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 the user with the same fun and excitement of an in-person arm wrestling match, except the game will be played over a distance between a user and a weighted pulley system. There will be one robotic arm powered by a motor, which drives a series of gears, and is attached to a pulley. There will also be a fixed are, with a forearm that will have strain gauges on it to find the force applied by the user. The motorized arm, will replicate the force being applied by the user to the fixed arm. There is also a phone application that allows a user to make a profile and join/schedule matches. The arms will query the database to find any matches for that day and allow the users to join their match.

# 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 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

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)

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 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 motorized 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 motorized arm and fixed arm both 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 also must 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 one 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 inside a 50 kg load cell to measure the stress applied to the forearm of the fixed arm. This will then, provide a voltage directly relating to the force applied. Using this number, the fixed arm will turn the voltage value into a torque through a few simple mathematical calculations within the ATmega328. Once the torque number is acquired we will set a duty cycle using our ATmega328, once again. Next, we will send this duty cycle value through our Raspberry Pi connection to the opposite motorized arm system. This duty cycle will relate to an output torque and furthermore through the torque constant it will relate to a torque. Once the duty cycle is received by the motorized arm, the duty cycle will be used within the PWM function within our ATmega328. The ATmega328 will then send the direction and PWM to the motor driver of the motor, thus driving the system with that torque.

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.

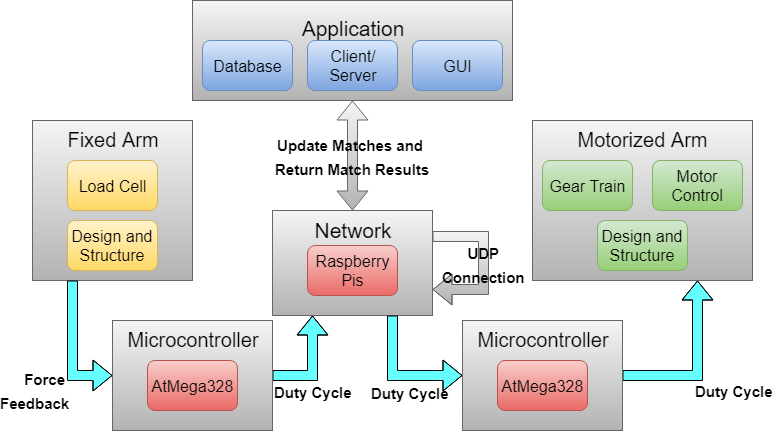


Figure 2. Block Diagram

## Fixed Arm - Force Sensing and Replication

  The fixed arm was created in order to create an alternative design from the original of idea of R.A.W. The fixed arm is comprised of a 50kg load cell [2], a metal bearing attached to the load cell, and a wood handle representing the length of an average persons forearm (0.3 meters). The way the fixed arm operates is to apply force to the handle of the arm which will activate the load cell. The load cell has internal strain gauges attached to it in a Full Wheatstone Bridge Configuration (Figure 3). The strain gauges work as variable resistors in the sense that a change in stress put on the load cell will result in a change in resistance provided by the strain gauges. When there is a change in resistance, there will be a resultant output voltage differential.

Furthermore, the output differential resulting from the user applying force to the arm will be inputted into the INA128P Instrumentation Amplifier (Figure 4) [3]. The instrumentation amplifier will provide enough gain to create a readable value to provide to our ATmega328. The ATmega328 will contain code within to alter this value read in into a force value. The force value will then be used to create a PWM signal relating to a duty cycle. The duty cycle will be sent over the network via Raspberry Pi to drive the motorized arm which will be discussed later on in the report.

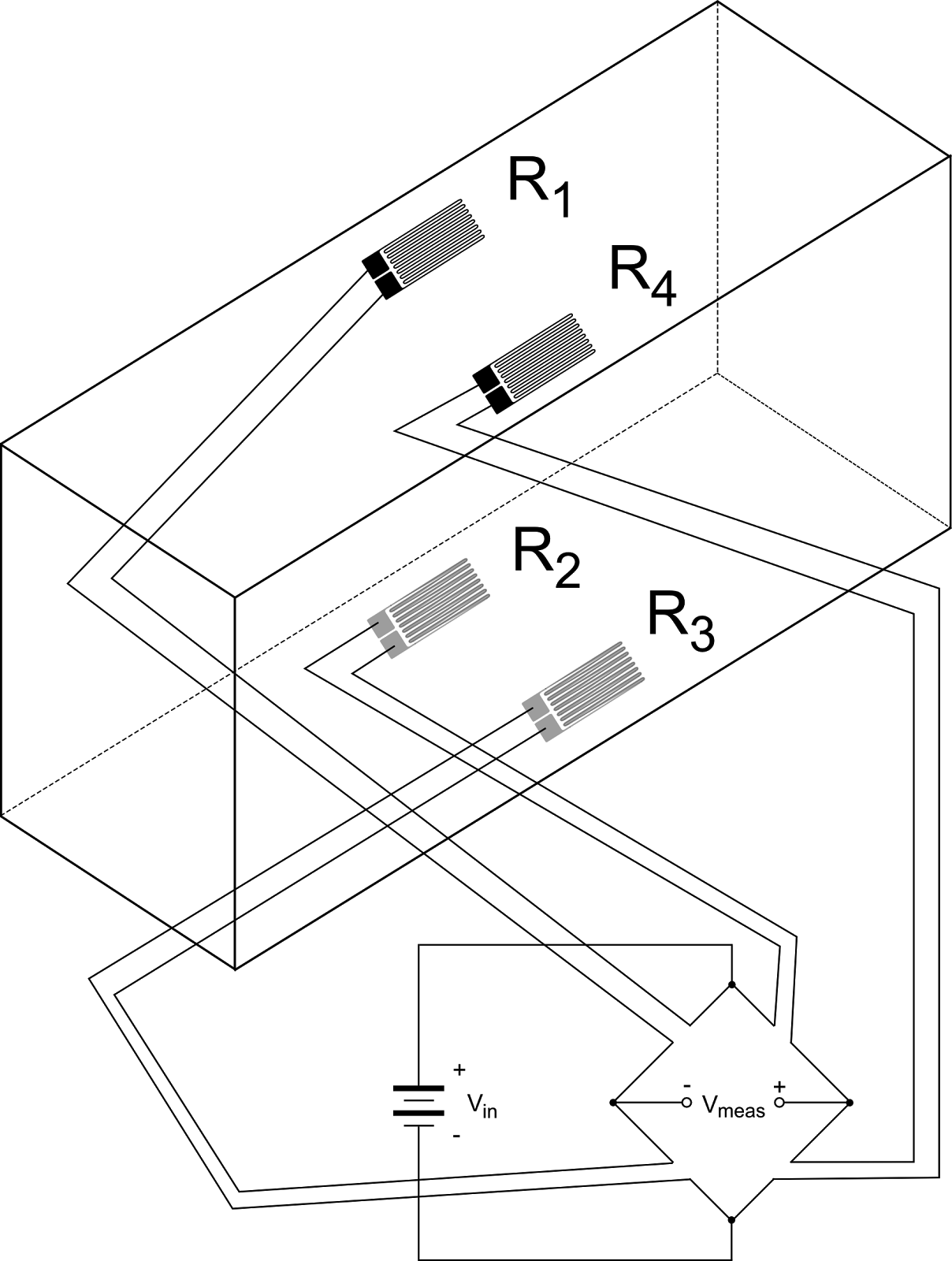


Figure 3. Full Wheatstone Bridge Strain Gauge Schematic

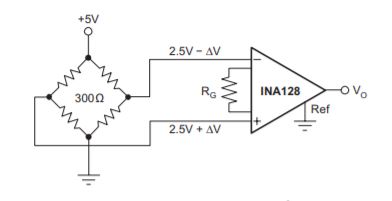


Figure 4. INA128P Instrumentation Amplifier connected to Full Wheatstone Bridge

    In order to properly drive the motor with the correct replicated force, we needed to test and map out several relationships. First, we applied different load to the fixed arm to get an initial correlation between pounds and voltage differential readings from the load cell (Graph 1).

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. Voltage Differential vs. Pounds

        From here, we needed to further map this relationship to a duty cycle. We converted pounds to force using Equation 1.

         Force = Pounds \*4.48         (1)

    Next, we converted this value to torque using Equation 2.

Torque = Force \* 0.235 meters (forearm length)            (2)

The results from this testing allowed our group to successfully map out an equation to relate voltage differential to applied force. This relationship then allowed us to properly correlate applied force to a PWM signal to drive the motorized arm with the correct power to accurately represent the force that the user was applying.

## D. Motorized Arm - Motor Control and Gear Train

We implemented a Cytron motor driver combined with an ATmega328 in order to correctly drive our motor [4]. Using the PWM (pulse width modulation) functionality 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.

The last step was to create an encasement around our arm as well as a plexiglass cover in order to not only look nicer, but also ensure safety during operation. In the figure seen below you are able to see the outside walls, support walls for our bearings, and the plexiglass cover which sits just inside the four walls. To hold up the cover we implemented blocks on each of the load walls as well as small handles in the back for it to rest on.

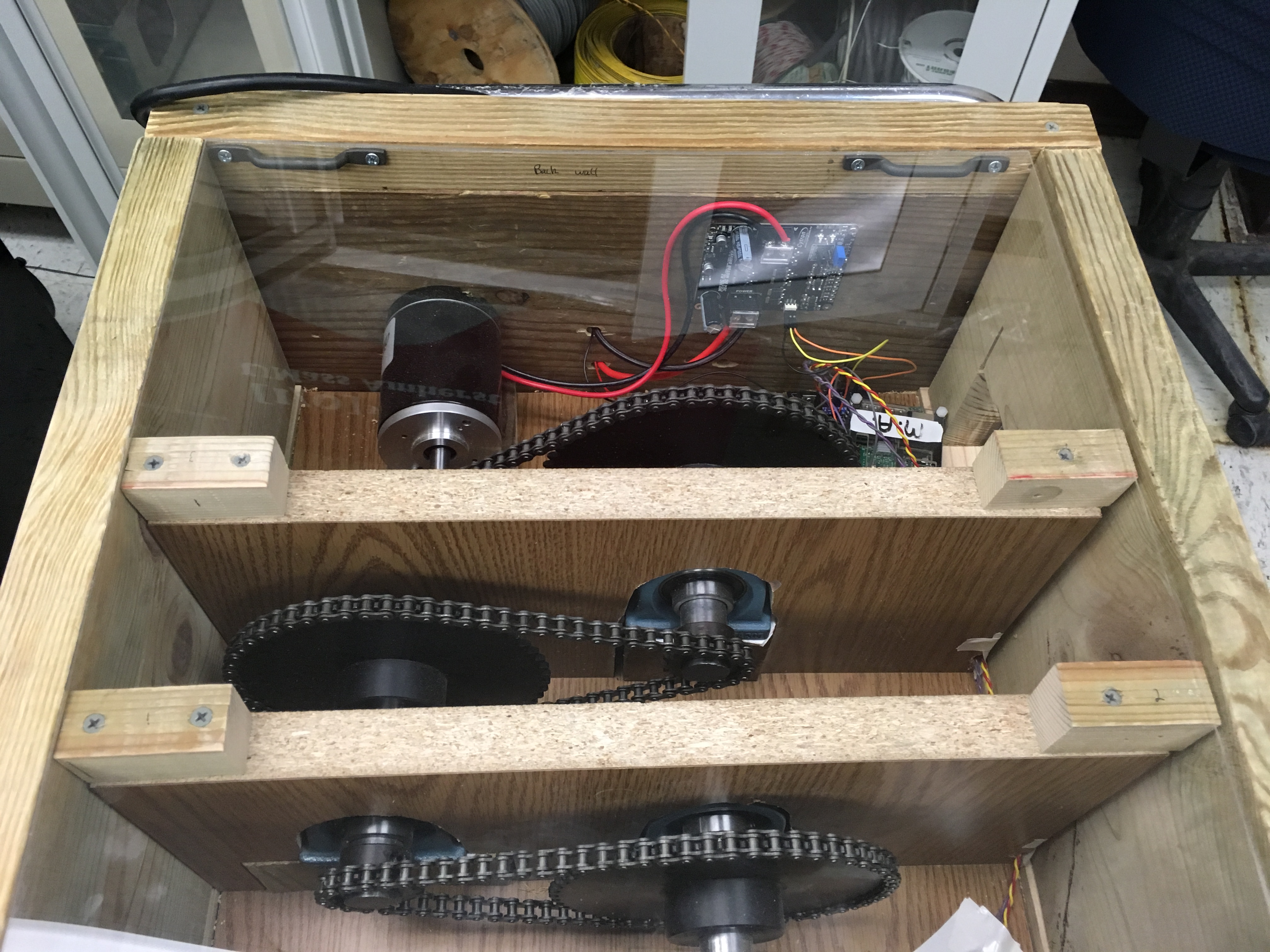


Figure 5. Motorized Arm

## E. Win/Loss Signaling

In order to be able to know the result of each match, we needed to implement a way to record the win or loss of a match. The way we did this was by developing two trip wires on either side of the motorized arm (one for win, and one for lose). The way the trip wire was implemented was by using two laser pointers [10], and two 30 kiloohm photocells. In the figure below you can see the set up. The two photocells are in the wall, on the left and right of the arm. The two laser pointers are set up on either side, across from the photocells to set up the trip wire.

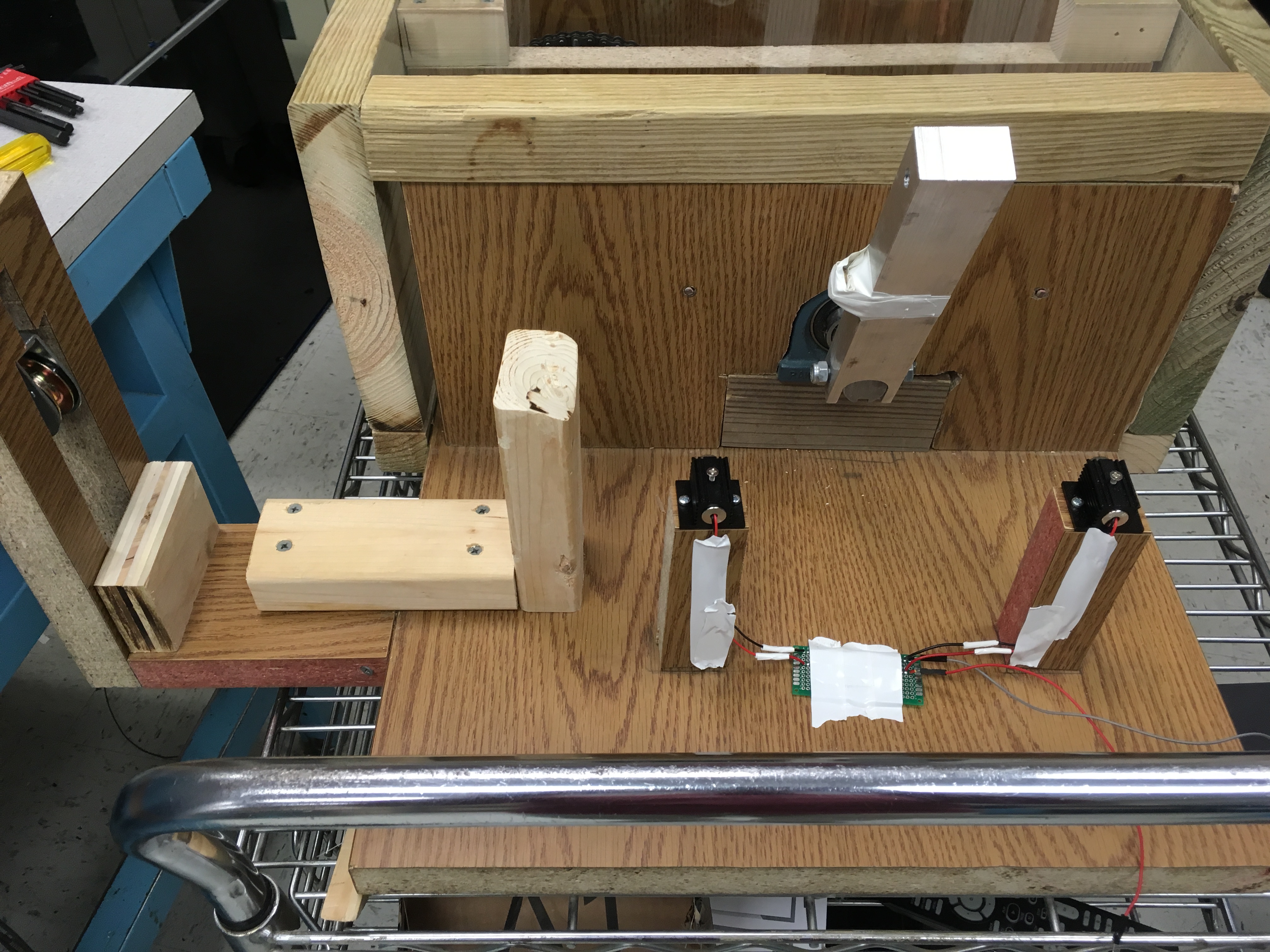


Figure 6. Win/Loss Signaling Configuration

Each of the photocells are connected to power on one leg, and to a 10 kiloohm resister, that is connected to ground on the other leg. The point where the leg of each photocell meets the resistor, is then connected to an analog pin on the ATmega328. These analog pins read a value of around 950 when the laser is hitting them. When the plain of the laser hitting the photocell is broken, then those readings drop to 350. Once this read happen, the code on the ATmega328 stops sending PWM signals to the motor. Instead the ATmega328, will return a win or a loss (depending on which plain was broken) message back to the Raspberry Pi. The Raspberry Pi will then carry this message across the network to the fixed arm. Once reaching the fixed arm, the GUI on the screen that the Raspberry Pi is connected to will display the match result message and the match is over.

## F. 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 re-programmability 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.

## G. Networking

The main purpose of the networking subsystem is to send the real-time armwrestling 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, if the project was continued, for this network connection is to send video of each user to the opponent to be displayed. For networking functions, 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 would implement a delay in the connection that will simulate a long distance network connection if given more time. This is necessary because the arm wrestling game must still function in perceived real-time by the user.  Although we never implemented adding a delay to simulate long distance network connection we found the arm was very responsive over the campus wifi connection. For the purposes of demo day, we created an adhoc wifi connection between the two Raspberry Pi’s.  This minimized interference and delay while the wife was bogged down by all the projects of demo day. This would only work at short distances though, so it was only useful for the purpose of demo day. For the intended purpose, a connection to the internet would be required which we demonstrated during FPR.

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 is also 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 created a server and database through AWS 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 provides the user with an interface, as discussed in the application block of this section.  The client on the Raspberry Pi initiates a match scheduled in the database and uploads the results of the match. We tested the server with both the Android application and the Raspberry Pi’s simultaneously to ensure that the server could handle multiple accesses at the same time.  This is sufficient for showing this product off in its prototype form and for our purposes of any demonstrations we may have.

## H. Match Login

In order for the user of the match, to be able to participate in the match, the fixed arm Raspberry Pi must have a GUI attached to it. This GUI is very basic as it just needs to take in the user’s username. In figure below, you can see the GUI that is used.



Figure 7. Login GUI

This GUI is created using wxPython [11]. This a package used in Python for GUI creation. This GUI is then connected to a client that connects to our server.

This client-server connection is done through an XML-RPC connection. This version of a client server connection was chosen for it simplicity to implement. The function calls that the client can make to the server are, request the match for the user that logged in and update the result of the match.

The server is then connected to the same Microsoft SQL Server database that the application is. The server will pull match data and update match data as needed.

## I. Application

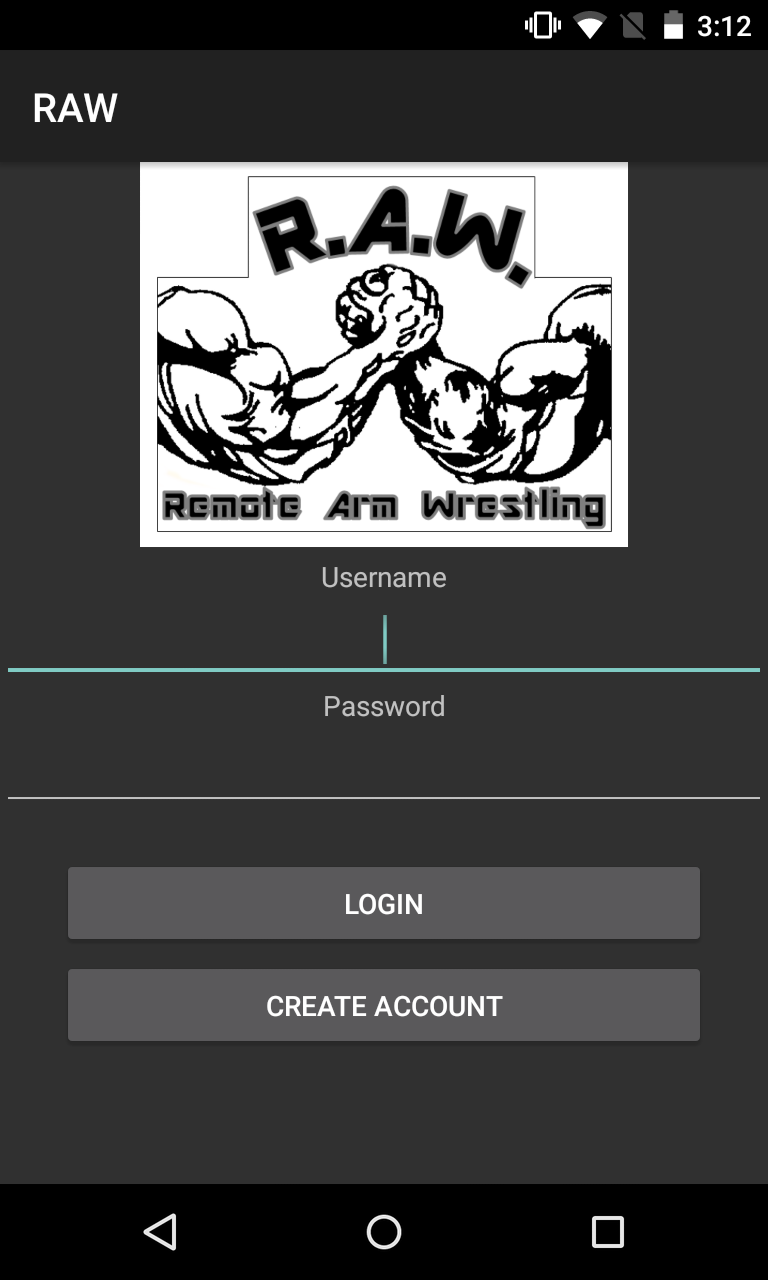


Figure 8. Login GUI for Application

In the figure above there is an image of the login screen for the 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 based on the weight of the user. This is so the matches are always as competitive as they can be. The next capability of the application 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 database will be a Microsoft SQL Server hosted by AWS (Amazon Web Services). [8] The database queries are written in SQL using a package in Microsoft Visual Studio that allows for easy Microsoft SQL Server connections. There will be three tables; profiles, matches, and connections. The profile table will hold all facts of a user’s 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 connections table will hold the username and their friends’ usernames. This program was 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.

In order to debug the database, we used Microsoft SQL Server Management Studio. [9] In the figure below is a display of the matches table within Microsoft SQL Server Management Studio.

A screenshot of a computer

Description generated with very high confidence

Figure 9. Matches Table

As you can see this environment allows you to see every column in each table. It also allows you to write SQL queries to organize the table in different ways so you can see the data from different perspectives.

The courses that were 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.

# Project Management

In order to complete our system more efficiently the team started working as two sub teams.  Costa and McGrath began working as a team to build the arms 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. Matthews and Lucey also created the network connection between the two arms, and connected the database to the arms.

To more clearly define each members expertise to the final product it broke down as such: Costa was completely in charge of the motorized arm. McGrath was completely in charge of the fixed arm. Lucey became more heavily in charge of the network connections. Finally, Matthews was in charge of the application, the GUI, client/server and database connections for the arms. Matthews was also team leader and was in charge of keeping the team on pace.

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. We also utilized tools like Gannt charts and other organizational tools to stay up to date.

The team members of RAW complimented each other very well, and had an excellent group dynamic. Every member had something unique to bring to the table. The greatest strength of the team was, how well we helped each other out and tried to become involved with every facet of the project. On more than one occasion a team member who was struggling was able to get back on track through the help of the team.

# IV. Conclusion

Since MDR we have implemented a fixed arm with a load cell to receive our force input rather than a simulated user.  We have also completely switched to ATmega328s rather than our Arduinos. Another big step we took was the implementation of a much more complex pulley as well as LED win/loss signals. We have also completely enclosed our motorized arm with wood walls and a clear plexiglass cover to ensure safety of users. Another improvement was fully implementing our own power source, which is capable of powering our system through a wall outlet.

  We have also implemented a full ad-hoc network for the two Raspberry Pis. The protocol that the Raspberry Pis communicate on is a UDP protocol. This protocol was chosen because there was less delay in this than in TCP. Having a low latency was very important for our system and UDP helped us achieve that.

The application, since MDR, has been moved to having all of its data hosted on AWS. For MDR the application accessed a local database within the phone itself. The application also had the functionality of creating a match change. The arms were changed to locations instead of just numbers, and the day and time picking switch to a pop-up button instead of manual input.

We have currently met all of our final goals at this point. Given an input into our fixed arm we accurately read the force input using our load cell and converted this to a duty cycle. This duty cycle was then sent over our wireless network and replicated through the motor. Attached to the motor arm was weight hanging from a pulley, which would only be moved once you reached or exceeded the amount of weight. We also implemented a real time graph of your force input in order to constantly view if you needed to input more force to beat the weight. Lastly we have two LED lasers pointing directly at photo resistors so that when the arm crossing either plane our microcontroller receives a signal, and ultimately triggers a win/loss signal. This is then displayed through our GUI with You Win! Or You Lose! This GUI is connected to an XML – RPC client/server. This server accesses the same matches table as the application. This client/server is in charge of finding the correct match for the user and then updating the results.

Appendix

## A. Application of Engineering

There are many areas of math, science, and engineering that apply to RAW, most notably: classical mechanics, circuit analysis, electronics, computer networks, hardware organization, data structures and algorithms, and system software design. For our software development portion of RAW the Android development was C#, the XML-RPC client/server and login GUI where done in Python and the software for the microcontroller was written in C. We have exposure to these programming languages through the coursework in ECE 122, ECE 242, ECE 353 and ECE 354. The power regulation circuitry design and the integration of strain gauges used for force feedback where crucial for the implementation of our system. These elements of RAW would not have been implemented correctly had we not had previous knowledge of circuit design, which we gained in courses like ECE 211, ECE 212, ECE 323 and ECE 324. Finally, a vast amount of mechanical engineering was needed to be able to build the gear ratio and motorized arm. The course MIE 497R was useful in being able to learn more about the relationship between mechanical and electrical engineering.

## B. Cost

Below is our total cost for the motorized arm of our system with the strain gauges added. This is what would be bought by a bar or arcade.

TABLE II

Cost

|  |  |
| --- | --- |
| Item | Price (dollars) |
| PCB | 25 |
| Motor | 79 |
| Driver/Power Supply | 81 |
| Load Cell | 14 |
| Gears Chain  Drive Shaft  Bearings  Key Lock  Forearm  RPi 3  Encasement  Win/Loss Signals  **Total** | 124  27  28  4  46  35  70  15  **573** |
|  |  |
|  |  |

For production grade we would expect this price to drop closer to 300 dollars.

Acknowledgment

We would like to extend a Special thanks to Professor Michael Zink. Thank you to professors Christopher Hollot, Baird Soules, Weibo Gong, and Russel Tessier. Thank you to Professor Frank Sup for help with the mechanical design. Also thanks to Fran Caron.

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
4. Cytron brushed DC motor driver datasheet
5. ACS712 current sensor  datasheet
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/
9. https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017
10. https://www.amazon.com/gp/product/B01CH53Y08/ref=oh\_aui\_detailpage\_o00\_s00?ie=UTF8&psc=1
11. https://www.wxpython.org/