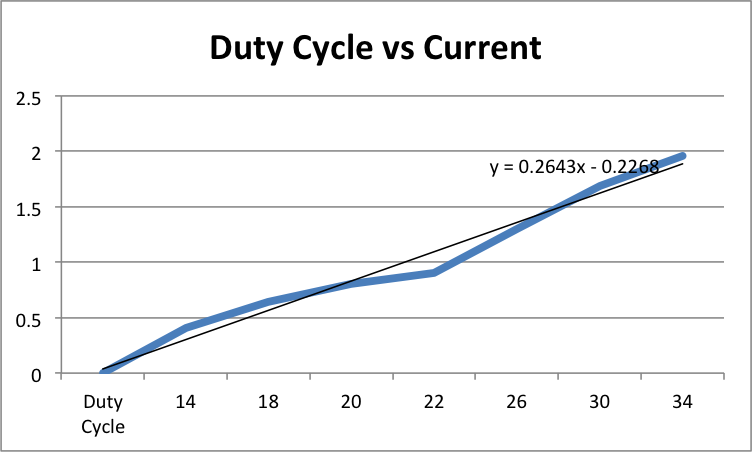
The next system in our block diagram addresses the safety concerns while constructing this project to ensure a safe and fun experience. We are going to incorporate a hard stop metal bracket safety mechanism surrounding the elbow joint of the robotic arm. The functionality of this mechanism is to disallow the robotic arm or user to advance past 45 degrees.

Additionally, a stop button will be attached to a gripped handle for the users to hold in their left hand. The button will be connected to the stopping mechanism as well so if for any reason the user would like to have an emergency stop to the game, the button can be pressed and the gear train for rotating the motor will be stopped. The stopping mechanism will consist of metal brackets with loaded springs attached to each end. The springs will be kept held back by an electrical magnet and while power is on, the magnets will continue to keep the springs loaded. If power is cut off or for any reason the game is malfunctioning, the magnets will be shut off, thus releasing the springs and initiates the metal bracket clamping.

Lastly, since we can sense the amount of force being applied over a certain time, we can calculate a specific acceleration of the arm. We will set a specific parameter to be implemented into our Arduino code to ensure that the acceleration of the arm does not exceed a specified amount. A function in the arduino code will be implemented to account for the emergency stop mechanism given the cases of lost power, emergency stop, and over acceleration of the arm.

## D. Motor Control and Gear Train

We implemented a Cytron motor driver combined with an ATmega328, for MDR we used the Arduino Uno which incorporates the ATmega328, in order to correctly drive our motor [4]. Using the PWM (pulse width modulation) functionality of the Arduino Uno we were able to send a modulated signal towards the motor driver which interprets this signal and drives our AmpFlow E50-150 brushed DC motor [4] with the correct amount of current needed. In order to control the current we simply have to adjust the duty cycle. The duty cycle will simply raise or lower the average output voltage to the motor which then, depending on the load, will produce a desired current. In order to recreate the correct voltage at a high enough speed to simulate a real time arm wrestling match we used a PWM frequency of 20kHz. We implemented many tests at multiple different applied torques so that we were able to acquire the corresponding current as well as torque associated with the accompanying duty cycle (seen in graph 2 below).

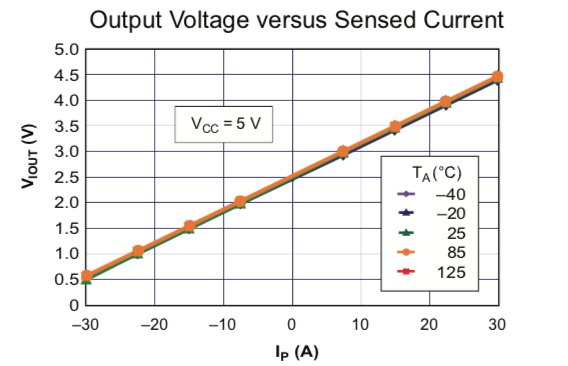


Graph 2. Duty Cycle vs. Current

The gear train will be needed to increase the output torque and lower the rotational speed of our system. In order to accomplish the correct amount of torque to rotational speed ratio we will need to appropriately size the gears in order to achieve our highest possible torque of ninety newton meters. With the motor output torque and rotational speed being 1.25 newton meters and five thousand rpms respectively we needed to achieve a gear ratio of just over seventy. The three stages had their own gear ratios of 1:6, 1:3.6, and 1:3.6. After multiplying the three stages together you get an overall gear ratio of 1:77.76, which in fact will allow us to generate enough torque with a low enough rotational speed to accurately recreate an arm wrestling match.

E. Current Sensing

The final step in ensuring the proper recreation of force will be the implementation of our current sensing and inner feedback control loop. We will incorporate a ACS712 current sensor that will use a hall effect sensing type to read the amount of instantaneous current driving the motor [5]. This current sensor will be in between the motor driver and the motor itself connected in series. Once the current is sensed it will output a voltage between -4.5V and 4.5V which will directly correspond to a current through graph 3 shown below. We will use this relationship to provide us with instantaneous feedback. When this current is ever not equal to the current we are attempting to drive our motor with the system will need to adjust the duty cycle accordingly in order to be providing the correct current and as a result the torque applied by the opposite user.



Graph 3. Output Voltage vs Current

## F. Networking

The main purpose of the networking subsystem is to send the real-time arm wrestling data between the two arm units competing in the match.  The network connection will be communicating the feedback information from the local arm, which will drive the opposite arm and vice versa.  An additional use for this network connection is to send video of each user to the opponent to be displayed.  To do this, we are using a Raspberry Pi for each arm unit.  This piece of technology is ideal because it has networking and video output capabilities.  For this communication, the Raspberry Pi simply forwards the information from the local microcontroller and the video feed to the other Raspberry Pi over the network.  Each Raspberry Pi takes the incoming data from the network and sends it to the microcontroller for controlling the arm or the video display.

To implement this connection, we will use the knowledge we learned in ECE 374 that pertains to establishing a host-to-host connection over the internet.  To test this block, we plan to implement a delay in the connection that will simulate a long-distance network connection.  This is necessary because the arm wrestling game must still function in perceived real-time by the user.  We will analyze this by testing the game ourselves along with comparing the delayed arm response time to the average human’s perceived motor function delay.

The secondary purpose for this block is to connect the phone application to the server and common database.  This is used to create user accounts and set up matches to be played on the arms.  There will also be a network communication between the server and arm units to initiate matches and upload match results at the conclusion of a match.

To achieve this, we will be creating a server and database through AWS [8] and implementing a client/server architecture like we learned in ECE 373.  The client for the connection will be Android phone applications and the Raspberry Pi’s at each arm.  The android phone application will provide the user with an interface, as discussed in the application block of this section.  The client on the Raspberry Pi will be used to initiate a match scheduled in the database and to upload the results of the match.  We will test the server with both the Android application and the Raspberry Pi’s simultaneously to ensure that the server can handle at least access from two separate arms and several phone applications at once.  This is sufficient for showing this product off in its prototype form and for our purposes of any demonstrations we may have.

## G. Microcontroller

The microcontroller has the main function of controlling the arm and collecting the feedback from the arm.  For this functionality, we have chosen to use the Arduino Uno with the ATmega328 chip and will eventually integrate only the ATmega328 chip into our PCB when our design is finalized.  Using the ATmega328 chip with the Arduino Uno at this stage allows for much simpler reprogrammability of the chip, which is critical at this stage since our code isn’t finalized.

To drive the arm, we are sending a pulse-width modulated signal (PWM) from the chip.  The arm driver powers the arm based on the percentage of the pulse that is high.  This percentage, or pulse-width is derived from the force on the “forearm” of the arm unit and the current that is driving the motor at the moment.  The ATmega328 uses these inputs to calculate the PWM and sends the signal to drive the arm.  To ensure that this is being calculated correctly, we can close the feedback loop on one arm.  If it is working correctly any force applied to the arm would be quickly matched by the motor driving the arm and the arm would stay stationary.

## H. Application

The main purpose of the application block is to create an Android application that will be used by the users of the system. This application will allow the user to create a profile. When creating the profile, the user will be placed into a specific strength tier. The strength tier will be chosen after performing a strength test on the arm. This is so the matches are always as competitive as they can be. The next capability the application will have is being able to connect with friends. You can send your friends, friend request based on their username. This is so that the user can schedule matches with their friends. You can also join available matches that your friends may have created. The finally capability of the application is being able to schedule and join matches. When scheduling a match, you will be asked to choose an arm, day, and time. The arms available for you to select will be based on your geographic location. They will be the arms closest to you. You will also be able to look for available matches to join. The matches available to you will be based on your strength tier. Once joining a match, you will be prompted to choose the arm you will be participating on. You will also be able to view your upcoming matches and match history. After having participated the result of the match will be sent from the Raspberry Pis associated with the network connection, to the server so it can be stored in the database. This is done so when the user views their match history they can see if they won or lost each match and to whom.

The GUI and Client aspects of this application will be written in Microsoft Visual Studio, using the Xamarin platform for application development. [7] The code for this part of the application will be written in C#. The server and database will be hosted on an amazon web server. This will be done by using AWS for both. The database will be written in SQL. There will be three tables; profiles, matches, and friends. The profile table will hold all facts of an users profile. This will consist of attributes such as first name, last name, username, password, and strength tier. The matches table will consist of attributes such as participant 1 and 2, arm 1 and 2, day and time. Finally, the friends table will hold the username and their friend’s usernames. This program will be tested using the Android debugging that comes with the Xamarin platform. Test accounts and will be made and tested so that any errors in the code can be caught and fixed.

The courses that will be used in order to assist with the creation of this application is Software Intensive Engineering and Integrated Systems Engineering. In Software Intensive Engineering, we learned about making an application from frontend to backend. We will use the information gained to create the application for this system. We will also use the software development lifecycles we learned to make sure the code is being designed and tested in stages. In Integrated Systems Engineering we learned about making requirements for systems. The system requirement we will create will be derived from the stakeholders of the program. Based on the system requirements we find, we will insure that application satisfies all of these. What we will need to learn more about is integrating our application with AWS so that the application can be used by multiple users at more than one time.

# Project Management

|  |  |  |
| --- | --- | --- |
| MDR Goal | Accomplished | To be accomplished |
| Arm with basic movement | * Arm is powered with variable power | * Close feedback loop to resist force on arm |
| Application with local database | * GUI with simulated functionality | * Completed all MDR goals |

Table 3. MDR Goals

For our MDR goals, we accomplished everything we promised with the application.  For the arm unit, we had promised to close the feedback loop with one arm so that the arm would resist any force provided.  We did not achieve this goal, but we did power the arm and collect all feedback except for the current going into the motor.  The main reason we were not able to deliver on this deliverable was that we received the current sensor extremely close to the demonstration for MDR.  As such, there was no time to integrate the sensor and close the feedback loop.  We did provide an arm that had basic movement provided by the motor, though.  This was one of our promised deliverables and the most important step moving forward.  Although the arm did start in the starting position, as was our promised MDR deliverable, we need to lock the arm into that starting position before a match.

The team started working as two sub teams.  Costa and McGrath began working as a team to build the arm and get all of the electrical components wired up and working.  Matthews and Lucey began working on creating a phone application to be used in conjunction with the arm.  After the arm was built and wired, we all began working on getting the arm ready for the MDR demo.  This included connecting an Arduino Uno to the motor driver and testing out various PWM’s.  We also hooked up the sensors to the Arduino Uno to analyze the data being received.  Although we didn’t use that feedback to power the motor we had already looked at the data being provided by them.

Over the course of this process we have found each of our specializations for future work on this project.  Matthews is specializing in application development.  Lucey is specializing in programming of the microcontroller and Networking.  Costa specializes in motor control and arm construction.  McGrath specializes in force replication and feedback, as well as safety implementation.  Although many of these areas are not done alone, the group member with the specialty in that area is the one who does most work and heads that part of the project.

To keep the team up to date on the various parts of the project that were being worked on, we had semi-regular meetings on Wednesday nights.  We also used this time to brainstorm any new ideas or solutions to problems we may have been facing at the time.  In addition to those meetings, we would meet as needed to communicate any issues either sub team had run into.  The sub teams themselves met on a very regular basis to actually work on the project and split up work for when they couldn’t meet.  Outside of actual meetings, we often used group text messages and emails to communicate meeting times and smaller issues.

# IV. Conclusion

Graph 3. Gantt Chart of MDR Progress

We have currently met most of our MDR goals excluding the force replication given a human input. This goal was modified and was changed to have the arm react given a certain amount of weight pulling against it based off the provided input current for the motor. First, the arm would begin to lose, then after a couple of seconds, we would have the arm stall and stay in a locked position. Finally we would provide enough input current to the motor in order to beat the given amount of weight. The Android application is fully functional and connected with a local SQLite database so local matches can be found and created. Even though most of our MDR goals were successfully met, we still have more to accomplish up until CDR.

Our next step forward for R.A.W, is to construct an additional fully functioning arm with the motor control and gear train fully implemented. The two arms will have to be completely networked with force replication values being read from one arm and fed into the other via Raspberry Pi connection. The force replication and transferring of data between arms will be proven to be one of the most challenging piece to our project. An emergency stop mechanism will be fully operational on at least one of the arms as well.

    Going forward from a software standpoint, we plan to have the two arms networked through a Raspberry Pi connection. Each Raspberry Pi will communicate with the Arduino in order to retrieve the vital force replication values that are to be sent to the other arm. The values acquire throughout the games will be stored in the database and accessible through the functioning application.

Acknowledgment

We would like to extend a special thanks to our advisor Professor Zink, the two course administrators Professor Hollot and Professor Soules, and lastly mechanical engineering Professor Sup for his continued help with our design.

References

1. Betchen, S. J. (2015, June 14). 4 Problems With Long-Distance Relationships. Retrieved December 20, 2017, from https://www.psychologytoday.com/blog/magnetic-partners/201506/4-problems-long-distance-relationships
2. Wheatstone Bridge. (n.d.). Retrieved from <https://www.transducertechniques.com/wheatstone-bridge.aspx>
3. [Strain Gauge Datasheet](http://www.electronicaestudio.com/docs/SHT-Y3.pdf)
4. [Cytron brushed DC motor driver datasheet](https://www.robotshop.com/media/files/images3/md30cusersmanual_1_.pdf)
5. [ACS712 current sensor  datasheet](https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf)
6. Soper, T. (2014, August 08). Study: 1.2 billion people are playing games worldwide; 700M of them are online. Retrieved February 05, 2018, from <https://www.geekwire.com/2013/gaming-report-12-billion-people-playing-games-worldwide/>
7. <https://www.xamarin.com/>
8. <https://aws.amazon.com/>