Final Report

Wirelessly Steered Robotic Arm

EECE498 - Electrical Engineering Design Project

EECE499 - Computer Engineering Design Project

Spring 2014

Team "Γ I Σ" Members:

Gamal Abu Saif 1004018438

Joseph Bejjani 1002018289

Sharbel Dahlan 1004018456

Project Supervisor: **Dr. Adnan El Nasan**

Due date: **May 1, 2014**

Abstract

The need for the robotics to aid people in different daily tasks is increasing. Today, building a robotic arm to flex and extend its fingers, roll, and pitch similar to a human's arm simultaneously, with the use of an augmented glove worn by a person, is possible. This can work by the glove being fit with sensors to sense the human arm motion and action, the robotic arm with strings and actuators, and a control unit on each side to record the state of the human arm and actuate the servo motors to move accordingly. The sensors used in the augmented glove include a flex sensors on each finger to detect the finger flexion and an Inertial Measurement Unit(IMU) on the wrist to detect the rolling and pitching of the arm. The actuators on the robotic arm include a low-torque servo motors attached to a string that is tied to each finger for the flexion and extension, and two high-torque servo motors to do the rolling and pitching. The two control units that are used on each of the augmented steering glove and the robotic arm include Arduino Mega and Arduino Uno, respectively. Both microcontrollers communicate with each other wirelessly through ZigBee technology, using an XBee transceiver each to transmit the data from the steering glove microcontroller to the robotic arm microcontroller. With the wireless communication achieved between the two microcontrollers, the robotic arm can be steered within a range of a 100 meters, and can be used for several applications. The currently useful for handling simple light objects system serves as a proof of concept for systems being used in larger applications like prosthetics or handling hazardous material in harsh environments.

Acknowledgments

Many thanks to Dr. Adnan El Nasan for his support and supervision in this project. We would also like to thank the professors of the Engineering department for their help and guidance through different parts of the project.

Additionally, we acknowledge our friends from Mechanical Engineering (Georges Beyrouti, Omar Labban, and Tarek Halabi) for their guidance with the .STL files for 3D printing. We also acknowledge Alexandra Semaan for her help in carrying out the process of 3D-printing the arm (Mark 3).

Big thanks to our colleagues, especially Mustafa Alloh and Omar Hegazi, for their feedback, suggestions, and support through the days and nights of the entire semester.

Finally, special thanks to our families for their support in the project, especially Sharbel's Grandma, Georgette, for stitching the pockets to carry the flex sensors on the glove.

FIGURE 1: Grandma Georgette's contribution to the project with the stitched pockets on the glove.

May 2014

Gamal, Joseph, Sharbel

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increase of useful applications of Robotics to serve humans, the need for simpler controlling mechanisms increases. Different fields of study are dedicated to improve the communication between humans and robots, one of which is Human-Robot Interaction (HRI). Being dedicated to understanding, designing, and evaluating robotic systems used by humans, HRI have been working towards simpler communication that is based on intuitive controlling mechanisms. One of those mechanisms is controlling robots by hand gestures, without the need of traditional control such as through remote controllers or control panels. This mechanism is the focus of this project, in which part of a robot, which is the robotic arm is steered wirelessly with through a steering glove. The main advantages of this system over the traditional control panel boards are the simpler mechanism of control, and the minimized need for special training of humans, since they are only required to move their arm. Hence, a system was required to be developed that makes use of the the flexibility of using the human arms, and the ability of robots to lift weights that sometimes a human cannot lift. Such a requirement is based on several motivations and possible ultimate uses of the robotic arm, a few of which will be discussed next.

1.1 Motivation

One motivation behind this project was the need for robots to replace humans at certain tasks that put the human life to danger. Those tasks include handling hazardous material or working in harsh environments. These type of tasks usually require precision and time, and usually come with a heavy burden and stress, making the task much harder. For example, bomb disposal (illustrated Figure 1.1) is a difficult task that usually requires a human risking his or her life in order to diffuse the bomb.

FIGURE 1.1: Man Disposing of Bomb

However, with the use well-developed robot system that is steered by a glove, this type of task can be done. In this way, robots, such as the one shown in Figure 1.2, are remotely steered by humans from a safe distance to do the job. This would reduce the stress factor when handling these material, and would usually result in keeping the user who is controlling the robot safe.

FIGURE 1.2: Robot Disposing of Bomb

Handling hazardous material is not the only possible application of robotics which motivated this project. Robots have great potential to be used in medical fields, such as helping the elderly or people with disabilities (Figure 1.3), or aiding humans to do tasks which require more power, such as carrying heavy weights.

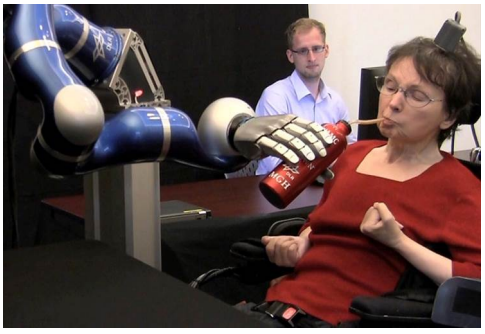
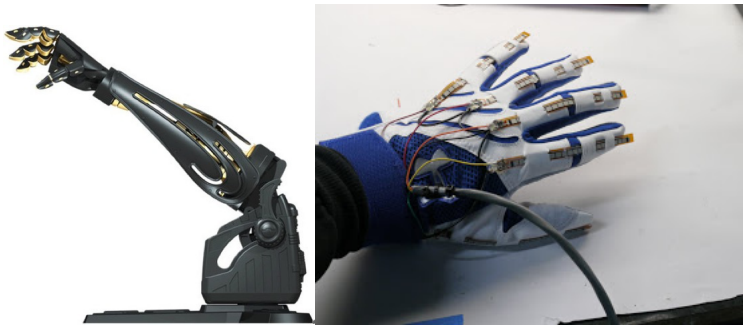


FIGURE 1.3: Robotic arm helping people with disabilities perform certain tasks.

What have been mentioned so far are applications that the would make use of the system after long-term development. The current system that is discussed in this project, however, is a prototype that serves as a proof of concept for the larger systems. This prototype targets the simple use of a robotic arm that follows the arm gestures of the controlling human and can carry light objects of up to 1 kg. Moreover, the prototype implements certain degrees of freedom of the human arm, which will be defined in the next section and described in detail in the Background chapter.

1.2 Problem Statement

This project involves a building a robotic arm (Figure 1.4a) that can roll, pitch, and flex its fingers while being wirelessly steered by an augmented steering glove (Figure 3.3).



(A) Robotic Arm (B) Augmented Steering Glove

FIGURE 1.4: The project's robotic arm system

More degrees of freedom, such as yawing, can be added as extensions to the project if both the budget and time of implementation were increased. For the purpose of the previously mentioned simple application, which involves the robotic arm carrying light-weighted objects, only the finger flexion, rolling, and pitching, are part of the project's scope. This implementation requires some biological background. Hence, before going to the proposed solution and the detailed implementation of the robotic arm system, the working mechanism of the human arm will be discussed. Based on parts from the human arm working mechanism, robotic arm system is designed.

Chapter 2

Background on the Biomechanics of the Human Arm

Before carrying on with the implementation details of the project, the background information necessary for this project needs to be discussed. The reason is that the robotic arm system will be analogous to the human arm. Hence, studying the working mechanism of the human arm is necessary. In the subsequent sections, the relevant parts from the biomechanics of the arm are briefly explained. First, the degrees of freedom of the human arm are generally described in section 2.1. Next, the flexion of the finger will be explained in section 2.2. Then, the rolling of the forearm arm will be discussed in section 2.3. Finally, section 2.4 will discuss the pitching of the arm which will be implemented in the project.

2.1 Degrees of Freedom of human arm

The human arm is a complex structure that is capable of moving in seven degrees of freedom (DOFs). Each DOF is an independent type of motion allowed to the arm structure. Figure 2.1 shows the degrees of freedom of the human arm. The shoulder gives pitch, yaw, and roll, the elbow allows for pitch, while the wrist can roll, pitch, and yaw.

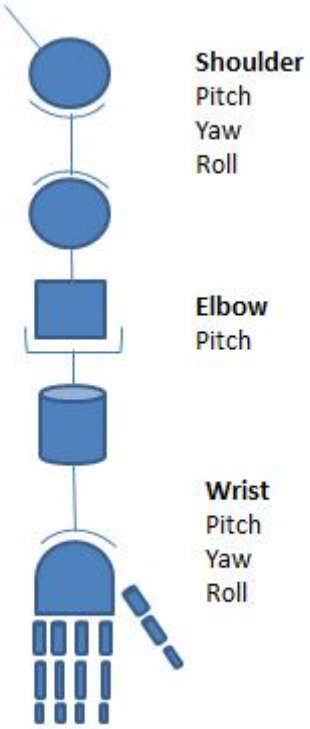


FIGURE 2.1: Degrees of Freedom of a human arm

From those degrees of freedom, the rolling of the arm, which is allowed by the muscles from the forearm, and the pitching of the elbow are relevant to this project. hence, the parts of the human arm that need to be focused on are both the hand and the *forearm*, shown in Figure 3.6.

FIGURE 2.2: The human hand and forearm, the focus structures of this project

2.2 Finger Flexion

Finger flexion involves beding of the fingers on the hand, as illustrated in Figure 2.3. The five fingers in a human hand can both *flex* and *extend*. The act of flexion and extension are achieved by the help of *flexor tendons* (shown in Figure 2.4).

FIGURE 2.3: Flexion of the Fingers

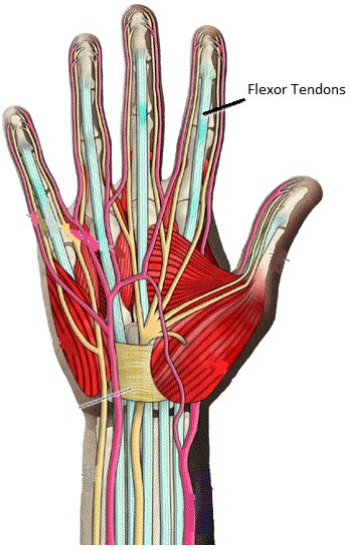


FIGURE 2.4: Flexor tendons of the finger

Generally, tendons are the structures that attach muscles to bones, thus allowing the bones to move according to contraction and relaxation of the muscles. In the case of the hand, which contains five fingers, each finger has three bones, called *phalanges*, except for the thumb, which has two.

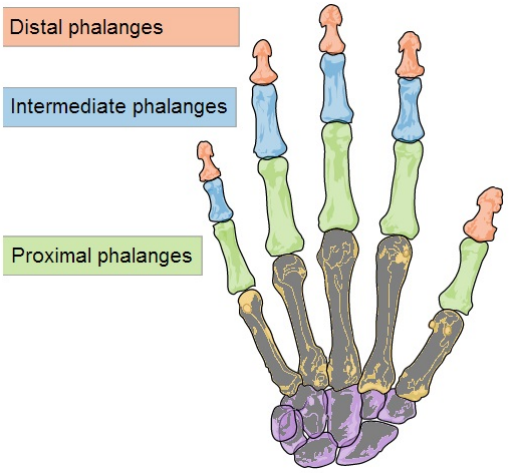


FIGURE 2.5: Phalanges of the fingers

Attached to each phalanx bone is a flexor tendon that is connected to muscles on the forearm. Once the muscles are contracted, the phalanges are pulled, hence the finger is flexed, as illustrated in Figure 2.6.

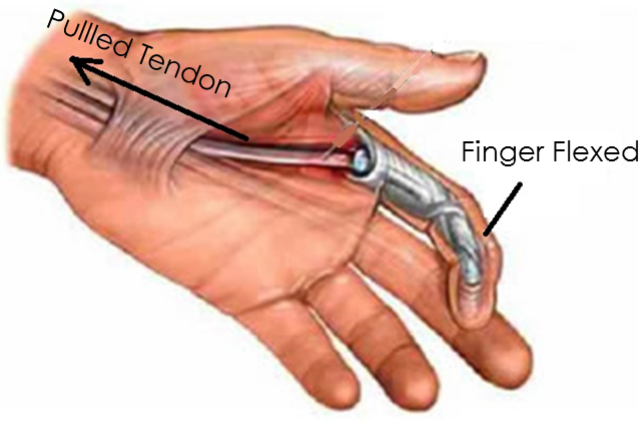


FIGURE 2.6: Flexor tendons of the finger (Edited by: Sharbel)

Having an independent tendon attached to each finger will allow independent finger flexion, based on the muscle contraction from the forearm. This mechanical energy provided by the muscles is transformed from chemical energy provided by adenosine triphosphate (ATP) molecules. In a similar manner, the rolling and pitching of the arm are achieved with the help of certain muscle contractions.

2.3 Rolling

The rolling, which is also known as *supination* and *pronation* of the hand, is shown in Figure 2.7.

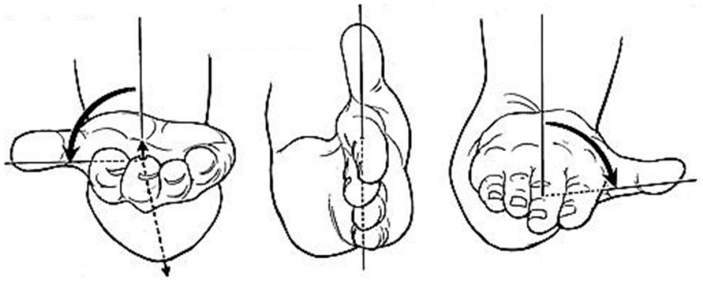


FIGURE 2.7: Second degree of freedom: Rolling of the arm

This act is also done through muscles being located in the forearm, which allow certain bones to move, thus making the forearm structure to revolve for more than 180 degrees.

2.4 Pitching

The pitching of the arm, also known as pivoting, is illustrated in Figure 2.8. This acion also occurs with the help of muscles that are located in the upper arm, namely the *biceps* and the *triceps*, both of which are linked through tendons to the forearm.

FIGURE 2.8: The pitching (pivoting) of the elbow

Similar to the mechanism of rolling and finger flexion, the pitching happens when the muscles, using chemical energy, contract an relax. When the biceps muscle contracts and the triceps muscle relaxes, the forearm is flexed, or brought nearer to the upper arm. When the opposite happens (i.e. biceps relaxes and the triceps contracts) the forearm extends.

With the biomechanics put in mind, the design of the human arm will be analogous to that of the human arm. The main components that were seen in this chapter are muscles, bones, and tendons. Different ways of contracting and relaxing the muscles results in different movements of the bones, thus positioning the fingers and arm differently.

Chapter 3

Proposed Solution

We propose to build a robotic arm which can flex its fingers, roll and pitch its arm. This robotic arm will be steered wirelessly by an augmented glove that is fitted with sensors. The robotic arm and augmented glove will communicate with one another wirelessly. Below is a detailed description of the proposed solution.

3.1 High-level Architecture

The proposed solution will have two parts: the action side, and the steering side. The action side consists of a robotic arm that will move and carry an object, and it will be controlled by a microcontroller on the action side. The microcontroller will control the flexing of the fingers and the movement of the arm (roll and pitching of the arm). The steering side is where a human will control the robotic arm through an augmented glove. This is done by also have a microcontroller that will process the commands collected from the glove and transmit them to the action side. A high level architectural view of the proposed system is shown in Figure 3.1.

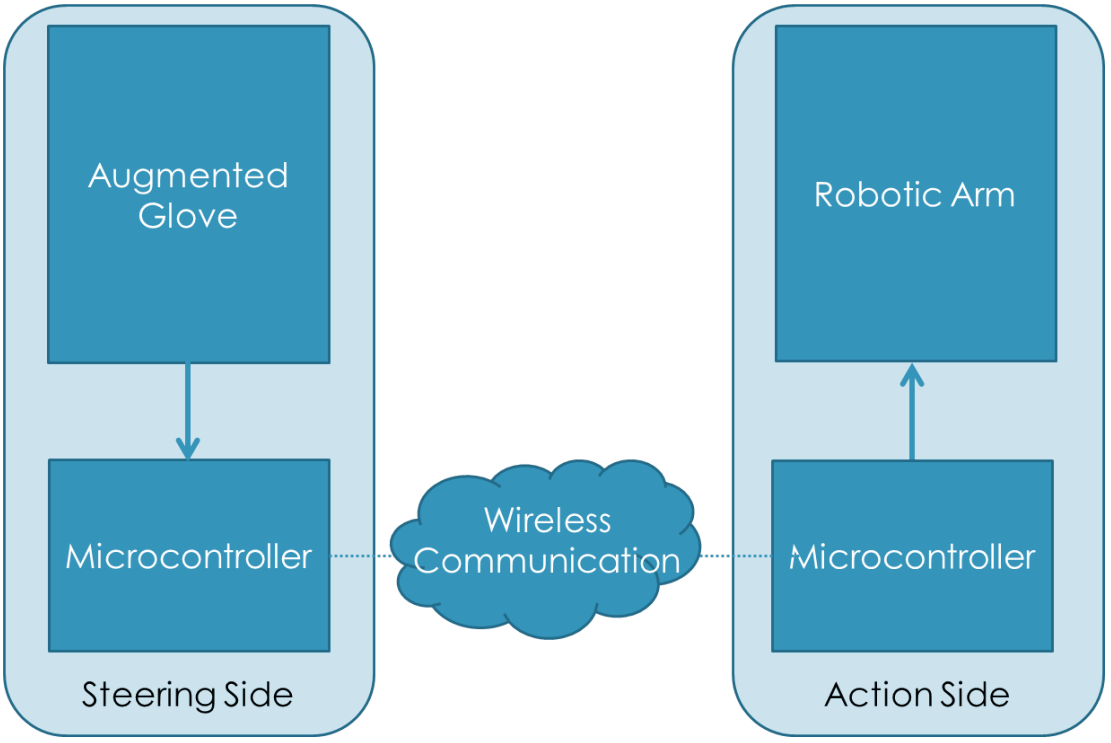


FIGURE 3.1: The architecture of the project

3.2 Steering Side

The steering side entails an augmented glove that is directly connected to a microcontroller. The microcontroller will record values that are detected from the sensors used in the project.

3.2.1 Augmented Steering Glove

The steering augmented glove has three components: a glove, flex sensors ,and an inertial measurement unit.

Although the glove is the most basic component of the whole project, some thought was put into it.It needs to be a stretchable and comfortable glove in a way that allows all sorts of people to use it especially that it will carry both sensors, IMU and flex, without damaging them or the glove. So, the glove must be sustainable where it should be easily

worn and removed without it getting torn up, and also it must be good enough to allow the sensors to detect accurate measurements through the glove. Below is a figure of the augmented glove that the team bought. Figure 3.2. This glove is made of both cotton and latex, which makes it sustainable, comfortable to wear ,and allows accurate measuring from the sensors.

FIGURE 3.2: Glove with pockets

In order to measure the flexing of the user's fingers flex sensors will be placed on each of the fingers, and the sensor would be connected directly to the microcontroller. as shown in the figure below 3.3

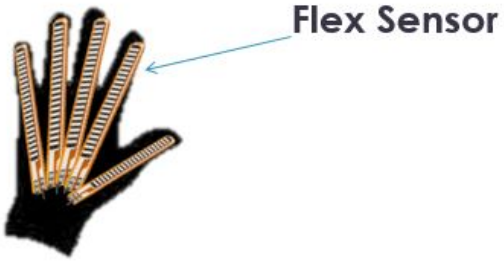


FIGURE 3.3: Glove Augmented with flex sensors.

As shown in the figure below 3.4 the resistance of the flex sensor would increase relative to the degree it is being bent. The increase of resistivity as the flex sensors are bent can be used for our advantage, and is the main reason they are used in this application

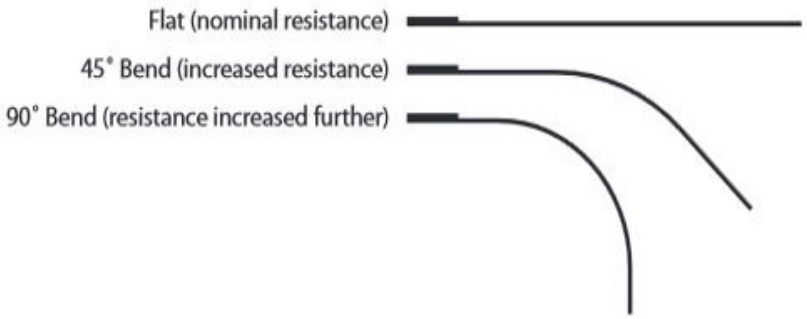


FIGURE 3.4: Flex sensor characteristics.

In order to provide data to the servo's moving the 2^{nd} and 3^{rd} degree of freedom, we will be using an Inertial Measurement Unit (IMU). The IMU is a combination of a gyroscope and accelerometer which offer precise and accurate depiction of movement to up to 6 degrees of freedom. IMU are constantly used in robotics.

3.2.2 Steering Side Microcontroller

The whole steering side will be controlled using a microcontroller. It will be responsible for reading the flex sensors and the IMU measurements, and transmit them wirelessly to the action side.

3.3 Action Side

The action side has the robotic arm directly connected to the microcontroller. The action side as mentioned previously will be steered by the augmented glove to perform exactly as the human arm controlling it. The robotic arm consists of the hand and the forearm. These two structures will be moved and controlled by motors called servo motors. The microcontroller controlling the servo motors that are used on the robotic arm is an Arduino Uno. Here the Arduino Uno is used because of its cost, computation power and the availability of wide range of pins. In the action side only digital pins are used and there are many of those pins. This is illustrated in Figure 3.1.

3.3.1 Robotic Arm

The robotic arm comprises a major part of the action side, and it consists of its structure and the servo motors that make it move. As stated in the problem formulation, the robotic arm is to operate with three degrees of freedom. Normally, the human arm has seven degrees of freedom, starting from the shoulder movements to the hand. For this project, the three degrees of freedom are the pivoting of the elbow joint, the rolling of the wrist, and the opening and closing of the hand (flexion and extension of the fingers). Figures 2.3, 2.7, and 2.8 show the three degrees of freedom that are to be implemented in this arm.

As mentioned in the proposal and midterm there are different approaches to constructing the robotic arm. The building of the hands and forearm using plastic tubes and cylindrical cans were achieved by the midterm assessment. The new structure shown in Figure 3.5 was printed using a 3D machine. Since the engineering department got a 3D printer the previous constraints of both time and budget were no longer an issue. The design of the printed structure was found online through an open source. Figure 3.6 shows the new robotic arm that is mark 3.

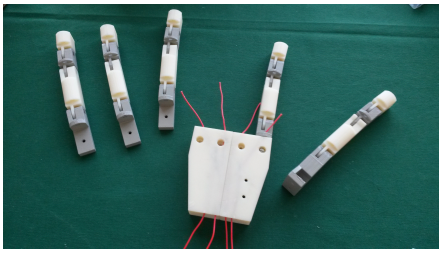


FIGURE 3.5: Material of the printed robotic arm

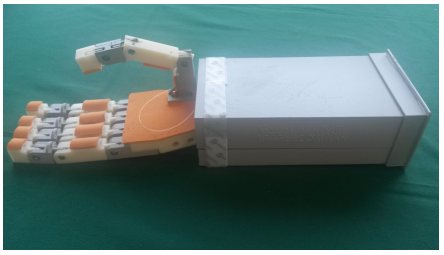


FIGURE 3.6: Printed robotic hand and forearm

Servo Motors

In order to make the robotic arm structure to move according to different degrees of freedom a set of servo motors are needed. For the first degree of freedom, a servo motor (shown in Figure 3.7) will be attached to each finger(a total of five). These will be called the finger flexion servo motors. For the second and third degrees of freedom, two more servo motors will be attached each to the wrist and the elbow joints. These will be called the rolling and pitching servo motors. The servo motors will be receiving information from the microcontroller to which the steering signals are sent from the steering side.

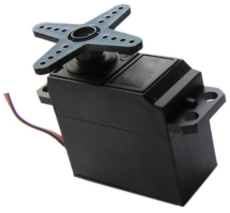


FIGURE 3.7: A Servo motor that will pull the strings of the fingers

3.3.2 Action Side Microcontroller

In the action side there is a control unit. This control unit receives information from the control unit in the action side to control the robotic arm. The microcontroller, Arduino Uno Figure 4.50, will process the information taken from the sensors and then control the servo motors respectively thus moving the robotic arm.

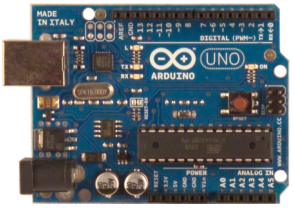


FIGURE 3.8: Arduino Uno

3.4 Wireless Communication

Given the nature of our design, three possible wireless communication mediums were considered:

TABLE 3.1: Comparison of different wireless technologies (based on [1] and [2])

	ZigBee	**Wi-Fi**	**Bluetooth**
Range	10-100 meters	50-100 meters	1 – 10 meters
Networking Topology	Ad-hoc, peer to peer, star, or mesh	Point to hub	Ad-hoc, very small networks
Operating Frequency	868 MHz, 900-928 MHz, & 2.4 GHz	2.4 and 5 GHz	2.4 GHz
Bandwidth	250 Kbps	Up to 54 Mbps	720 Kbps
Power Consumption	Very low (low power is a design goal)	High	Medium
Security	128 AES plus application layer security	WEP, WPA, WPA 2	64 and 128 bit encryption

We need a form of wireless communication that have a high range, and can work in any environment, these criteria excludes Bluetooth from the choices because it does not satisfy the range requirement. Moreover, Wi-Fi was also excluded due to the fact that Wi-Fi does not work in all environments, so we will are left with ZigBee.

ZigBee uses IEEE protocol 802.15.4 which reserves 3 radio frequencies [(868 MHz (Europe)900-928 MHz (NA), 2.4 GHz (worldwide)], we will be using the 2.4 GHz frequency. Moreover, the brand we decided to get is Xbee which is a multiplatform transmitter and receiver that operates on ZigBee technology.

3.5 Big Picture

Our project will have an augmented glove fitted with both flex and IMU sensors that is then directly connected to a microcontroller. This microcontroller sends data wirelessly to the second microcontroller entailing details of the flex and IMU sensors. The second microcontroller takes this data and moves the servo motor accordingly. The servo motors are connected to the arm thus moving the arm in the same manner. This is what the team will achieve at the end of the semester. Figure 3.9. The built prototype achieved by the team is shown in Figure 3.10 shows

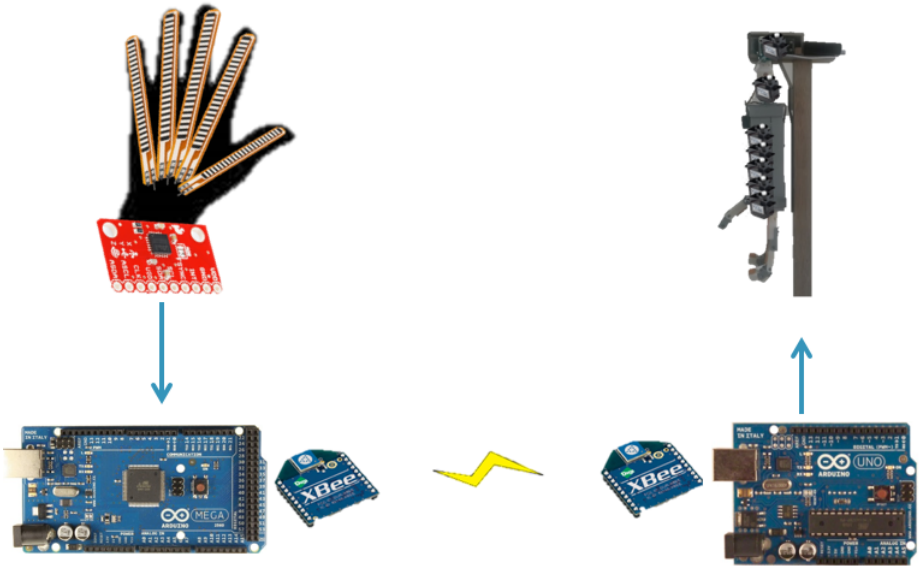


FIGURE 3.9: Big Picture

FIGURE 3.10: Current built prototype

Chapter 4

Implementation

The upcoming sections dig deeper into each part of the high level architecture that was stated in the proposed solution. First, the implementation details of the steering side will be explained, followed by those of the action side. Then there will be discussions regarding the microcontrollers and the reasons behind the choice of each microcontroller. Finally, the section of wireless communication will be discussed, this stage is new in regards to the project implementation since it is to be achieved after the midterm assessment.

4.1 Augmented Steering Glove

The augmented steering glove is a very important part of the project, it controls the whole robotic arm structure. The control is done through a combination of sensors that the glove is augmented with; the sensors include: Flex Sensors, and Inertial measurement unit or IMU. As mentioned in the section, the flex will be used to detect movement of the fingers, and the IMU will be used to detect the rolling and pitching movement.

4.1.1 Flex Sensors

Flex sensors are analog resistors that change in resistance depending on the amount of bend or flex on that particular sensor. They are made of carbon resistive elements that are placed within a flexible substrate; when this substrate is bent, the sensor changes in resistance, depending on the bend radius, As shown in Figure 4.1. Given the following characteristics, the flex sensor is the best option for capturing the flexing motion of the fingers, as illustrated in Figure 2.3

FIGURE 4.1: Characteristics of Flex Sensors

In order to start working with the flex sensors, their operating range must be measured, as illustrated in Figure 4.2. The operating range includes the least resistance or least bend and also the maximum resistance or full bend.

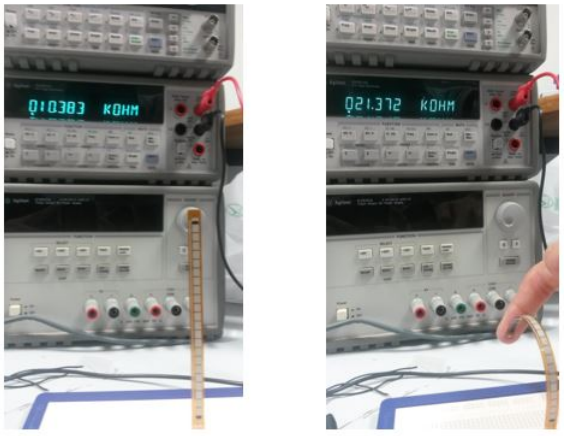
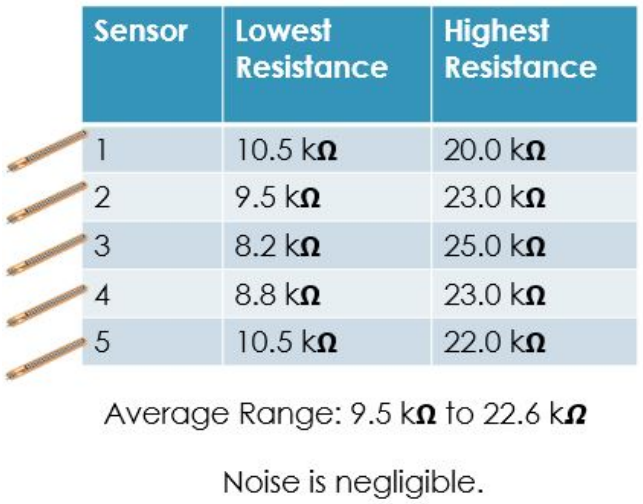


FIGURE 4.2: Measuring Least and Maximum resistance.

The maximum and minimum resistance of each of the flex sensors is measured and recorded, as shown in the Figure 4.3. It is directly apparent that the resistances varies among the sensors. However, it was observed that when each of the flex sensors are flexed the sensors did not fluctuate in resistance. This proves beneficial since that will not cause any fluctuation in the movement of the robotic hand. Since there are no fluctuation measured in regards to the flexing there will be no need for an external noise filtering circuit. In order to capture the true values - flexing of the fingers precisely - the flex sensors should be mounted on the augmented glove. It is observable in Figure 4.4 that when the flex sensors are fitted into the glove, they are always at a certain bend. So, before

connecting it to the microcontroller, all the flex sensors were fitted on the glove and then were normalized . Normalization of the flex sensors will be discussed later.



Sensor	Lowest Resistance	Highest Resistance
1	10.5 kΩ	20.0 kΩ
2	9.5 kΩ	23.0 kΩ
3	8.2 kΩ	25.0 kΩ
4	8.8 kΩ	23.0 kΩ
5	10.5 kΩ	22.0 kΩ

Average Range: 9.5 kΩ to 22.6 kΩ

Noise is negligible.

FIGURE 4.3: Measurement of Flex sensors

FIGURE 4.4: Characteristics of Flex Sensors

In order to connect the flex sensors to the microcontroller, a simple voltage division circuit is use. The Figure 4.5, shows the connection of the flex to the microcontroller; However, it is worth noting that the flex sensor can be connected directly to the analog pins of the microcontroller, but the voltage division circuit is used to reduce the voltage entering the analog pins, because the analog pins on an arduino microcontroller can take a maximum

of 5 volts. Moreover, the voltage division circuit acts an extra layer of safety if the flex sensor would get damaged. In other words, there would not be a complete short circuit, because the 10K ohm resistor would always be present. After the connections to the microcontroller were made, a digitized value for each of the flex sensors can now be read. It is worth mentioning that the when the microcontroller reads the analog values at each of the inputs, the value is directly mapped to 0 to 1024.

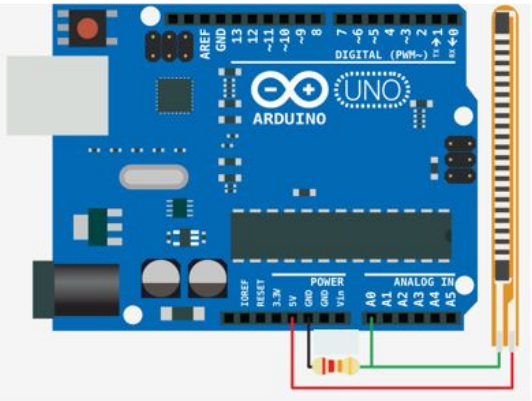


FIGURE 4.5: Flex Sensors connections to the microcontroller

From Figure 4.6 it can be seen that the values of both the lowest and highest readings are different from one flex sensor to the other. The fluctuations in the readings do not cause any problems because each of the sensors are mapped from 0 to 170 degrees. This mapping is crucial since it represents the degrees of movement of the servo motors. When the mapping was first implemented it was observed that it accomplished its purpose. However due to the nature of the mapping, it does not limit the values to only 0 to 170. For example if we take sensor 1, which has a lowest reading of 575, and the highest reading of 690. If the sensor is outputting a value of 570 for example, then the microcontroller will map this reading to a negative angle to the servo, which might cause errors and damage to the servos. In order to solve this problem, normalization was implemented. In essence all what was added was a few lines of code that would force the values to stay within 0 and 170 as shown in Figure 4.7, the full flex sensor code is found in Appendix A.

Sensor	Lowest Reading	Highest Reading	Lowest Mapping (Degrees)	Highest Mapping (Degrees)
1	575	690	0	170
2	575	690	0	170
3	545	655	0	170
4	575	650	0	170
5	610	665	0	170

FIGURE 4.6: Values of Flex Mapped from arduino, then mapped to degrees

Sensor	Lowest Reading	Highest Reading	Lowest Mapping (Degrees)	Highest Mapping (Degrees)
1	575	690	0	170

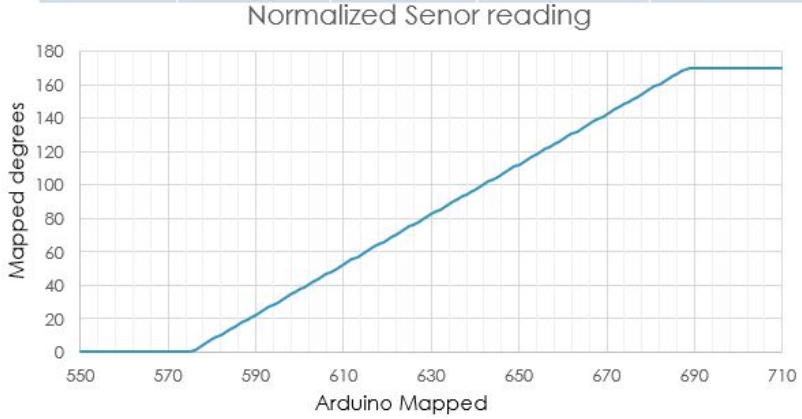


FIGURE 4.7: Normalization of the flex sensor values

The Flex code is found in Appendix A. The code runs as follows: First the servo library is included, and the serial communication is setup, and then the servos are attached to a specific pin on the microcontroller Arduino Mega. The program enters through an infinite loop where the microcontroller keeps reading the analog values from the flex sensor. After the reading is done the values of the flex sensors are normalized, and then mapped from 0 to 170 degrees in order to be sent to the servo motors. Finally the program will go through a delay before running the loop again. The reason a delay is used, is to allow

time for the servo motors to reach the specified value. Although the flex sensors best fit our application, another alternative was consider; the alternative is discussed later.

In every project there are alternative device that can be used in the project; an alternative of the flex sensors is the Xbox kinect sensors. Although the flex sensor looked the most suitable option for this implementation, one alternative that was considered is implementing the flexing of the fingers using Image processing and a kinect sensor. However, the kinect sensor is not feasible due to the reasons stated in the Table 4.1. The main reason the kinect sensor was not used is because the scope of this project is to have a glove that can be worn by the user, then the user would be able to use this glove to control the robotic arm; given that this is not feasible using the Kinect, it was removed as an option. Flex sensor remained the best choice to use, due to its usability and easy maintenance.

TABLE 4.1: Comparison between flex sensors and Xbox Kinect

Implementation	Flex Sensor	Xbox Kinect
Ability to mimic flexing of the fingers	Yes	Yes
Ability to be placed around the hand	Yes	No
Ability to be integrated with Arduino	Yes	No
Amount of Devices needed	5	1

The flex sensors are relatively easy to understand, use. and maintain, so not many problem were faced. However, due to its fragile structure (sustainability), some extra flex sensors were ordered, just in case one of them broke. However, none of the flex sensor broke. The other slight problem that was faced, was the problem with mapping, before normalizing. However, with normalization there are no longer negative values that are being sent to the servo motor

4.1.2 Inertial Measurement Unit (IMU)

In order to provide data to the servo's moving the 2^{nd} and 3^{rd} degree of freedom, an Inertial Measurement Unit (IMU) will be used. The IMU is a combination of a gyroscope and an accelerometer which offer precise and accurate depiction of movement to up to 6 degrees of freedom. IMU are constantly used in robotics. Figure 4.8 and Figure 4.9 are illustrations of the pitching and rolling of the arm. A more detailed introduction on the IMU can be found in Appendix E.

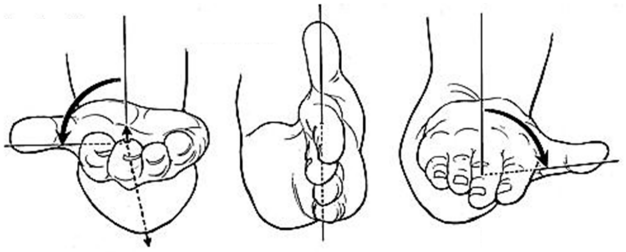


FIGURE 4.8: Rolling of the arm

FIGURE 4.9: Pivoting of the arm

When we started working with the IMU we first connected it to the Arduino and ran some sample test to understand and check its functionality. After doing so and with the aid of present libraries we were able to develop a code that works with the project perfectly. As a reminder the IMU will be used to move the second (rolling of the arm) and third (pitching of the arm) degrees of the robotic arm.

The libraries that were included in the code are the I^2C and the DMP or digital motion processing algorithms that are programmed on the IMU chip. The DMP focuses on three angles called the Euler's angles. The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body. The three angles are the psi, phi and theta, and these angles represent three degrees of freedom of which two will be used for the project. The degrees of movements are the yaw of the arm, pitch of the arm, and roll of the arm.

In order to understand the IMU, the first step is understanding the code. First the MPU 6050 libraries were included. The setup is where specifying baud rate, initializing the MPU, and testing the functionality of the MPU happens. Next, the program moves on to the main loop where it checks for either interrupts or FIFO over count. If there is an

interrupt and no FIFO overflow then the program will continue to read values from the IMU until the queue is filled. Once the queue is filled the code will break and reinitialize again. The program actually reads from the IMU twice in order to avoid having the robotic arm lose functionality in the middle of its operation. The values that are read from the IMU are first normalized and then mapped to degrees in order for them to be read by the servo motors. The code that is used for the IMU sensors is found in Appendix B.

After developing the code, the team worked with IMU (moving it around) and recorded the maximum and minimum values for each of the degrees. The rolling of the hand moved from -20 to +10 rads. While the pivoting of the arm moved from -70 to +70 rads. These values recorded were of importance because later they were to be used to be mapped to degrees from 0 to 170 degrees so that the servo motors would move accordingly. Another usage of knowing the maximum and minimum values was for normalizing the values taken. For the rolling, values below – 20 rads are to be omitted meaning they are all mapped to zero degrees and all values above +10 rads are all mapped to 135 degrees. Figure 4.10 contains a graph showing the normalization of the rolling of the arm. The same thing was also done for the pitching of the arm, but for the pitching the values were mapped from 0 to 170 degrees. The graph in Figure 4.11 shows the normalization of the rolling of the arm.

The next step was connecting the IMU to the pitching and rolling servo motors and observing the results and communication between the servo motors and the microcontroller.

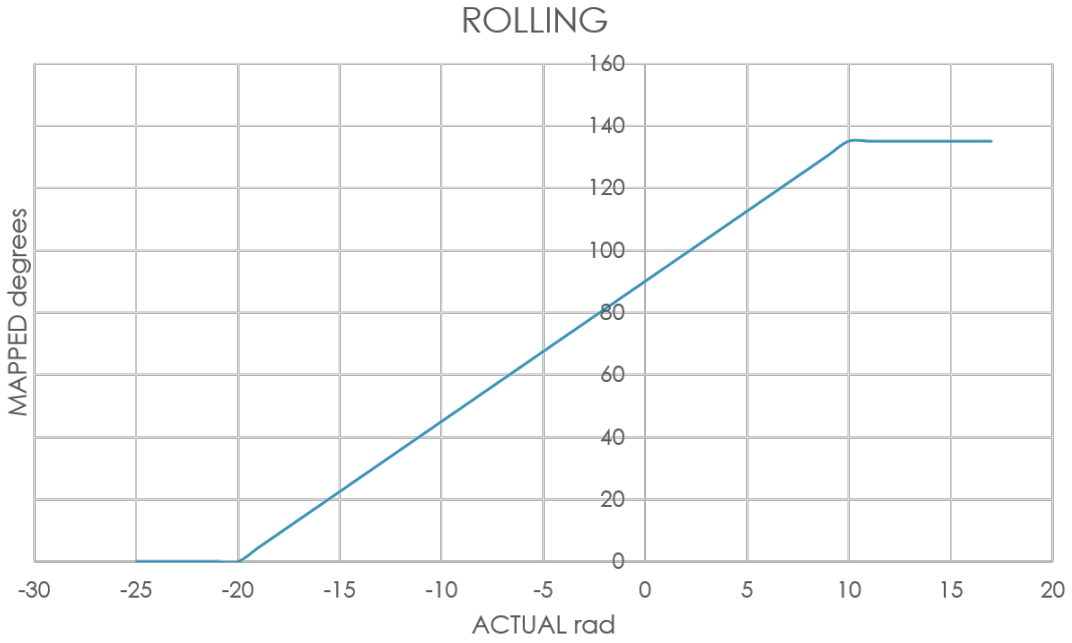


FIGURE 4.10: A graph showing the normalization and mapping to degrees of the rolling values.

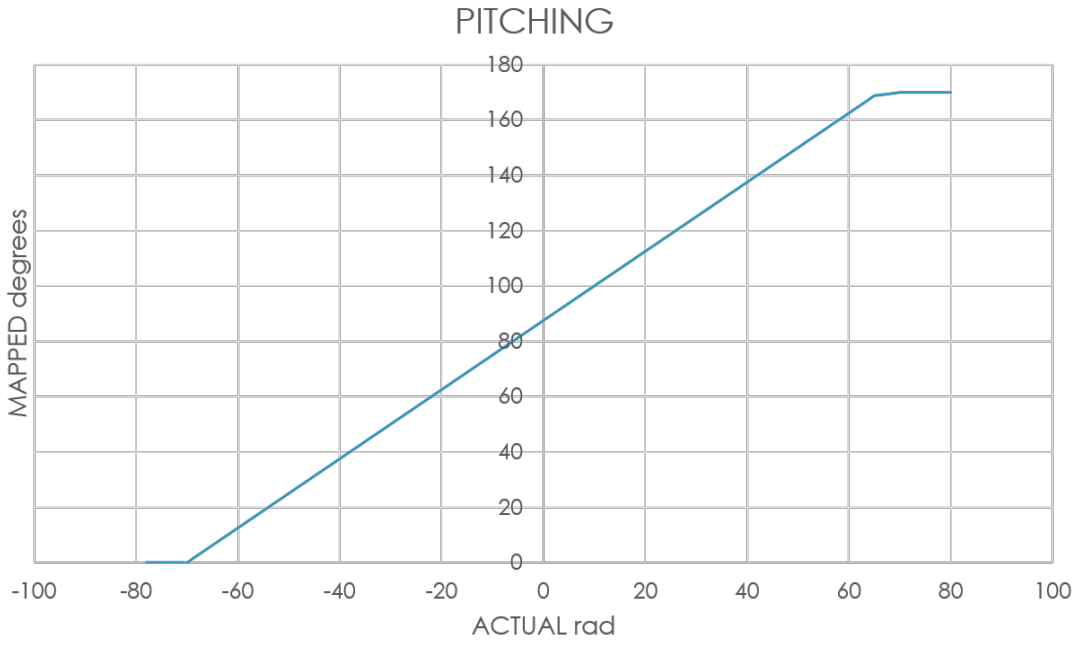


FIGURE 4.11: A graph showing the normalization and mapping to degrees of the pitching values.

In every project there are alternative device that can be used in the project; an alternative of the IMU is the Triple-Axis Digital-Output Gyro ITG-3200 Breakout. The Triple-Axis Digital-Output Gyro ITG-3200 Breakout is a breakout board for InvenSense's ITG-3200, a groundbreaking triple-axis, digital output MEMS gyroscope. The ITG-3200 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyro outputs, a user-selectable internal low-pass filter bandwidth, and a Fast-Mode I2C (400kHz) interface. Additional features include an embedded temperature sensor and a 2 percent accurate internal oscillator.

The ITG-3200 can be driven at any voltage between 2.1 and 3.6V. For power supply flexibility, the ITG-3200 has a detached VLOGIC reference pin (labeled VIO). Also it has an analog supply pin (VDD) which arrays the logic levels of its serial interface. The VLOGIC voltage may be any voltage from 1.71V min to VDD max. For common use, VLOGIC can be tied to VCC. The normal working current of the sensor is just 6.5mA.

Communication with the ITG-3200 is accomplished over a two-wire (I2C) interface. The device also has an interrupt output, and a voluntary clock input. A jumper on the top of the board permits the user to simply select the I2C address, by pulling the AD0 pin to either VCC or GND. If the user does not plan on using the CLKIN pin, they can easily short the jumper on the bottom of the board and connect it to GND.

The main difference between this breakout board and the breakout board that the project has is its cost. The ITG 3200 is worth 24.95 while the MPU 6050 is worth 39.95 and it comes with an accelerometer and an encoded DMP algorithms. So having the IMU which includes both a gyro and an accelerometer is much cheaper and takes less space and connections is a much better choice. Another reason is that when dealing with movement of robotic structure the IMU is a much better choice.

The IMU proved difficult while operating at the beginning due to a number of different factors. These factors included the fact that the datasheet was not helpful with regards to the IMU connections. There were errors regarding the SDA and SCL connections; in the datasheet the SDA and SCL can be connected to the digital pins of the Arduino. After sifting through Arduino Forums, it was observed that the actual pins in use for IMU were the analog pins. Another problem faced was the sample code that was used. The code was not functional so it either gave results as zeroes or garbage. The solution to that problem

was then found on the Arduino forums. Next, the development of the code. Since some libraries were included, the team had to understand the working of these libraries before moving on with the development of the code. After researching and understanding the code of the libraries the team moved forward with the writing of the code. The final problem the team faced was that the IMU needed connectors in order to connect the IMU to the Arduino. The IMU was a QFN (quad flat no leads package), so the problem was that before finding any connectors the team had to connect the wires directly to the IMU. The result was that the wires sometime touched one another causing the IMU to stop working and the team had to always restart the operation from beginning once again. This was not functional because, firstly the IMU might break and stop working all together and secondly having the program not working in the middle of its operation is not efficient. The problem there was the difficulty of finding the connectors, once the connectors was found and soldered to the IMU the problem then was resolved.

4.2 Robotic Hand (Mark 2)

For the robotic arm to act the same way the human arm acts, both its structure and functionality needs to be designed to meet those of the human arm. With the biological working mechanism of the human arm in mind, the robotic arm was built from the fingers to the forearm. Basic materials, such as electric tubes, cylindrical potato chips containers, and fishing lines, were used. The reason is that their physical structure is suitable enough to serve as a proof of concept of how the structure of the robotic arm can be. For example, the fact that electric tubes can retain their original relaxed position after being bent made them a good choice for the robotic fingers that need to be flexed and relaxed. In a similar fashion, the use of fishing lines as the tendons to be attached to the finger tips at one end was chosen, since pulling them at the other end would achieve the flexion of the fingers (as shown in Figure 4.22). Eventually, The pulling of strings, rolling, and pitching of the forearm will be done by actuators, which are servo motors integrated with the structure to provide the functionality. The subsequent sections dig deep into the implementation of the robotic arm structure and functionality.

4.2.1 Robotic Hand Structure

After a basic version of the hand was built in the proposal stage, a newer version was built to increase the hand's functionality. For that to happen, improvements were done on both the material used to construct the hand and the method of building it. Figure 4.12 shows the two versions of the robotic hands that were built, named *Mark 1* and *Mark 2*,

(A) Mark 1 (B) Mark 2

FIGURE 4.12: Mark 2 was made as an improved version of Mark 1

Moving from Mark 1 to Mark 2 There are many differences between the two versions, aside from Mark 2 obviously looking more realistic. In order to build to a newer version of the hand (Mark 2), there should be justifiable reasons regarding the lack of robustness and functionality of the previous version (Mark 1). First, Mark 1 was made of very thin plastic electric tubes that were split-through, which made the structure not capable of carrying objects properly. Hence, Mark 2 was made of non-split-through, stronger plastic tubes. Figure 4.13 shows the type of tubes used for each Mark.

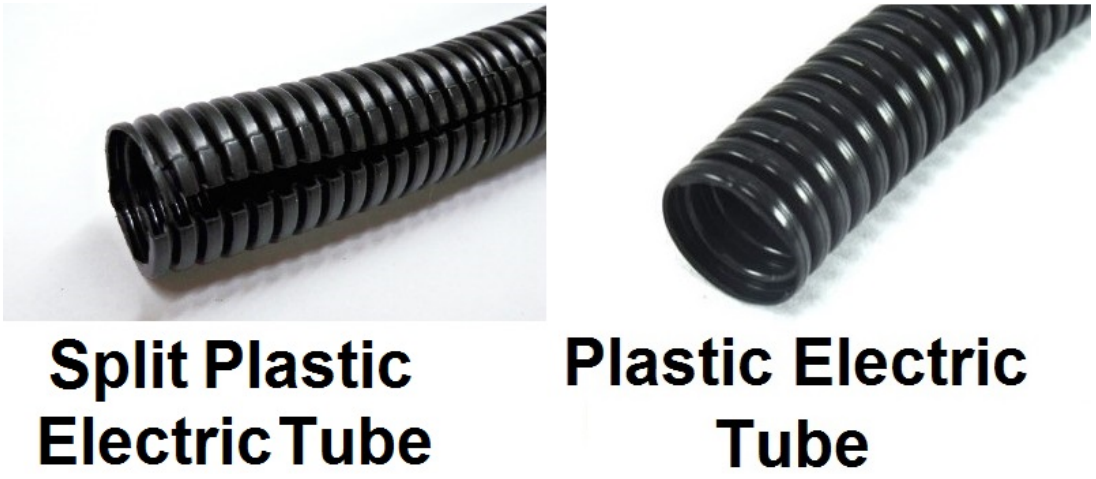


FIGURE 4.13: Split-through plastic electrical tubes (used in Mark 1) and closed tubes (used in Mark 2)

The second choice of thicker plastic tubes provides a stronger structure which makes the hand better capable of holding objects, as well as faster in returning to the original relaxed position. The second reason to switch to Mark 2 is that the flexion of the fingers in Mark 1 is far away from natural, as shown in Figure 4.14. Although both Marks do flex in a circular fashion, Mark 2 flexion is closer to the realistic flexion. This is mainly due to the cutting of the joints, which is explained later.

FIGURE 4.14: Flexion of Mark 1 fingers is circular, which is a limitation in the design.

More obvious than the drawback of the flexion of the four of the pointer, the middle, the ring, and the pinky fingers, the thumb of Mark 1 does not flex in a natural position. Not only does this drawback make the design of Mark 1 unrealistic, but it also strongly affects

the functionality of the hand. Figure 4.15 shows the top view of both hands with the thumb flexed.

(A) Mark 1 Thumb Fexion (B) Mark 2 Thumb Fexion

FIGURE 4.15: Thumb flexion of Mark 1 and Mark 2

When the thumb joint is flexed, its flexing mechanism enables it to wrap around the object to carry it. This is shown in Figure 4.16, with the thumb of Mark 2 flexing in a third dimension, rather than the limited two-dimensional flex of the thumb of Mark 1 (Figure 4.15a).

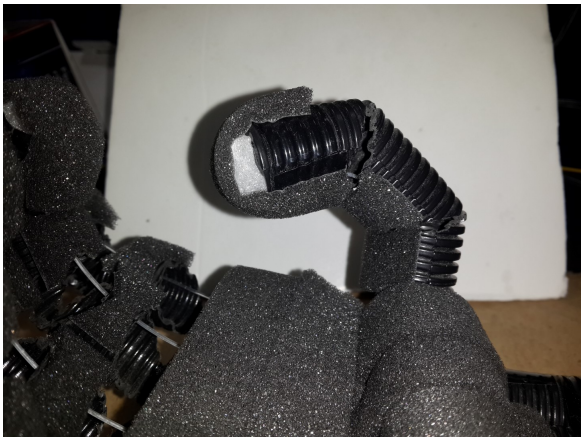


FIGURE 4.16: Flexion of Mark 2 thumb adds more functionality in addition to looking more natural.

Besides having the Mark 1 fingers not flexing properly to the maximum wanted flexion, they do not flex straightly. Figure 4.17 illustrates how the finger in mark 1 does not flex straightly, since it is mostly on one side of the red axis rather than being bisected by it.

FIGURE 4.17: Flexion of Mark 1 fingers is not straight. The red line represents an axis that ideally should have the mirror image of the flexed finger around it, but is not the case in Mark 1.

On the other hand, the flexion of the fingers in Mark 2 is closer to the natural straight flexion. Figure 4.18 illustrates this natural flexion.

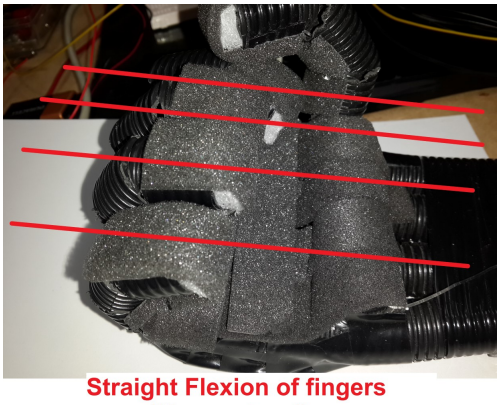


FIGURE 4.18: The red axis is shown to verify that the flexion of Mark 2 is closer to straight, hence more natural.

Designing the Joint Cuts for Mark 2 Given the previous problems, the cutting of the tubes of Mark 1 to implement the robotic finger flexion at the joints had to be improved. This is essentially what caused the several drawbacks of Mark 1, and what made the switch to Mark 2 necessary.

Basically, to enable the plastic electrical tubes to be flexing, they should be cut-through at several places at which the tubes will be bent, thus mimicking the flexion of the fingers. This is illustrated in Figure 4.19.

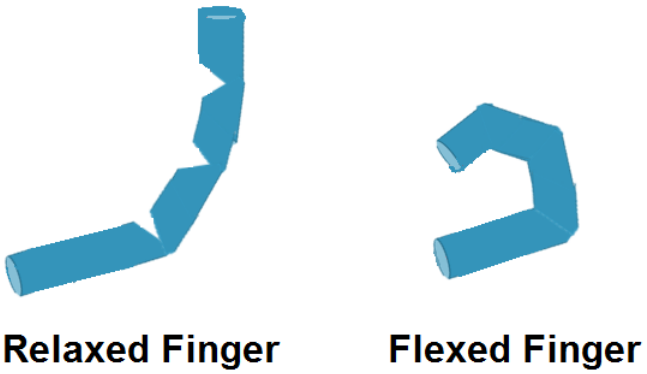


FIGURE 4.19: Robotic finger flexion position

Those joint cuts were implemented differently in Mark 2. By testing different alternatives (some of which are shown in Figure 4.20), the final joint cut was chosen to be the a v-shaped cut, such as the one shown in Figure 4.21.

FIGURE 4.20: Different joint cuts were implemented to reach the decision on the best.

FIGURE 4.21: The best alternative for joint cuts was found to be a v-shaped cut.

Essentially, Mark 2 was not to be built unless there were serious limitations with Mark 1. The large differences between the two marks (summarized in Table 4.2) are reasonable since the time and effort spent to design Mark 2 was much more than that of Mark 1. The first Mark was built in a short time to serve as a proof of concept in the proposal stage of the project.

Since the robotic fingers were built from plastic tubes, which are static material, there should be a way to make those fingers flex. Similar to flexing mechanism of the human

TABLE 4.2: Summary of Differences between Mark 2 and Mark 1

	Mark 1	**Mark 2**
Finger Material	Plastic Tubes	Plastic Tubes
Material Strength	Weak	Strong
Split-through	Yes	No
Joint Cut Style	Rectangular cut	V-shaped cut
Finger flexion	Less natural	More natural
String Material	Fabric Strings	Fishing Lines
Grip Material	Electrical tape	Foam tape

fingers, which are attached to tendons at different parts of the finger, the tips of the robotic fingers are attached to strings that act as the tendons. When the tendons are pulled, the finger flexes. Section 4.2.2 contains details about the strings that are tied to the robotic fingers to make the fingers flex. On the other end, the strings are attached to servo motors, which will rotate and pull the strings to make the robotic fingers flex (as explained in Section 4.2.2.2).

4.2.2 Finger Flexion

As mentioned previously, the robotic finger flexion mechanism is going to be implemented with the help of the strings that will be pulled by servo motors. With Mark 2 design and implementation of the robotic hand, the extension of is not implemented. That is, the fingers can only flex with when the strings are being pulled, and then they are only relaxed when the strings are released and the structure of the plastic tubes makes them go back to their resting position, as mentioned earlier. The extension is only possible with different designs, however, they are out of the scope.

4.2.2.1 Strings

In order to pull the finger tips and make the finger to flex in an efficient manner (i.e. fast, complete, and as strong as possible), different choices of the strings had to be tried and studied. For that to happen, several parameters were kept in mind, including the tensile strength of the string and its friction. Three types of strings that were tested were

fabric strings, fishing lines (fishing strings), and thin electrical wires. Figure 4.22 shows the operation of the strings.

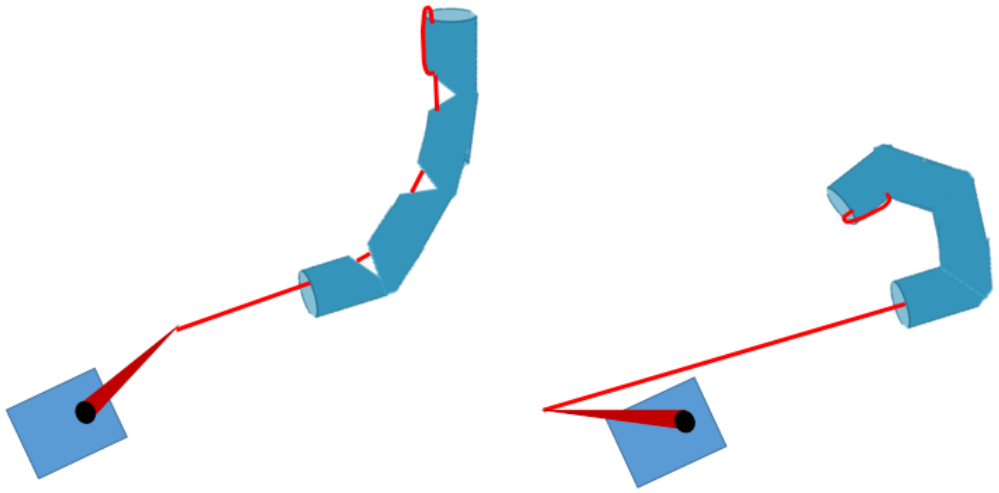


FIGURE 4.22: Implementation of finger flexing using strings

Fabric Strings The first type that we used was the thread wire, the thread wire was our first choice while using it as a prototype. We found out some advantages and disadvantages when using the threads as our strings. The advantage was that the pulling of the finger was good where the threads did not get tangled up inside the fingers thus allowing full flexing while using less radius on the motor. There are many cons to the fabric strings and this can be seen through Figure 4.23. The cons of the thread string was that of its material where the threads are degradable and tying it to the servo motor becomes difficult. Also another reason for not going with the threads is that they easily get twisted on the servo motor which then reduces the life efficiency of the motor and the material of the hand. Lastly the threads can easily get mixed-up over each other when they are all packed together.

FIGURE 4.23: Fabric String

Thin Wire Strings After working with the threads and understood its pros and cons we went with a much smaller string, a thin copper wire because of its composition and its tension. There are also some advantages and disadvantages to the use of the thin copper wire. The first advantages of the copper wire was that it could be tied easily to the servo motor. The wire also had a good tension where it can easily bend the robotic fingers. Also an advantage of the wire over the threads was that the plastic coating of the wire was not as degradable as the material of the threads. Lastly the copper wire did not strain the material of the fingers. The first disadvantage was that when the wire is connected to the servo motor, the wire does not pull the fingers in the flexing fashion that is desired. The wire bypasses the rotation of the servo motor where the wire is not pulled by the motor and thus not pulling the fingers. The second disadvantage was that the wire bends inside the fingers as if it is crawling inside the fingers. This causes the servo motor to pull, if the wire rotated with the motor, the bend wire thus straightening it first before actual pulling of the fingers could happen. Figure 4.24 identifies the thin wire strings.

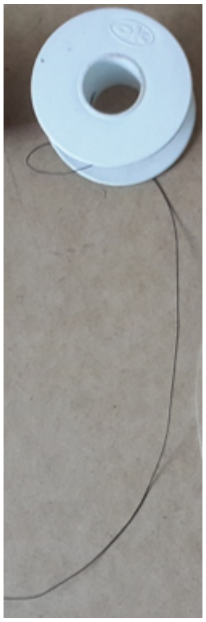


FIGURE 4.24: Thin Copper Wire String

Fishing Wire Strings Lastly is the fishing wire string. Upon working with this material we realized that this one was the best of the three used in the previous prototype. The fishing wire has a lot more advantages with regards to the other two types of strings. The fishing wire is made from plastic and unlike the thread strings it is not degradable. It fits perfectly on the servo motor and can easily pull the fingers. It also does not damage the material of the robotic hand. The fishing wire does not bend or curl up inside the finger thus not needing extra rotation to pull the finger. The fishing wire rotates in the same direction of the servo motor rotation, and it does not bypass or go over the servo motor. Lastly the fishing wire has high tension so it needs low torque to pull the fingers of the robotic hand. The disadvantage is that it cannot be tied nicely to the fingers and servo motor. Figure 4.25 shows the type of fishing wire strings used.

FIGURE 4.25: Fishing Wire String

Table 4.3 summarizes the above characteristics of every string shown in Figure 4.26.

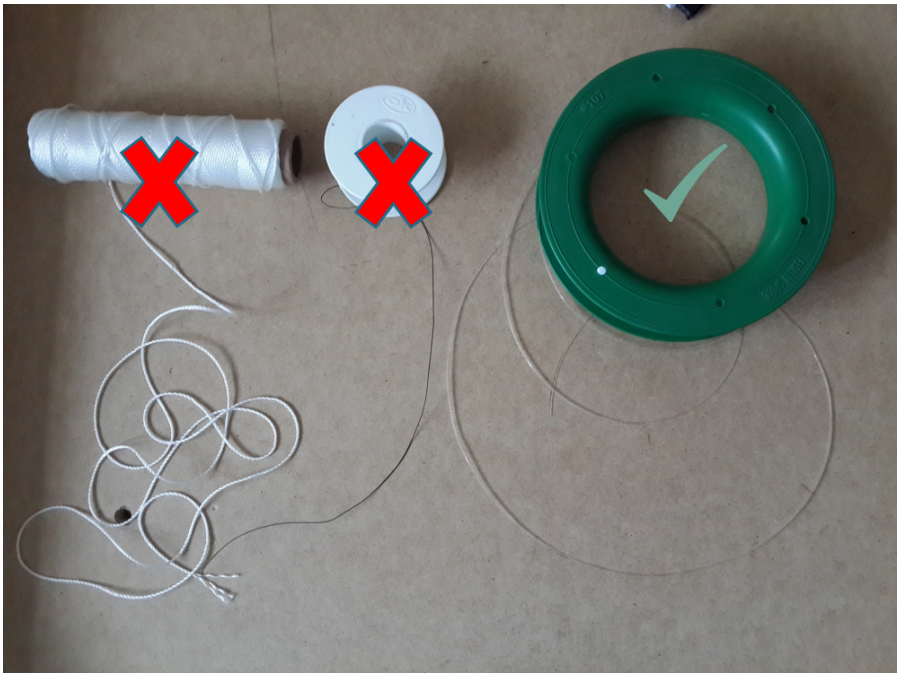


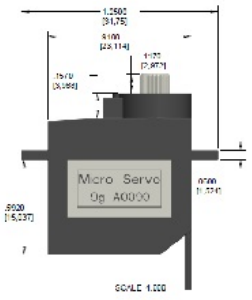
FIGURE 4.26: All Strings

TABLE 4.3: Comparison of different strings characteristics

	Fabric	**Copper Wires**	**Fishing Lines**
Tensile Strength	Low	Medium	High
Stiffness	Medium	High	Medium
Friction	High	Medium	Low

4.2.2.2 Lower-torque Servo Motors

A servo motor is an actuator that allows precise control of angular position, velocity and acceleration. In other words it is no different than a dc motor, with the exception that is also has an on board sensor to help direct current and voltage to achieve an accurate motion. Servo motors are used in applications such as robotics, and automated manufacturing. In order to achiever the flexing, that was discussed in 4.2 the low torque servo motor was used. Figure 4.27 shows the specification of the low torque servo motor. It is worth mentioning that the torque output is more than enough. However, due to the size of the of the finger, the servo was not able to achieve its full flex. The maximumn flex of the current configuration is shown in Figure 4.28. Given this problem, the only possible solution was to work with the servo motor itself.



Features
Voltage: 4.8-6.0 Volts
Torque: 16.6/20.8 oz-in. (4.8/6.0V)
Speed: 0.15/0.10 sec/60° (4.8/6.0V)
Rotation: ~160°
Single Top Ball Bearing
Nylon Gears
3-Pole Ferrite Motor

FIGURE 4.27: Characteristics of Low Torque Servo motor

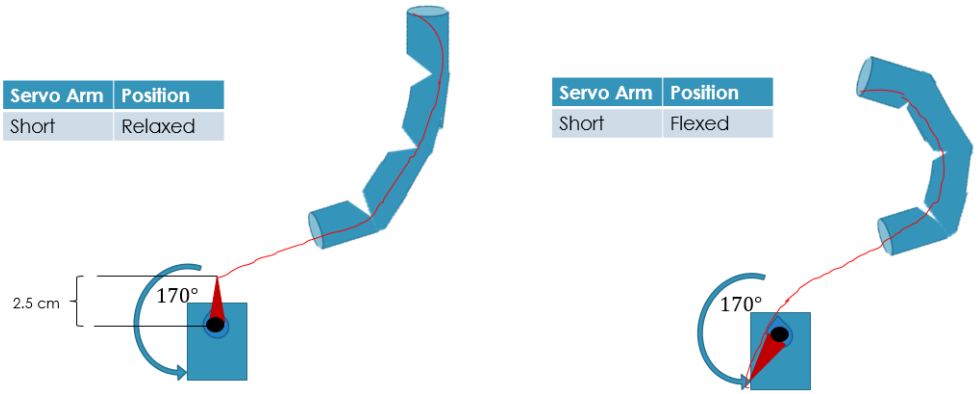


FIGURE 4.28: Maximum Flex with current configuration of servo arm

In order to maximize the flex of the fingers, there was shift of focus from the servo motor to the servo motor arm. The premise was simple, as illustrated in Figure 4.29, in order to increase the flex with the current servo motor, the servo motor arm was extended.In other words, to increase the flexing distance, the radius of rotation (servo arm) is increased. The extension of the servo arm allowed for maximum flexing of the fingers, at maximum angle of rotation, as show in Figure 4.29

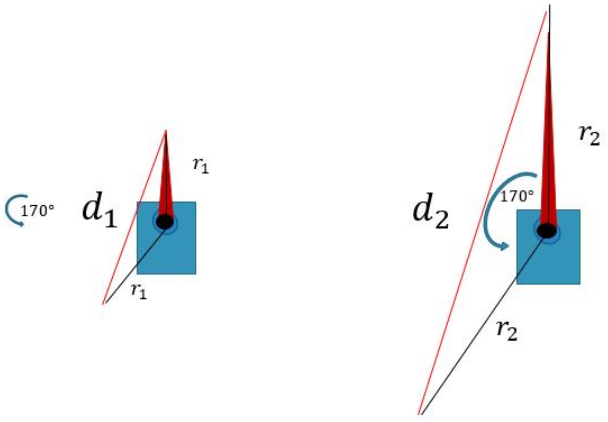


FIGURE 4.29: Relation of Servo arm to Flexing distance

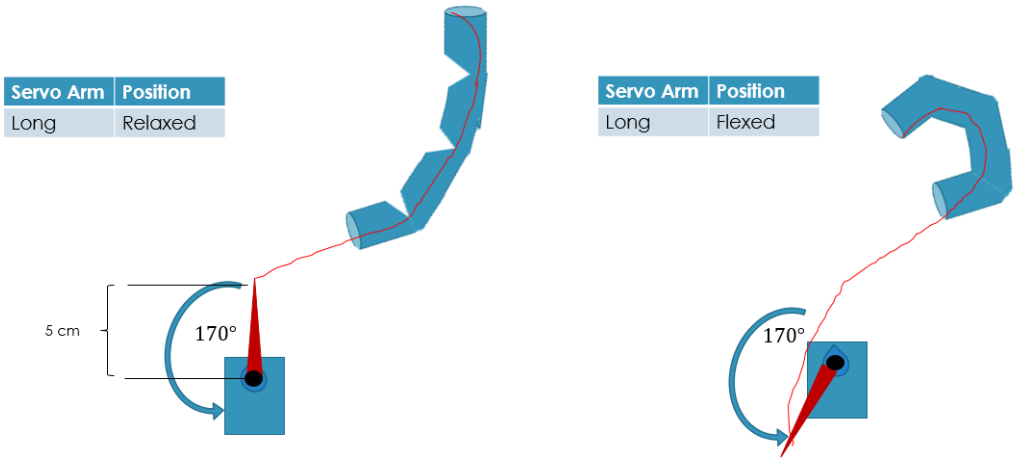


FIGURE 4.30: Maximum Flex of finger achieved when servo arm is increased

To have room for the lower-torque servo motors, they will be put in a protected structure that provides room for their full rotation, as well as makes them optimize the pull of the strings to achieve the maximum finger flexion possible. This structure will be sheltered in the robotic forearm, whose implementation is explained in Section 4.3.

4.3 Robotic Forearm (Mark 2)

In order to have a structure that holds the lower-torque servo motors, the forearm should be designed in a way to provide room for the five servo motors, as well as to account for the rotating arm of each servo motor. Thus, the factors (or design constraints) that should be taken into account when designing the positioning of the servo motors are the following:

- There should be room provided for both the five lower-torque servo motors and their rotating arms (with space allowing the maximum rotation).
- The servo motors arms should be positioned in a way such that they optimize the pull of the strings. That is, each degree of the servo arm rotation actually makes the arm pull the string.
- Neither the rotating servo motor arms nor the strings that they are attached to should get tangled with the others servo motor arms or strings.

- The servo motors should be sustainably fixed in position since they are continuously being stressed when they are functional (i.e pulling the strings).

Keeping the above design constraints, different design alternatives were used to make the servo motors positioned in a proper way to achieve optimized finger flexion.The subsequent section describes the different design alternatives used for the servo motor positioning on a robust base structure.

Positioning The Servos on a Base Structure The alternative designs used to position the servos on a base structure were designed in a way to ensure the servos are apart from each other, yet compact and able to have the servo arms rotating to all their degrees. Figures 4.31 through 4.33 show the different design alternatives of the positioning structure of the lower-torque servo motors. In the first version (Figure 4.31), two servos on the front (shown at the top) were positioned from the other side of the base structure because their arm rotation is the opposite of the others. This was done as an attempt to avoid the strings being tangled with each other. In addition to that, some structures were added in order to regulate the string path, also in attempts to make a distinct path for each string.

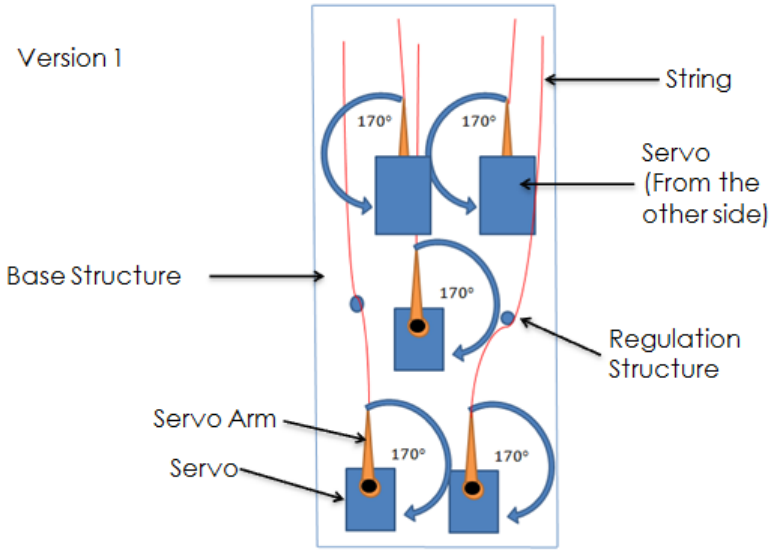


FIGURE 4.31: The first design alternative of the servo motor positioning

The second version (Figure 4.32) is similar to the first, except for having one servo motor using the regulation structure instead of two.

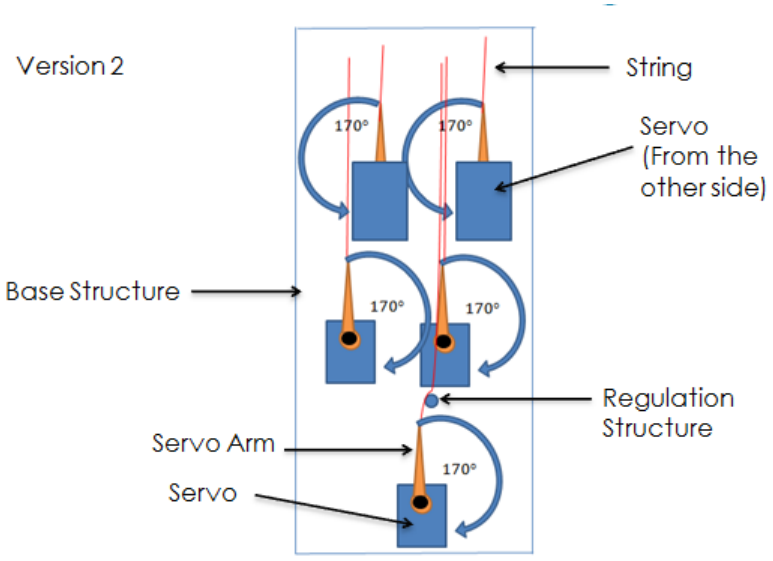


FIGURE 4.32: The second design alternative of the servo motor positioning

The difficulties of implementing designs with up-side-down servos was foreseen, hence another design had to be implemented. The third design structure (shown in Figure 4.33) was the one chosen design mainly because of the compactness of the servo motors and the the uniform functionality of each.

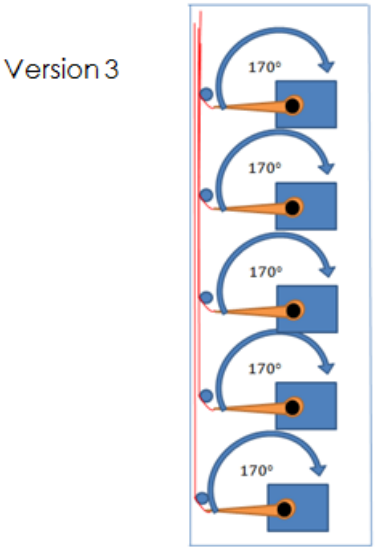


FIGURE 4.33: The third design alternative of the servo motor positioning, which is the chosen one.

This base structure is to be contained in what will be the robotic forearm. Which is discussed next.

4.3.1 Robotic Forearm Structure

Not only does the forearm need to have enough room for the servo motors and the rotating arms, but also it should be both strong to carry weights and light. A good choice of the forearm that has those three characteristics is a light cylindrical container, which will contain the base structure of the servos as shown in Figure 4.34. More precisely, the container that is used is a cylindrical potato chips container, which provides room for the rotating servo arms, is rigid since it is made of hard cardboard, and is light.

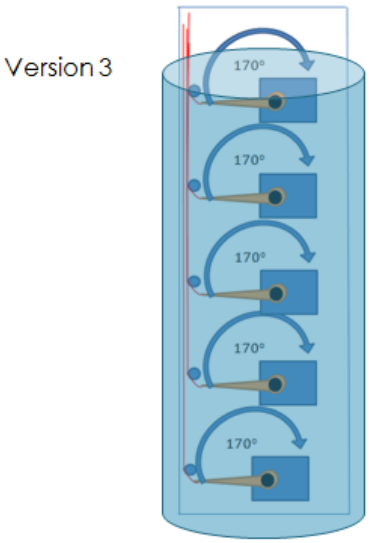


FIGURE 4.34: The forearm implemented is a cylindrical structure in which the servo motor base structure will be put.

The base structure whose design is shown in Figure 4.33 was implemented by using Styrofoam board as the base structure and toothpicks as the regulation structures. The design was accounting the fact that the strings were made of fishing lines that have minimal friction with each other, thus not affecting the functionality of flexing the fingers. Figure 4.35 shows the implemented design of the servo motor positioning.

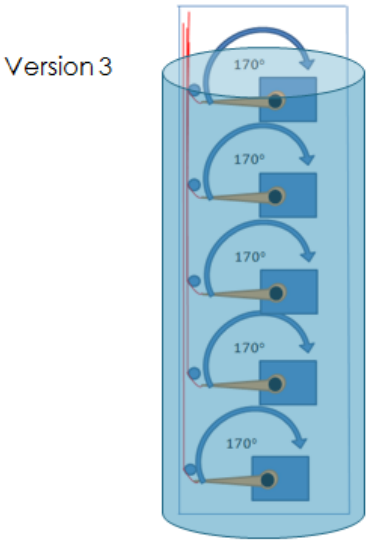


FIGURE 4.35: The five servo motors positioned in a foam-board base structure. The toothpicks act as the regulation structures.

With the dimensions designed to be perfectly fitting in the cylindrical container, the servo motor positioning structure was cut and inserted in the cylindrical forearm structure, as shown in Figure 4.36.

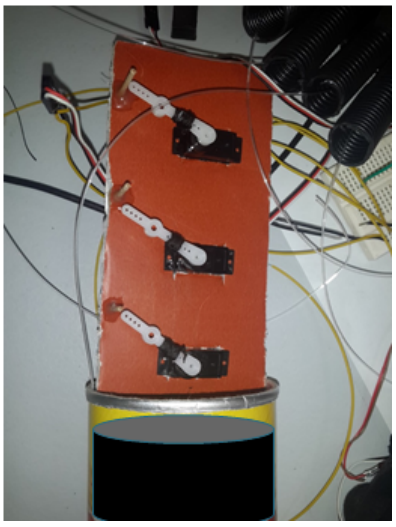


FIGURE 4.36: The base structure of the five servos inside the cylindrical potato chip container.

Section 4.3.2 discusses the rolling and pitching of the forearm, including both the structure used and the actuation mechanism (the use of higher-torque servo motors).

4.3.2 Robotic Forearm Rolling and Pitching

Since the forearm structure for Mark 2 is composed of cardboard cylinder, the rolling and pitching were implemented by attaching the servo motors directly to the cylinders. This design provides the forearm with minimal weight, which minimizes the load on the higher-torque servos. Figure 4.37 shows s sketch of Mark 2 design, which has one servo positioned on an outer structure while another mounted in the structure on a smaller cylindrical container.

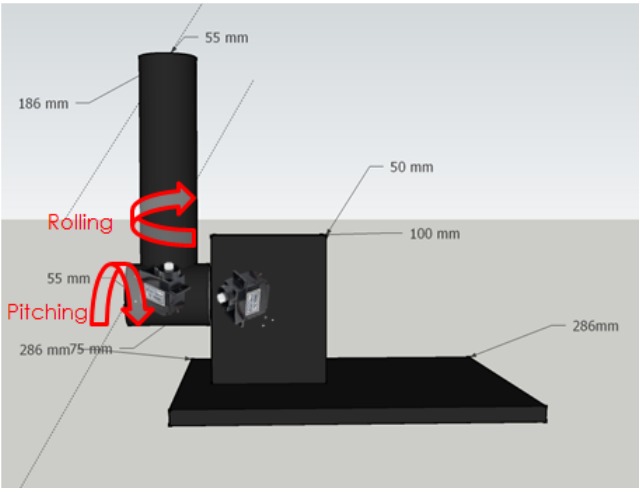
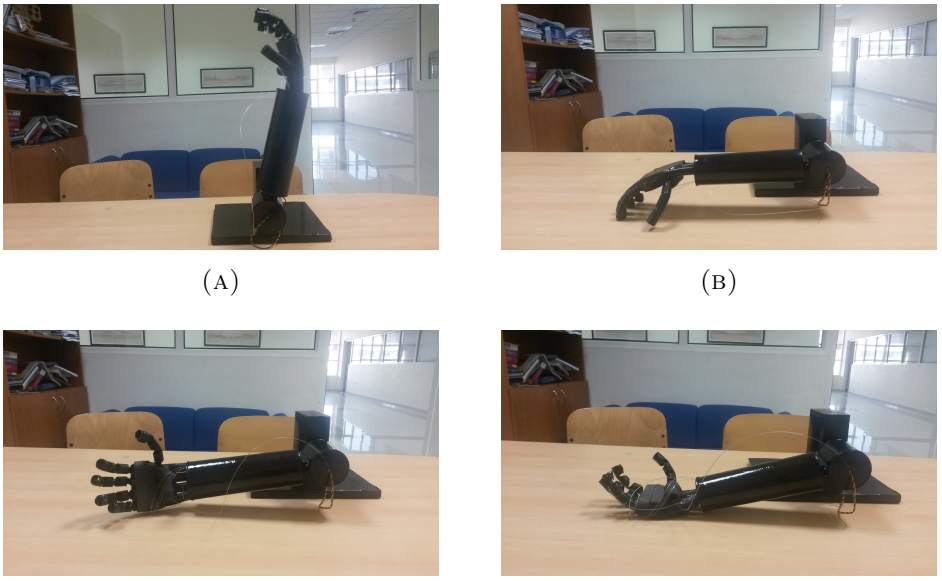


FIGURE 4.37: The Mark 2 design of the rolling and pitching, with a smaller cylindrical container added to contain the rolling servo, and is attached to the rotating arm of the pitching servo.

Mark 2 structure is put on a wooden base, with box structure carrying the pitching servo motor. Attached to the pitching servo motor is another cylindrical structure similar to the forearm structure (but smaller in size). This small cylinder contains the other higher-torque servo motor, which is attached to the base of the forearm and takes care of rolling it. Figure 4.38 shows the built structure.

FIGURE 4.38: Mark 2 design for the rolling and pitching built from two cardboard cylindrical containers, a cardboard box, and a wooden base.

Figure 4.39 shows the complete structure of the Mark 2 with different combinations of rolling and pitching positions.



(A) (B)

(C) (D)

FIGURE 4.39: The complete structure of Mark 2 in different rolling and pitching positions

4.3.2.1 Higher-torque Servo Motors

The high torque servo motor operates on the same principle as the low torque servo motors discussed in 4.2.2.2.However, the higher torque servo motor produces substantiuly higher torqe as illustrated Figure 4.40. However, due to there high torque output, powering them from the microcontroller was never an option. There are three possible alternative of powering the servo motors, both low torque and high torque.

The three modes for driving the servo motors are:

- Driving circuit
- Servo motor shield for the microcontroller/arduino
- Power bank.

Due to time constraints, the power bank was chosen to drive the servo motors.(the power banks where at our disposal) Moreover, the power banks have the ability to supply 5 volts at 2.1 amps, which is more then enough to power all 7 servo motors (5 lower-torque and 2 higher-torque). However, is it worth mentioning that power banks have an auto switch off circuitry, that would automatically switch the power bank off if it doe not detect an output current higher of a certain level, in order to prevent the power bank from switching off; the power bank was hacked into and the auto switch off circuitry was disabled

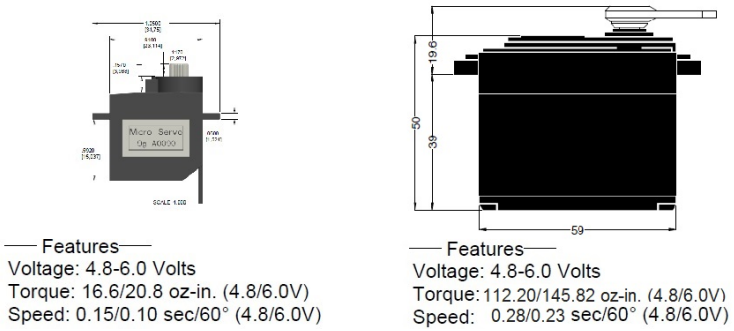


FIGURE 4.40: Difference between the lower and higher torque servo motors

After Mark 2 was complete, there were several issues with the prototype that were compromising its functionality. First, the material of which the Mark 2 robotic hand is made weak. The weakness was concluded after testing the prototype to carry objects of different weights. The prototype was only able to carry cylindrical objects with a mass of 0.3 kg or less. One reason for this weakness is the material from which Mark 2 is made. Although the hollow plastic tubes composing the fingers and the cardboard composing the forearm were good options for a light prototype, they were barely able to lift objects. Another reason for the weakness is the way the pitching was implemented. The forearm of Mark 2 is held by the pitching servo motor from only one point, as apparent in Figure 4.37. A second issue with Mark 2 is that it lacks sustainability. This was discovered when the fingers began breaking after several trials of operating the system. the issue of sustainability was also faced with the grip of Mark 2 hand, which is made of sponge tape. Aside from providing little grip, the material took less than a month to start wearing off, thus making the prototype not functional since it has a short life. Additionally, the structure that holds the finger-flexion servo motors, which is made of Styrofoam, is also not sustainable. After several times of usage and testing of the prototype, and due to the continuous tension on the servo motors, the structure was breaking apart. Furthermore,

the positioning of the finger-flexion servo motors and the method of implementing a better tendon-pull was also not sustainable. this is due to the elongation of the servo motor lever arms, which was implemented by hand and could not provide a longer rigid lever arm. All those reasons have made the move to a better functional and sustainable structure necessary. As a result, there was a move to Mark 3.

4.4 Mark 3 Hand and Forearm

Almost all the functionality and sustainability issues faced with Mark 2 are not found in Mark 3 (Figure 4.41). The printed robotic hand and forearm were used from open-source designs by [3] and [4]. One of the reasons for that is the material from which Mark 3 is made. Since Mark 3 is 3D printed, it is made of strong plastic material. The phalanges, shown in Figure 4.42, are strong, unlike the hollow plastic tubes used in Mark 2.

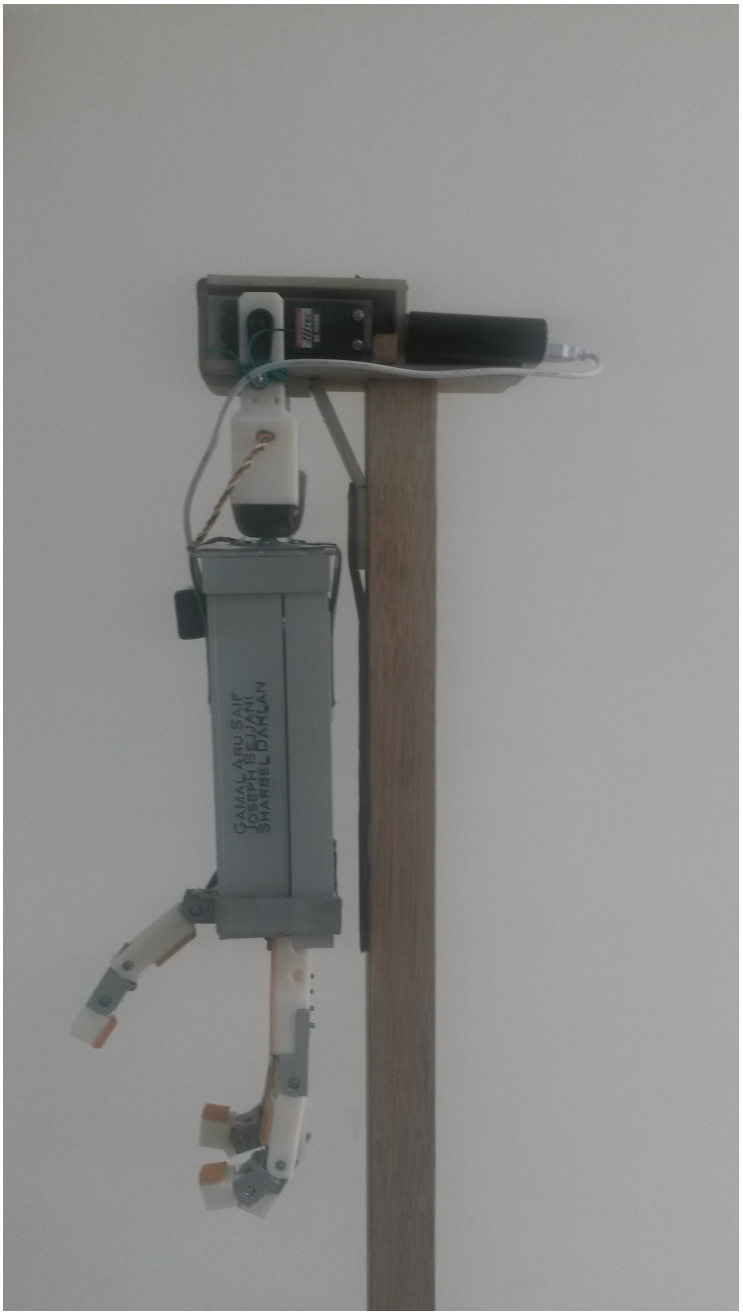


FIGURE 4.41: Mark 3 printed structure

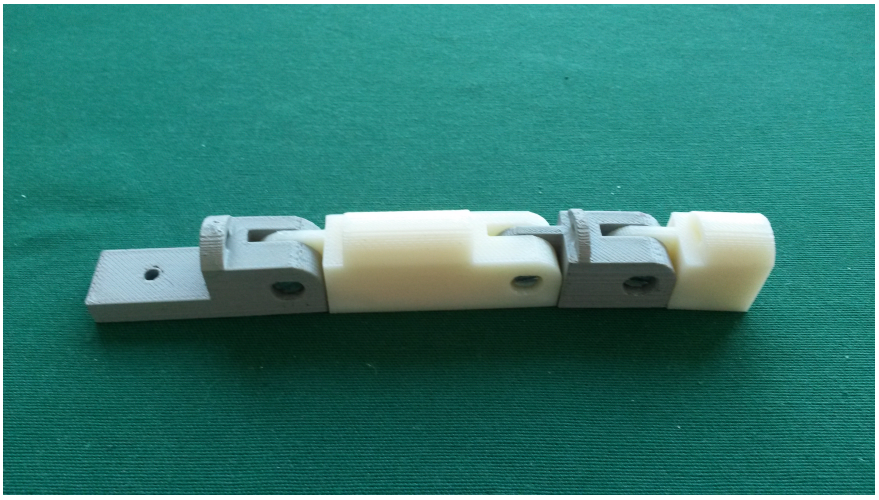


FIGURE 4.42: Mark 3 printed finger

Another reason for Mark 3 being better than Mark 2 is the sustainability of the grip material. Rather than using the spongy material that was used previously, rubber material obtained from table tennis rackets was used. The two factors that led to using this material are the better grip and higher sustainability of the rubber material. Figure 4.43 shows Mark 3 arm at the stage of adding rubber material to increase its grip.

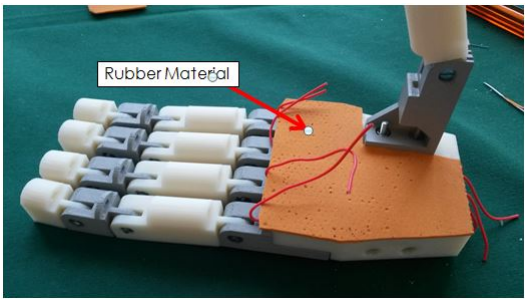


FIGURE 4.43: The grip material of Mark 3 hand made of rubber

A third advantage of Mark 3 over Mark 2 is the way of implementing the positioning finger-flexion servo motors. The forearm that is printed accounts for the five finger-flexion servos, as shown in Figure 4.44. Each servo has a room in which it is put and held firmly, while providing more elevation for the servos at the back. This in turn makes the tendons to be attached to each servo while not getting tangled with the other tendons or servos.

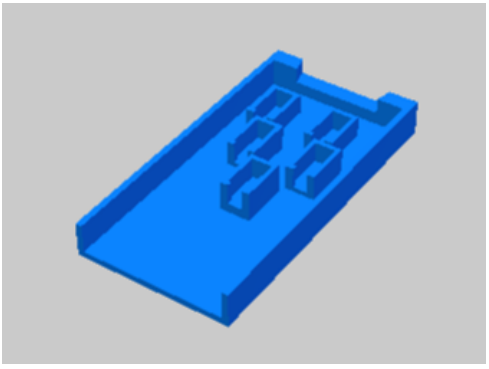


FIGURE 4.44: A sketch of the robotic forearm, which contains rooms for the five finger-flexion servo motors.

A fourth advantage is the way the maximum possible tendon pull is achieved. The servo motor lever arms have pulley-wheel-like structures (Figure 4.45) on which they are mounted and then attached to the servo body.

FIGURE 4.45: Printed pulley wheel structures that are attached to the servo motor lever arms and mounted on the servo body.

Those structures achieve a better pull of the strings by allowing the strings to be trapped in a track and be wrapped around the wheel as the servo motor arm is rotating. The result is shown in Figure 4.46.

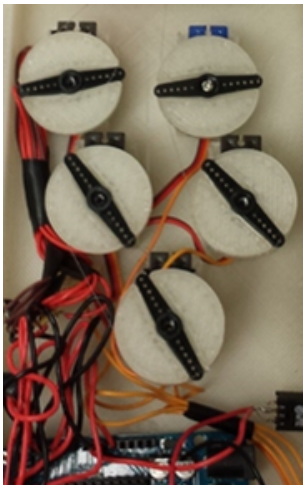


FIGURE 4.46: Positioning of the servo motors on the printed forearm. Those pulley wheels provide a track for the strings to be trapped in and wrapped around them, while achieving the maximum pull of the tendons attached to the fingers.

A fifth advantage is the implementation of the pitching in Mark 3, which provides a better support for the arm structure. With the use of U-shaped brackets around the pitching servo motor, the arm structure is given double the support of Mark 2, with one side of the bracket attached to the servo motor arm and another side to the bearing at the back of the servo. This is illustrated in Figure 4.47.

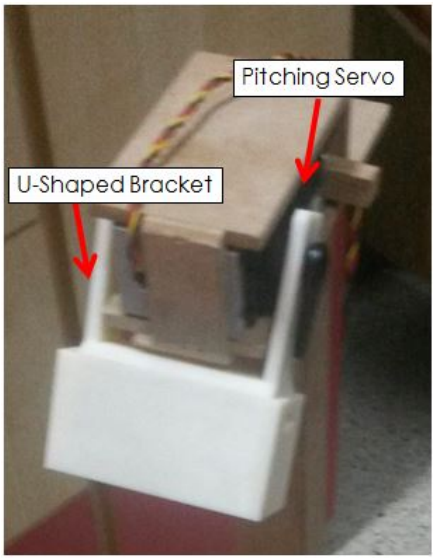


FIGURE 4.47: U-Shaped Bracket attached to the pitching servo motor

With this printed structure, the pitching and the rolling of the servo motor is done, as shown in Figure 4.48.

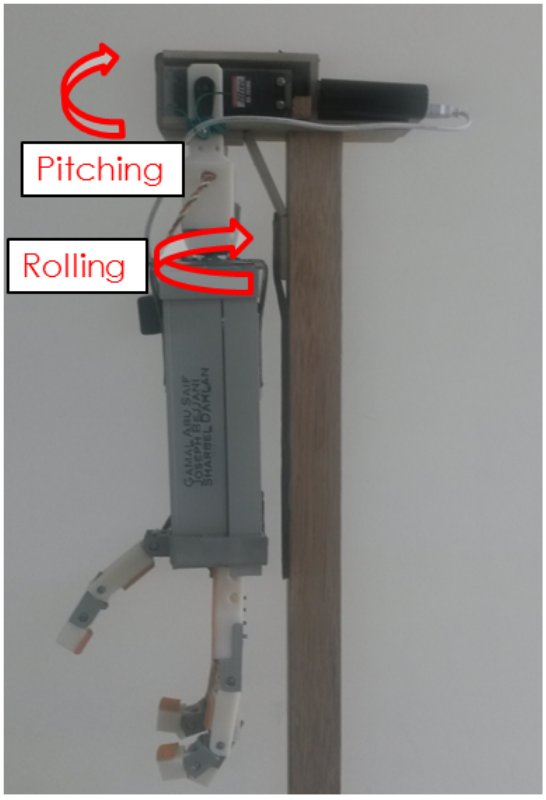


FIGURE 4.48: Printed pulley wheel structures that are attached to the servo motor lever arms and mounted on the servo body.

With all those advantages of the printed robotic arm, the design was tested with different objects and found to be successful of carrying light objects.

in addition to the

4.5 Microcontrollers

The microcontroller(s) we choose to use is the Arduino Uno and an Arduino Mega 2560, the reason behind choosing this specific brand of microcontroller is for the following reason:

1. Small in size.

2. Cheap, in order stay within budget.
3. Not a lot of computational power is needed.

The reason for moving from an Arduino Uno to an Arduino Mega was the lack of analog pins needed. The project needed seven analog pin of which five will be for the flex sensors and two for the IMU. Moving on from the Uno to Mega did not prove difficult since the two microcontroller have almost the same setup with little difference in the number of ports available.

The Arduino Mega with the following specifications, satisfies all our needs related to this project

Specifications:

1. ATmega2560 microcontroller
2. Input voltage from 7V to 12V
3. 54 Digital I/O Pins (14 PWM outputs)
4. 16 Analog Inputs
5. 256k Flash Memory
6. 16Mhz Clock Speed

Arduino is a single-board microcontroller, projected to make the use of interactive objects or environments more open. The hardware comprises of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. Present devices feature an USB interface, 6 analog input pins, as well as 14 digital I/O pins which allow to attach various extension boards. A main piece of the Arduino is the typical way that connectors are exposed, thus permitting the CPU board to be attached to a range of substitutable add-on modules known as shields. Some shields communicate with the Arduino board directly over several pins, but many shields are separately addressable via an I^2C serial bus, letting many shields to be loaded and used in parallel. Certified Arduinos have used the mega AVR series of chips, specially the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560.

4.5.1 Steering Side Microcontroller

The steering microcontroller uses the Arduino Mega 2560. The Arduino Mega 2560 is a microcontroller board centered on the ATmega2560. It has 54 digital input/output pins that of which 15 can be used as PWM (pulse width modulation) outputs. It also has 16 analog inputs, 4 UARTs (universal asynchronous receiver transmitter) hardware serial ports, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP (in circuit serial programing) header, and a reset button. It comprises of everything required to support the microcontroller. To use it simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery. The Arduino Mega (Shown in Figure 4.49) is well-matched with most shields designed for the Arduino Duemilanove or Diecimila. The Mega 2560 is an update to the Arduino Mega, which it replaces.

When using the Mega the connections of the IMU changed from being analog pins to there designated pin SDA and SCL. This made the usage of the Mega 2560 a much better choice than the initial choice which was the Uno. The Table 4.4 contains the specifications of the Mega 2560.

TABLE 4.4: Arduino Mega Specifications

Microcontroller	**ATmega2560**
Operating Voltage	5V
Input Voltage	7 - 12 V
Digital I/O Pins	54
Analog Input Pins	16
Flash Memory	256 KB
Cloak Speed	16 MHz

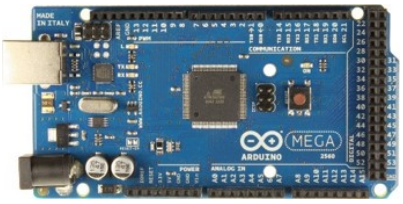


FIGURE 4.49: Arduino Mega Steering Side

4.5.2 Action Side Microcontroller

Here in the action side, the microcontroller used is the Arduino Uno. The Arduino Uno is a microcontroller board centered on the ATmega328. It has 14 digital input/output pins that of which 6 can be used as PWM (pulse width modulation) outputs. It also has 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP (in circuit serial programing) header, and a reset button. It comprises of everything required to support the microcontroller. To use it simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery. The Uno varies from all previous boards in that it does not use the FTDI USB-to-serial driver chip. As an alternative, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. Figure 4.50 shows the Arduino Uno.

There reason an Arduino Uno was used here and not a Mega is because the pins needed at the steering side are all only digital pins. There are 14 digital pins and the project uses 7 digital pins overall. Table 4.5 summarizes the specifications of the Uno.

TABLE 4.5: Arduino Uno Specifications

Microcontroller	**ATmega328**
Operating Voltage	5V
Input Voltage	7 - 12 V
Digital I/O Pins	14
Analog Input Pins	6
Flash Memory	32 KB
Cloak Speed	16 MHz

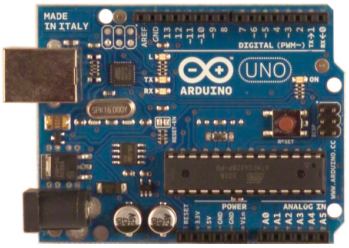


FIGURE 4.50: Arduino Uno Action Side

4.6 Wireless Communication

In the robotic arm system, both the action side and the steering side communicate wirelessly. This communication, which involves transmitting and receiving data between the microcontrollers on each side, should be achieved with high ranges and minimal delay, as the project requires. Therefore, a study of the available modes of wireless communication was done to make the best decision on the one that fits the criteria needed for the robotic arm system.

Due to having both the needed range of operation and the low power consumption, ZigBee was the technology chosen for the wireless communication of this project. More information about ZigBee technology can be found in Appendix F, while Section 4.6.1 discusses the ZigBee-based product that will be used in the project.

4.6.1 XBee

The XBee 802.15.4 RF module that will be used in this project, Digi's XBee™ 802.15.4 OEM RF modules can be set up to operate in a point-to-point, point-to-multipoint or a peer-to-peer configuration. The reason for choosing it from the other modules is that XBee 802.15.4 are suitable for low-power and low-cost applications, while XBee-PRO modules are suitable for extended-range applications with higher power requirements. In addition, since XBee module is specified to provide point-to-point 802.15.4 communications, it is simpler to implement, these XBee modules are easy to use, and are fully interoperable with other XBee products utilizing the same technology. This in turn keeps the interchanging of the modules with minimal development and time risk [5]. Figure 4.51 shows a Digi XBee radio transceiver.

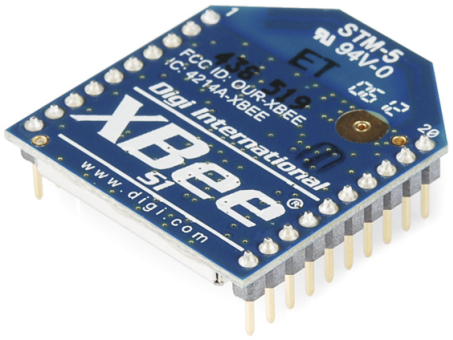


FIGURE 4.51: An XBee radio transceiver

Currently, the two XBee radio transceivers have been worked with and were configured to communicate with each other. A guide on configuring the two XBees is provided in Appendix C. Eventually, those XBee radios will be mounted on each microcontroller and communicate with each other by sending packets. Since this is the working mechanism of serial communication, there should be a study on how the packets are implemented in the design.

4.6.2 Packets

When dealing with communication there is a data side and a medium side. The medium side can either be wired so wired communication or wireless so wireless communication. It has been made obvious earlier that the medium of communication for the project will be a wireless communication. Moving to the data that will be sent, the team will be using packets. The packets will contain information from the flex sensors and IMU that will be sent to the other microcontroller in order for the servo motors to move. The packets will have a start and stop byte (one byte each), there is also the ID byte, and finally the data bytes. The reason that everything is represented in a byte is because after mapping the IMU and flex sensors, everything is represented in degrees from 0 to 170 degrees. If the degrees are to be mapped into bytes then having 2^7 which is 128 is not enough to represent all the degrees the servos need. Another reason for representing them in one byte is that the Zigbee technology has a wide range of bandwidth of 2.4GHz. There are

more than one scenario to create the packets. First scenario is to have all the data bytes in one packet with the start and stop bytes. In this case there would be no need for an ID byte. This case can be shown in Figure 4.52. This is one of the extreme ways of creating a packet.

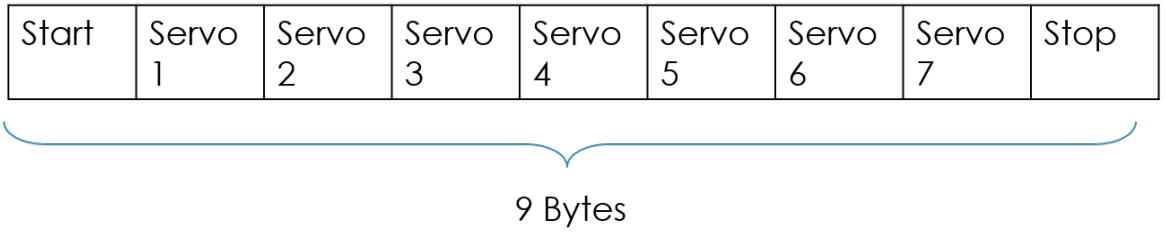


FIGURE 4.52: Case 1: all data bytes together in one packet

The next case is also an extreme case where all data bytes will be presented in a single packet each. Here there will be seven different start and stop bytes with seven IDs one for each packet. The overall design is that there will be seven different packets to send from one microcontroller to the other. An illustration regarding the second case is shown in Figure 4.53.

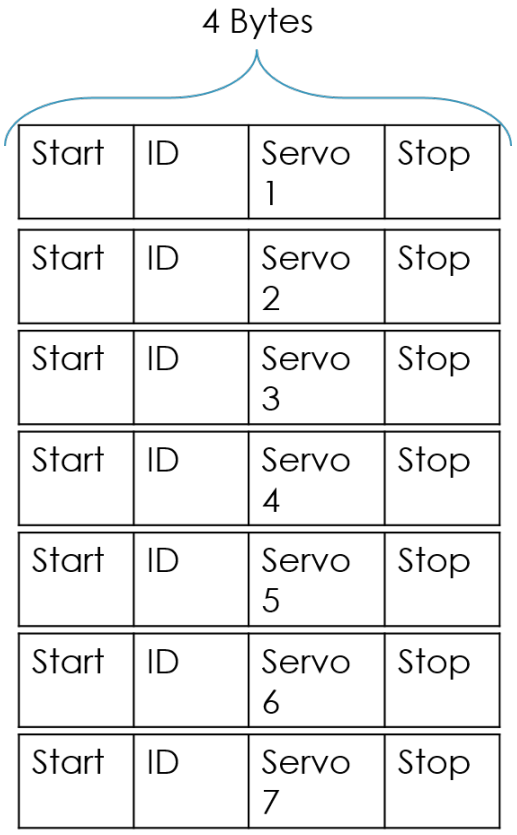


FIGURE 4.53: Case 2: all data bytes separated in seven different packet

The final case is having five data bytes that are related to the flex sensors with a start, stop and ID bytes. The other two data bytes that belong to the IMU are in a separate packet that also has a start, stop and ID bytes. Figure 4.54 shows the configuration of the final case.

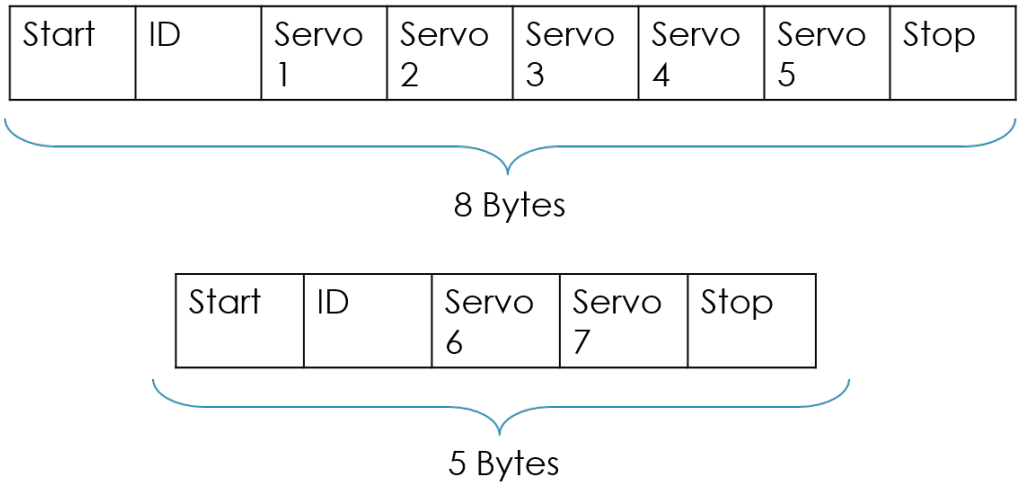


FIGURE 4.54: Case 3: five data bytes are together in one packet and the other two data bytes in a single packet

After looking at the three different cases we ended up implementing Case 2 as shown in Figure 4.53. The reason we picked case 2 was due to the nature of how the Xbee operates. We simply write the values from the sensors on the steering side Xbee. Which in turn would send them to the Xbee on the action side.

Chapter 5

Limitations, Schedule, and Budget

Whether technical or non-technical, there are limitations that exist to the wirelessly steered robotic arm system. However, those limitations were kept in mind since the beginning, and most of them could be tackled should there be more time and budget. The technical limitations are related to the biomechanics of the arm, while non-technical limitations include time, budget, economical, environmental, sustainability, social impact, etc.

5.1 Limitations and Constraints

The proposed solution has a number of technical limitations, mainly due to how the scope of the project was defined. Other limitations may include time, knowledge, and budget the degrees of freedom of the robotic arm and the flexing of the robotic arm.

5.1.1 Technical Limitations and Constraints

The first and major limitation is the degrees of freedom implemented in this project. As mentioned, the degrees of freedom are as follows, the first is the flexing of the fingers, second is the rolling of the wrist on finally the pivoting of the elbow up and down. These limitations occur because of several reasons, lack of knowledge in regards to mechanical aspects of the project, also the time and budget constraints. Both the time and budget allocated is insufficient for having a fully-fledged working arm. Figure 5.1 shows the degrees of movement that cannot be achieved in this project.

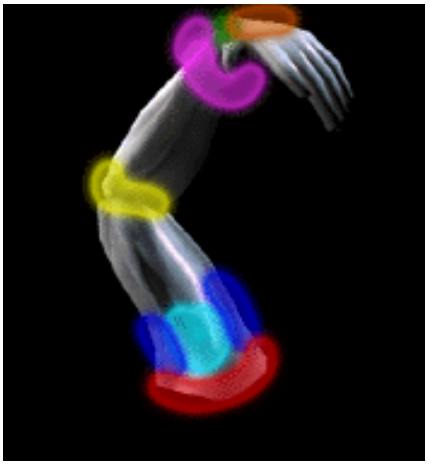


FIGURE 5.1: Degree of Movement

The second limitation is the flexing of the fingers of the robotic hand. The implantation of the project limits the flexing the robotic fingers into a circular fashion only and this happens because of the use of threads to move the fingers instead of having a motor for every joint in the finger. This is a limitation because humans can flex their fingers in any way or shape they like. In this case also the lack of knowledge in mechanics and both the time and budget constraints play a huge role in this limitation. Figure 5.2 shows the flexing of the robotic fingers.

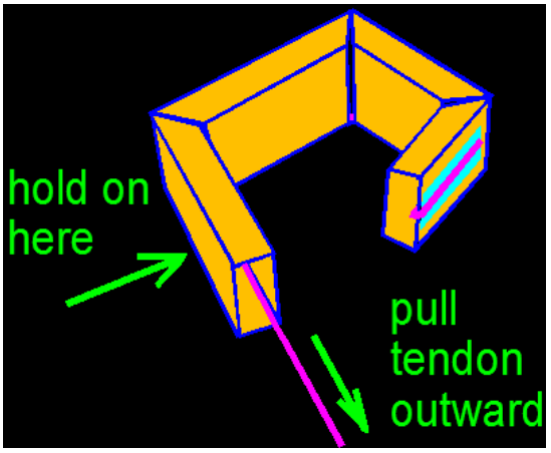


FIGURE 5.2: Robotic finger flexes in circular fashion

The third limitation is the robotic structure's limited functionality. To this point in the semester the functionality of the robotic arm is limited to only objects to a certain weight and size. The objects that are the main focus of this project are:

- Square/Rectangular
- Cylindrical

However, the weight of the objects that the robotic arm can carry is yet to be determined; more experimentation has to be done in order to accurately determine the weight

5.1.2 Non-technical Limitations and Constraints

There are many non-technical constraints that any group working on a project should consider. Unfortunately, many engineers tend to focus more on the technical aspect of the project and neglect all non-technical constraints. However, the non-technical constraints are very important, and if not taken into consideration, can result in the failure of the project. First let us consider the most important of all non-technical constraints, Economic.

Economic One of the most important non-technical constraint in a project is cost. Solving the problem is not always the best solution, for example: what is the point of find the cure for a specific disease, if the cure is going to be sold to public for 10 million dollars; although the cure is functional and important, it will not reach the public due to its inflated price. Given that every group had a budget of 1,500 Dhs, we where obliged to stick to this amount. However, our project required around 700 Dhs more, an approval was given by the department, because we justified about the additional costs. More information about the budget can be found in the 5.3. The second non-technical constraints any group should consider is the project's effect on the environment.

Environmental Every project - no matter how small it is - will have some sort of effect on the environment. The engineer has to consider what impact the project has on its surroundings, and the impact of the surroundings on the performance of project. The engineer has to consider whether the material used has any effect on air, water, noise, Energy saving devices, etc... . In the case of the wirelessly steered robotic arm, the current robotic structure mainly consist of recycled materials, which were found around the university or house; for example, the forearm is made entirely out of cylindrical chip

cans. However, if we consider implementing a 3D printed structure, then there should be a re-evaluation of the environmental factors, because there will be many prototypes that are not going to be used, thus this will cause an increase in materials that will be disposed. However, the materials will be recycled and not disposed. Considering that recycled materials are used, sustainability is a factor that should be considered, because the sustainability of the product should not suffer if recycled materials are used.

Sustainability Sustainability is the ability of an engineer to design a project or product that would perform under normal operating conditions for a given length of time, and the engineers have to consider using local materials and labor. In this project, the materials were order from the United States, which does not comply with the sustainability aspect. However, the structure of the robotic arm was made from materials that where found locally. The Life time of any product is usually determined by the shortest lifetime of all the products that make the project. In our case, it was noticed that the low torque servo motors have the lowest life time, if not placed or used in the correct manner; this is usually a problem with all micro sized servo motors, because their gears produce a large amount of heat. In general sustainability can not be completely covered without mentioning build quality and construcatability of the project.

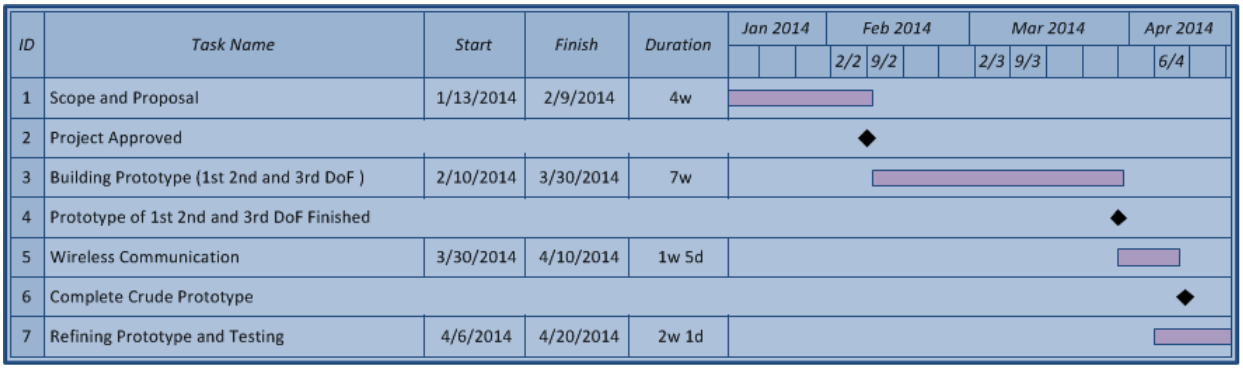
Constructability The material used for the robotic arm at the current stage is all made by the 3D printer. However, when it came to the steering side(Augmented glove), most of the materials were sourced from the United States, making it hard to replace materials if anything should happen. This is why five extra flex sensors and two extra low torque motors were ordered. In case one would fail, a backup is ready to be used. Since the main aspect of any project is functionality, when it comes to the robotic arm built from recyclable materials it might not be at full functioning capacity (Functionality in this case is the ability to mimic the gestures of the hand and hold light material). This is why 3D printing would increase its functionality, because the grip would increase. Every project has a an impact on society in one way or another, it is very important that the social impact on of the wirelessly steered robotic arm be discussed.

Ethical With any project comes a load of ethical responsibilities, given that this project is an electrical engineering project, The IEEE codes of ethics have to be considered, and some ethical regulations regarding robotics have to be considered. First, it is safe to say that the IEEE codes of ethics are being followed, because they are present in every single engineering course. Second, the ethical issue is regarding the product liability laws regarding robotics, since it is a genuinely a new field. With this said if for example a robotic structure were to fail then there are no indications as to which party is to be blamed. However ethical issues are not complete if, the team did not consider whether their project is abiding by a certain health and safety regulations. The IEEE code of ethics can be found in Appendix D.

Health and Safety Health and Safety are two of the most important aspects all engineers have to consider when they are working on a project. They have to consider the safety of the public, safety of the workers, and the hazardous materials and environment for workers. In the case the Wirelessly steered robotic arm, the workers in this case are the students, although there was some materials that have to be cut using an automated drill, the students maintained the appropriate level of safety. The wirelessly steered robotic arm currently in the stage were there are alot of exposed wires; however, the wires will be covered and reduced in time of the final presentation. As mentioned in previous sections of the non-technical limitations, the robotic arm is only a proof of concept, and will not be used to handle dangerous tasks. Last but not the least, with every project comes a deadline. Dealing with deadlines is one of the most important issues that needs to be considered.

Time Time is always an issue when it comes to project. Without deadlines, projects would simply run their course, and become obsolete before their release. For the wireless robotic arm, there was only three months to come up with a functional prototype; this is not an easy task, if time was not managed properly. This is why the a schedule was proposed in the beginning of the semester. This schedule has to be followed precisely, or risk not having a prototype by the end of the semester. for more information about time management and scheduling please refer to 5.2.

5.2 Schedule

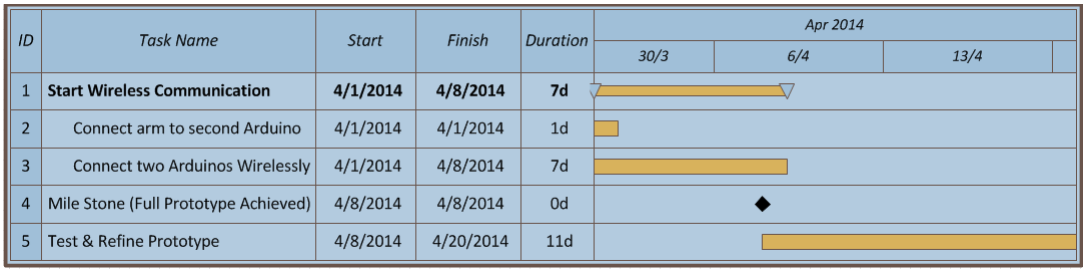


ID	Task Name	Start	Finish	Duration
1	Scope and Proposal	1/13/2014	2/9/2014	4w
2	Project Approved			
3	Building Prototype (1st 2nd and 3rd DoF)	2/10/2014	3/30/2014	7w
4	Prototype of 1st 2nd and 3rd DoF Finished			
5	Wireless Communication	3/30/2014	4/10/2014	1w 5d
6	Complete Crude Prototype			
7	Refining Prototype and Testing	4/6/2014	4/20/2014	2w 1d

FIGURE 5.3: High level Schedule

Figure 5.3 shows the high level schedule of the project. For the first month we will be working on the scope and proposal of our project by then we will have reached our first milestone. For the next month and a half we will be working on building our prototype which will entail our three degrees of movement. Hopefully we will be done with it by the time of our midterm presentation where we will be displaying the prototype before the committee. The rest of the third month will be for the wireless communication. We will work on making the two microcontrollers, one on the glove and the second on the robotic arm, communicate wirelessly. The final task will be for refining i.e. printing the robotic hand and forearm, and then testing the new final prototype before presenting it in the final presentation.

5.2.1 Wireless Communication



ID	Task Name	Start	Finish	Duration
1	**Start Wireless Communication**	**4/1/2014**	**4/8/2014**	**7d**
2	Connect arm to second Arduino	4/1/2014	4/1/2014	1d
3	Connect two Arduinos Wirelessly	4/1/2014	4/8/2014	7d
4	Mile Stone (Full Prototype Achieved)	4/8/2014	4/8/2014	0d
5	Test & Refine Prototype	4/8/2014	4/20/2014	11d

FIGURE 5.4: Wireless Communication Schedule

After finishing the midterm presentation the group started working on the wireless communication. A gantt chart representing the schedule of the wireless communication is shown in Figure 5.4. The procedure entailed first knowing the type of packets that will be sent and received by the two Arduinos. Next, the team worked first with wired communication in order to understand how the system works and also to check if the program was working perfectly. After finishing the wired communication the team moved to wireless where they started with setting up the Xbee devices and then connecting the two Xbees to communicate. Next we test the final prototype for any irregularities in the system before moving on to refine the prototype and presenting it in the final presentation. The process of setting up the wireless communication took approximately one week, after that, the team had two more weeks to refine and test the prototype.

5.3 Budget

The price for all the products needed is within range of the projects budget as can be seen in Table 5.1. We needed two types of servo motors because we have two different torque levels, five finger flexion servo motors for the fingers and two motors, pitching and rolling servo motors, for the rolling of the wrist and pitching of the arm. Some new products have been added to the list like the finger flexion servo motors and portable charger. The additional products were approved by the supervisor Dr. Adnan El Nasan. In the case of the servo motors, two servos were damaged. All the prices seen are taken from the same source in order to reduce the shipping price.

Table 5.1: Price Table

Product	Unit Price	Quantity	Total Price
Robotic arm structure	$20	1	$20
Glove	$15	1	$15
Flex Sensors	$12.95	5	$77.70
Gyro Board	$39.95	1	$39.95
Arduino Uno	$30	2	$60
XBee Wireless	$95.95	1	$95.95
Servo Motors (Low Torque)	$8.95	5	$44.75
Servo Motors (Low Torque)	$10	2	$20
Servo Motors (High Torque)	$39.95	3	$119.85
Portable Chargers	$13	1	$13
Total Price in Dollars			$ 506.2
Total Price in Dirhams			AED 1857.75

Chapter 6

Conclusion

Research and projects in the field of robotics and control systems target the solution of real-life problems. With the wirelessly steered robotic arm, several real-life applications that benefit the people are targeted. Those applications include handling hazardous tasks or work that requires power beyond the human power (such as the exoskeleton), and helping disabled people. Although these application are vastly targeted, the project wirelessly steered robotic arm is a proof of concept where the robotic arm can only work in three degrees, flexing the fingers, rolling the wrist and pitching the elbow. The robotic arm here can only do minor applications like lifting small objects with small weights. Through future improvements the robotic arm can reach to the ultimate goals that are discussed in the motivation.

6.1 Future Work

In this section the future improvements that can be made to the project will be discussed. Some of the future improvements include a GPS feature, extended degrees of freedom, improving functionality, and finally a camera. The GPS feature will have a position tracking. This entails having the robotic arm position always visible for the human steering the robotic arm. Another feature is to avoid kidnapped robot problem. This problem is a situation where when a robotic structure is used and sent to a remote location far away from the human steering it, having a GPS tracker will aid the robotic structure to find its way around and back to the human. The current robotic arm moves only in three degrees, and a human arm can move in seven to nine degrees as mentioned before. Adding more degrees of movement to the robotic arm would increase the amount of tasks the robotic arm can do. Improving functionality includes first having the robotic arm to be as accurate as the human arm that it will replace. Secondly applying a feedback and

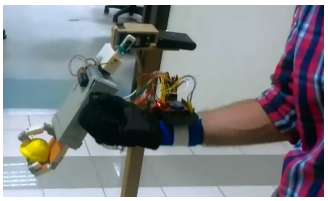
making the system a closed loop will also improve functionality since the output recorded at the action side will always be seen back again at the steering side thus reducing any errors. This will allow the human steering to get a better feel of the robotic arm. A camera is also important since the steering side and action side will be communicating wirelessly so the camera will be the eyes of the human steering the robotic arm.

6.2 Summary

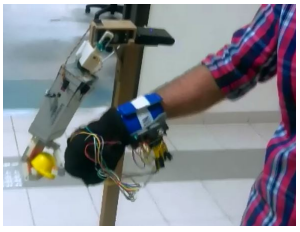
Building a robotic arm that can flex and extend its fingers, as well as roll and pitch its forearm in the same manner a human arm is acting is achieved with the use of the wirelessly steered robotic arm system. Comprised of an augmented glove which the human wears to steer the robotic arm, the system can act in real-time. The augmented glove is fitted with five flex sensors and an IMU to sense the human arm motion and action, while the robotic arm is attached to five lower-torque servo motors and two higher-torque servo motors to actuate its motion. Additionally, a control unit on each side records the state of the human arm and actuate the servo motors to move accordingly. One set of sensory information that the steering-side microcontroller reads is the voltage values passed across each flex sensor to get the finger flexing information, which is mapped to degrees by which the servo motor arms rotate, thus pulling the strings attached to the robotic fingers in a controlled fashion. Likewise, another set of values the steering-side microcontroller reads is the position and acceleration values of the IMU, which are also mapped to degrees by which the higher-torque servo motor arms attached at the base of the forearm turn. Furthermore, the two sides communicate wirelessly with the help of two XBee transmitters and receivers (each on one microcontroller), to allow the action side to be up to a 100 meters away from the steering side. Using the ZigBee technology which is based on the IEEE 802.15.4 protocol, the XBee radio transceivers periodically send two packets from the augmented steering glove to the robotic arm, with one packet containing flex-sensor information and another containing IMU information. The full system made of Mark 3 structure was tested, with some captures shown in Figure 6.1.

(A)

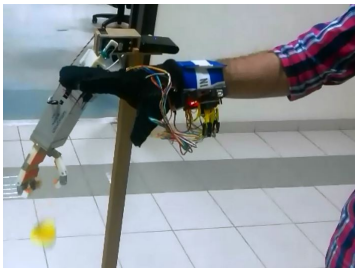
(B)



(C)



(D)



(E)

FIGURE 6.1: The complete structure of Mark 3 being tested in different positions

Serving as a proof-of-concept for larger systems, the wirelessly steered robotic arm is successful in reaching the goal that was set in the scope. With such a system, the robotic arm is capable of both mimicking the gestures of the human arm, and holding objects of certain light weights.

References

[1] "The comparison of wi-fi, bluetooth and zigbee." Sena Blog. [Online]. Available: http://www.sena.com/blog/?p=359. [Accessed 13 March. 2014].

[2] "White paper: Demystifying 802.15.4 and zigbee®." Digi International Inc. [Online]. Available: http://www.digi.com/pdf/wp_zigbee.pdf. [Accessed 13 March 2014].

[3] embeddedjunkie, "Robotic hand v3.0." MakerBot Thingiverse ®[Online]. Available: http://www.thingiverse.com/thing:14986, note = [Accessed 3 April 2014],.

[4] travistag, "Full robotic hand based on robotic hand v3.0." MakerBot Thingiverse ®[Online]. Available: http://www.thingiverse.com/thing:225683. [Accessed 3 April 2014].

[5] "Xbee® 802.15.4." Digi International Inc. [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module. [Accessed 13 March 2014].

[6] J. A. Stankovic *et al.*, "Real-time computing," *Byte, pág*, pp. 155–162, 1992.

[7] J. A. Stankovic, "Misconceptions about real-time computing: A serious problem for next generation systems," *IEEE Computer*, vol. 21, 1988.

[8] "Zigbee technology." ZigBee Alliance. [Online]. Available: https://www.zigbee.org/About/AboutTechnology/ZigBeeTechnology.aspx. [Accessed 10 March 2014].

[9] "Imu introduction." InvenSense. [Online]. Available: https://www.invensense.com/mems/gyro/mpu6050.html. [Accessed 2 April 2014].

[10] "I²c introduction." Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/I%C2%B2C. [Accessed 6 April 2014].

[11] "Need wireless? choose xbee." Digi International Inc. [Online]. Available: http://www.digi.com/xbee/. [Accessed 13 March 2014].

[12] "Zigbee® low-cost, low-power, wireless networking for device monitoring and control." Digi International Inc. [Online]. Available: https://www.digi.com/technology/rf-articles/wireless-zigbee. [Accessed 12 March 2014].

[13] S. Narasimhan, "Dexterous robotic hands: kinematics and control," tech. rep., DTIC Document, 1988.

[14] "Anatomy of the human hand." Paradoja [Online]. Available: http://www.paradoja7.com/human-hand-anatomy-pictures/. [Accessed 28 April 2014].

[15] "Phalanges." [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/a/ab/Scheme_human_hand_bones-en.svg. [Accessed 28 April 2014].

Appendix A

Flex Sensor code

Listing A.1

LISTING A.1: Code used to read from Flex sensors and output to lower-torque servos

```
#include <Servo.h> // Includig servo libraries

Servo Servo1,Servo2,Servo3,Servo4,Servo5; // creating 5 different servos
    for the 5 fingers
int Finger1,Finger2,Finger3,Finger4,Finger5,degrees1,degrees2,degrees3,
    degrees4,degrees5; // creating integer values for Fingers, which will
    have the analog read values,

                    // degrees would be the mapped values from the analog
    values

void setup()
{

  // read the voltage from the voltage divider (sensor plus resistor)

  // initialize serial communications
  Serial.begin(115200);

  Servo1.attach(7);// Attach servo 1 to digital pin 7
  Servo2.attach(3);// Attach servo 2 to digital pin 3
  Servo3.attach(4);// Attach servo 3 to digital pin 4
  Servo4.attach(5);// Attach servo 4 to digital pin 5
  Servo5.attach(6);// Attach servo 5 to digital pin 6
}

void loop()
{
```

```
25  Finger1 = analogRead(0); // read value from analog pin 0 and store it
     Finger1
    Finger2 = analogRead(1); // read value from analog pin 1 and store it
     Finger2
27  Finger3 = analogRead(2); // read value from analog pin 2 and store it
     Finger3
    Finger4 = analogRead(3); // read value from analog pin 3 and store it
     Finger4
29  Finger5 = analogRead(4); // read value from analog pin 4 and store it
     Finger5

31  // Normalization of the flex sensros using simple if, and else if
     statments.

33  if (Finger1 < 550) Finger1 = 550;
    else if (Finger1 > 690) Finger1 = 690;
35
    if (Finger2 < 550) Finger2 = 550;
37  else if (Finger2 > 690) Finger2 = 690;

39  if (Finger3 < 510) Finger3 = 510;
    else if (Finger3 > 655) Finger3 = 655;
41
    if (Finger4 < 570) Finger4 = 570;
43  else if (Finger4 > 650) Finger4 = 650;

45  if (Finger5 < 595) Finger5 = 595;
    else if (Finger5 > 665) Finger5 = 665;
47


49  degrees1 = map(Finger1, 550, 690, 0, 170); //number 29,pointer
    degrees2 = map(Finger2, 550, 690, 0, 170); //number 4,ring
51  degrees3 = map(Finger3, 510, 655, 0, 170); //number 12,pinky
    degrees4 = map(Finger4, 570, 650, 0, 170); //number 19,thumb
53  degrees5 = map(Finger5, 595, 665, 0, 170); //number 10,middle finger


55
   //print out the result of each of the fingers
57
    Serial.print("Pointer : ");
```

```
59   Serial.print(Finger1,DEC);
     Serial.print("   degrees1: ");
61   Serial.println(degrees1,DEC);
     Servo1.write(degrees1); // write mapped pointer flex sensor values to the
       assigned servo motor.
63
     Serial.print("Ring    : ");
65   Serial.print(Finger2,DEC);
     Serial.print("   degrees2: ");
67   Serial.println(degrees2,DEC);
    Servo2.write(degrees2);// write mapped Ring flex sensor values to the
      assigned servo motor.
69
     Serial.print("pinky   : ");
71   Serial.print(Finger3,DEC);
     Serial.print("   degrees3: ");
73   Serial.println(degrees3,DEC);
     Servo3.write(degrees3);// write mapped Pinky flex sensor values to the
      assigned servo motor.
75
     Serial.print("Thumb   : ");
77   Serial.print(Finger4,DEC);
     Serial.print("   degrees4: ");
79   Serial.println(degrees4,DEC);
    Servo4.write(degrees4); // write mapped Thumb flex sensor values to the
      assigned servo motor.
81
     Serial.print("Middle  : ");
83   Serial.print(Finger5,DEC);
     Serial.print("   degrees5: ");
85   Serial.println(degrees5,DEC);
     Servo5.write(degrees5);// write mapped Middle flex sensor values to the
      assigned servo motor.
87
     Serial.print("\n");
89
     delay(250);
91
     // pause before taking the next reading
93 }
```

Appendix B

IMU Sensor code

Listing B.1

LISTING B.1: Code used to read from the IMU and output to higher torque servos

```
#include "I2Cdev.h"
#include <Servo.h>
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif
MPU6050 mpu;
#define OUTPUT_READABLE_YawPitchRoll
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 =
    success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float pr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and
    gravity vector
float pivot;
float roll;
int degrees1, degrees2;
```

```
Servo Servo1, Servo2;
29
31
33 // ================================================================
   // ===                INTERRUPT DETECTION ROUTINE                 ===
35 // ================================================================

37 volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt
      pin has gone high
   void dmpDataReady() {
39     mpuInterrupt = true;
   }
41

43
   // ================================================================
45 // ===                      INITIAL SETUP                         ===
   // ================================================================
47
   void setup() {
49     // join I2C bus (I2Cdev library doesn't do this automatically)
       #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
51         Wire.begin();
           TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
53     #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
           Fastwire::setup(400, true);
55     #endif

57     Serial.begin(115200);

59     while (!Serial); // wait for Leonardo enumeration, others continue
      immediately

61     // initialize device
       Serial.println(F("Initializing I2C devices..."));
63     mpu.initialize();
       Servo1.attach(9);
65     Servo2.attach(10);
```

```
67      // verify connection
        Serial.println(F("Testing device connections..."));
69      Serial.println(mpu.testConnection() ? F("MPU6050 connection successful"
    ) : F("MPU6050 connection failed"));

71      // load and configure the DMP
        Serial.println(F("Initializing DMP..."));
73      devStatus = mpu.dmpInitialize();

75      // supply your own gyro offsets here, scaled for min sensitivity
        mpu.setXGyroOffset(220);
77      mpu.setYGyroOffset(76);
        mpu.setZGyroOffset(-85);
79      mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

81      // make sure it worked (returns 0 if so)
        if (devStatus == 0) {
83          // turn on the DMP, now that it's ready
            Serial.println(F("Enabling DMP..."));
85          mpu.setDMPEnabled(true);

87          // enable Arduino interrupt detection
            Serial.println(F("Enabling interrupt detection (Arduino external
    interrupt 0)..."));
89          attachInterrupt(0, dmpDataReady, RISING);
            mpuIntStatus = mpu.getIntStatus();
91
            // set our DMP Ready flag so the main loop() function knows it's
    okay to use it
93          Serial.println(F("DMP ready! Waiting for first interrupt..."));
            dmpReady = true;
95
            // get expected DMP packet size for later comparison
97          packetSize = mpu.dmpGetFIFOPacketSize();
        } else {
99          // ERROR!
            // 1 = initial memory load failed
101         // 2 = DMP configuration updates failed
            // (if it's going to break, usually the code will be 1)
```

```
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}



// ================================================================
// ===                      MAIN PROGRAM LOOP                   ===
// ================================================================

void loop() {
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize)
    {

    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too
   inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
```

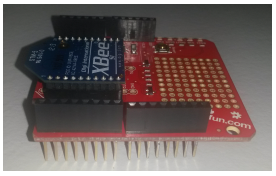
```
141     // otherwise, check for DMP data ready interrupt (this should happen
    frequently)
    } else if (mpuIntStatus & 0x02) {
143         // wait for correct available data length, should be a VERY short
    wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
145
        // read a packet from FIFO
147     mpu.getFIFOBytes(fifoBuffer, packetSize);

149     // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an
    interrupt)
151     fifoCount -= packetSize;


153
        #ifdef OUTPUT_READABLE_YawPitchRoll
155         mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
157         mpu.dmpGetYawPitchRoll(pr, &q, &gravity);
            pivot = pr[1] * 180/M_PI;
159         roll = pr[2] * 180/M_PI;
        #endif
161
        if (pivot < -70) pivot = -70;
163     else if (pivot > 70) pivot = 70;
        degrees1 = map(pivot, -70, 70, 0, 170);
165     Serial.print("Pivot: ");
        Serial.print(pivot,DEC);
167     Serial.print("   Degrees1: ");
        Serial.println(degrees1,DEC);
169     Serial.print("\t");
        Servo1.write(degrees1);
171


173     if (roll < -75) roll = -75;
        else if (roll > 75) roll = 75;
175     degrees2 = map(roll, -75, 75, 0, 90);
        Serial.print("\t");
177     Serial.print("Roll: ");
```

```
        Serial.print(roll,DEC);
179     Serial.print("   Degrees2: ");
        Serial.println(degrees2,DEC);
181     Servo2.write(degrees2);


183

        // blink LED to indicate activity
185     blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
187 }
}
```

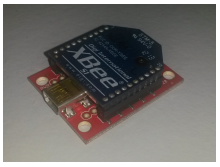
Appendix C

XBee Configuration

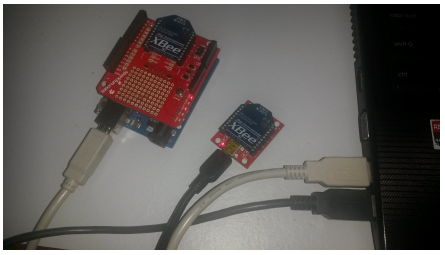
In this appendix, the configuration of the XBee radios will be shown. Before starting with the configuration, the XBee arduino shield was soldered with the connectors so that it can be mounted on the arduino. One XBee radio was mounted on the shield (Figure C.1a), while the other was mounted on the XBee explorer(Figure C.1b). Both the arduino with the mounted XBee shield and the explorer were connected to the computer (Figure C.1c).



(A) The arduino shield with the soldered connectors and the XBee radio mounted



(B) The XBee radio mounted on its explorer



(C) Both the arduino with the XBee radio on a shield and the explorer connected to the computer

FIGURE C.1: The steps done before running the X-CTU software two XBee radios connected to the computer

C.1 Downloading the X-CTU Software

First the X-CTU Software was downloaded from www.digi.com/xctu, which when entered, the page in Figure C.2 appears. The *Diagnostics, Utilities, and MIBs* tab was

selected (Figure C.3) and the relevant version is selected. For this case, XCTU ver. 5.2.8.6 installer was selected.

FIGURE C.2: The XCTU software page from Digi.

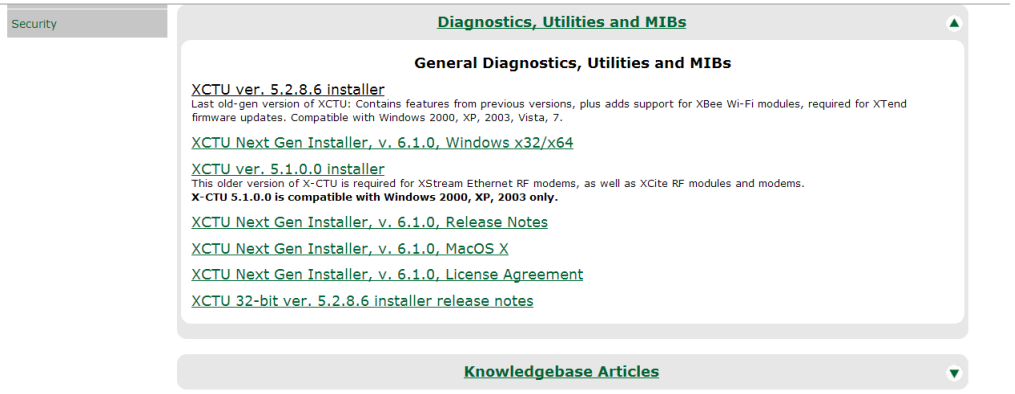


FIGURE C.3: The download link contained in *Diagnostics, Utilities, and MIBs* tab].

Once the Software has finished downloading, the program was installed on the computer.

Next, the X-CTU icon was run, with all the new updates being accepted. With the two XBees connected to the computer as in Figure C.1c, a window similar to the one in Figure C.4 appears in order to test the XBee explorer.

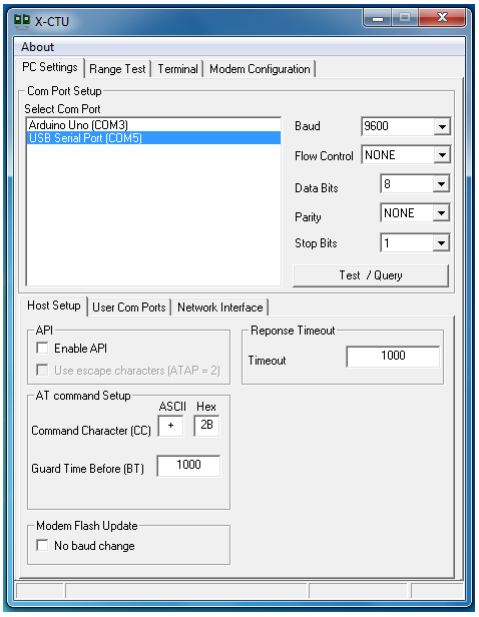


FIGURE C.4: Testing the XBee explorer

When the explorer was done testing, the result in Figure C.5 appears.

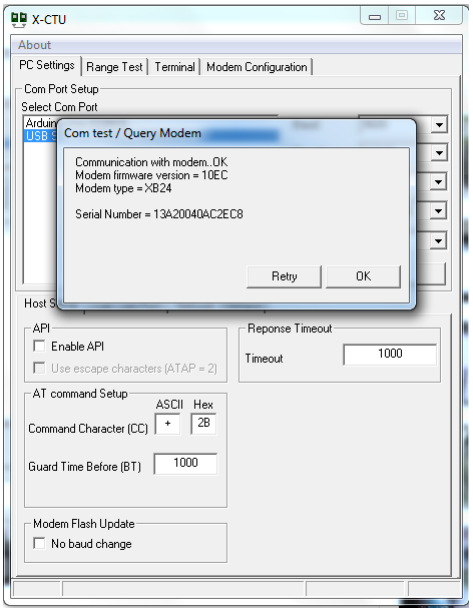


FIGURE C.5: Testing the XBee explorer

Next, the *Modem Configuration* tab was clicked, the *Always Update Firmware* checkbox in the *Modem Parameter and Firmware* group was checked, and the *Download New Versions...* button was clicked. Both the Modem and the Function set were set to XB24 and

XBee 802.15.4, respectively. This is shown in Figure C.6.

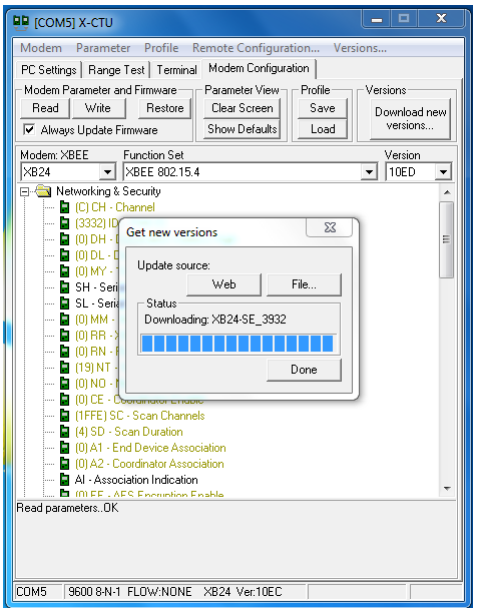


FIGURE C.6: Testing the XBee explorer

In order to change any of the configurations in the white box, the values were double-clicked and the desired value were entered. If you want, you can up the baud rate in order to speed up transmission.

The same process was repeated after mounting the second XBee on the explorer. The channel in which the XBee is working on, in addition to the network ID, and the baud rate should be the same, so that the XBees can communicate with each other.

Appendix D

IEEE Code of Ethics

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
3. to be honest and realistic in stating claims or estimates based on available data;
4. to reject bribery in all its forms;
5. to improve the understanding of technology; its appropriate application, and potential consequences;
6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

Appendix E

IMU Introduction

The MPU 6050 is the world's first and only 6-axis Motion Tracking device made for the low power, low cost, and high performance necessities of smart phones, tablets and wearable sensors. [9]

The MPU 6050 device brings together a 3-axis gyroscope and a 3-axis accelerometer on the same silicon mold. This can be seen in Figure E.1. The MPU 6050 also has onboard a Digital Motion Processor (DMP) adept of handling difficult 9-axis Motion Fusion algorithms. The parts incorporated 9-axis Motion Fusion algorithms that can access peripheral magnetometers or other sensors through an auxiliary master I^2C bus. This permits the device to collect a full set of sensor data without interference from the system processor. The size of the device is 4x4x0.9 mm QFN (quad-flat no-leads) footprint and pin-out thus giving a simple improvement path and making it easy to fit on space constrained boards. A more detailed architecture of the MPU 6050 is shown in Figure E.2.

In order for precision tracking of both fast and slow motions, the IMU feature a user-programmable gyro full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

The Inven-Sense Motion Apps Platform that comes with the MPU-6050 extracts motion-based difficulties, offloads sensor management from the operating system, and delivers an organized set of APIs for application advancement. Added features contain an embedded temperature sensor and an on-chip oscillator with ±1 percent variation over the operating temperature range.

FIGURE E.1: IMU Architecture

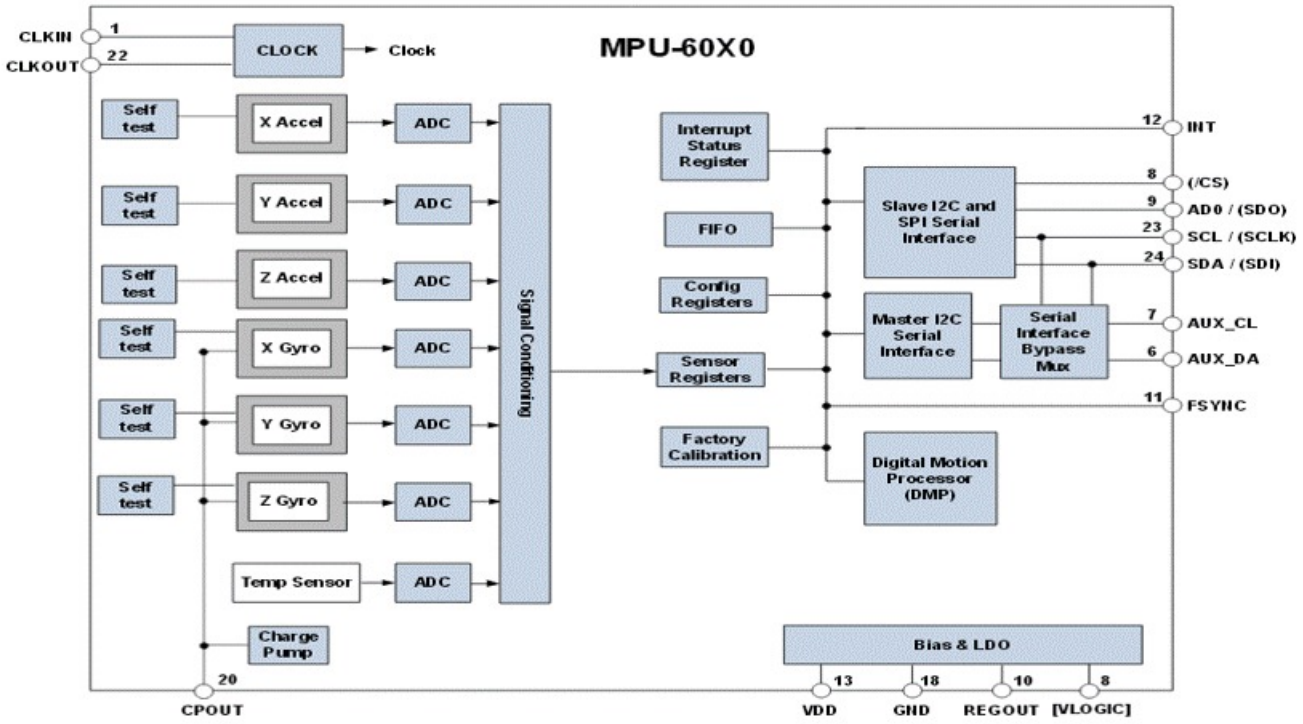


FIGURE E.2: IMU Detailed Architecture

Accelerometer

An accelerometer is a device that computes proper acceleration. The proper acceleration computed by an accelerometer is not automatically the coordinate acceleration that is the rate of change of velocity. Rather, the accelerometer perceives the acceleration related with the phenomenon of weight experienced by any test mass at rest while in reference to the accelerometer device. For instance, an accelerometer at rest on the surface of the earth will evaluate an acceleration of g= 9.81 m/s2 straight upwards, due to its weight.

In comparison, accelerometers in free fall or at rest in outer space will be evaluated as zero.

Another terminology for the type of acceleration that accelerometers can compute is g-force acceleration. Accelerometers are used to sense and observe vibration in rotating machinery. Single- and multi-axis models of accelerometer are able to sense magnitude and direction of the proper acceleration or g-force acceleration, as a vector quantity. They can also be used to sense orientation since the direction of weight changes, coordinate acceleration as long as it generates a g-force or a change in g-force, vibration, shock, and falling in a resistive medium which is a situation where the proper acceleration changes, where it starts at zero then increases. Theoretically, an accelerometer performs as a damped mass on a spring. When the accelerometer experiences an acceleration, the mass is displaced to the point that the spring is able to accelerate the mass at the same rate as the casing. The displacement is then computed to give the acceleration.

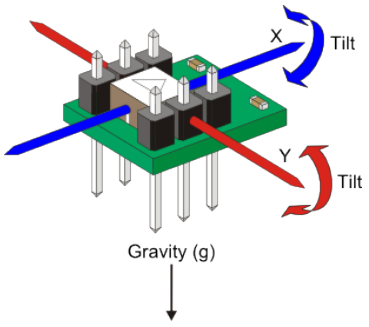


FIGURE E.3: A diagram of an Accelerometer

Gyroscope

A gyroscope is a device for computing or sustaining the position of an object, centered on the values of angular momentum. Mechanically, a gyroscope is a spinning wheel or circle where the axle is able to take up any position. Figure E.4. While this orientation does not stay stationary, it varies in reaction to a peripheral torque much less and in a changed path than it would with the huge angular momentum related with the circle's high level of spin and moment of inertia. The device's orientation rests almost stationary, unrelated to the mounting of the platform's motion, because mounting the device in a gimbal minimizes peripheral torque.

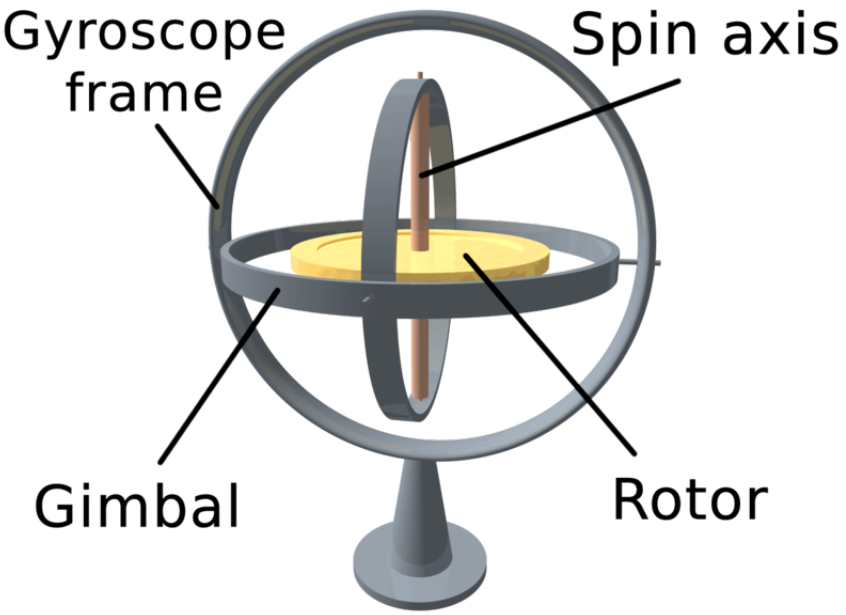


FIGURE E.4: A Gyroscope description

A vibrating structure gyroscope, standardized by IEEE as Coriolis vibratory gyroscope (CVG), is a wide group of gyroscope using solid-state resonators of different shapes that functions much like the halteres of an insect. The fundamental physical standard is that a vibrating entity inclines to remain vibrating in the same plane as its support rotates. This kind of equipment is also known as a Coriolis vibratory gyro because as the plane of oscillation is rotated, the reaction perceived by the transducer results from the Coriolis term in its equations of motion ("Coriolis force").

A MEMS gyroscope takes the idea of the Foucault pendulum and uses a vibrating element, known as a MEMS (Micro Electro-Mechanical System). Cheap vibrating structure gyroscopes mass-produced with MEMS technology have become commonly accessible. Figure E.5. These are enclosed alike to other integrated circuits and may provide either analog or digital outputs. In a lot of cases, a single part consist of gyroscopic sensors for multiple axes. Some parts integrate multiple gyroscopes and accelerometers or multiple axis gyroscopes and accelerometers, to achieve output that has six full degrees of freedom. These units are called Inertial Measurement units, or IMUs.

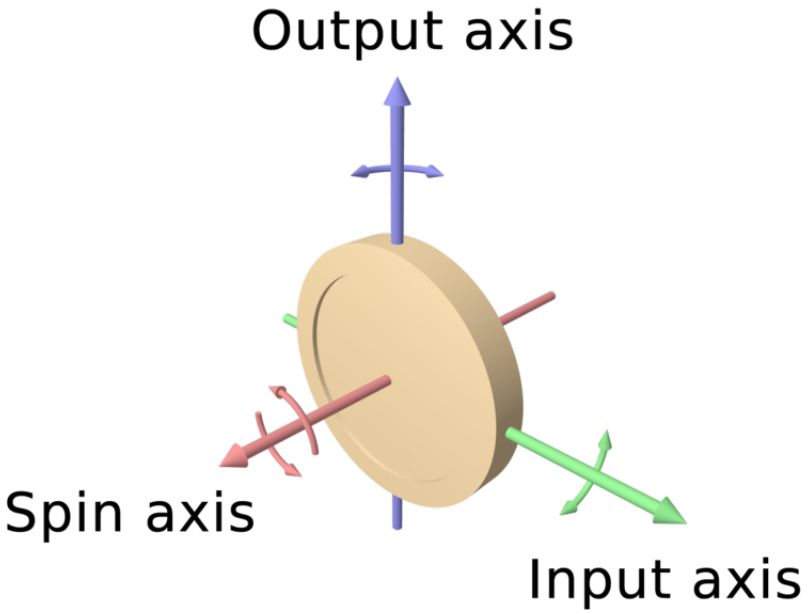


FIGURE E.5: A gyro wheel

I²C

I²C (Inter-Integrated Circuit, referred to as I-squared-C, I-two-C, or IIC) is a multi-master serial single-ended computer bus invented by the Philips semiconductor division, today NXP Semiconductors, and used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other digital electronic devices. [10]

I²C uses two two-way open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors which is directly configured on the chip. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. Figure E.6 shows an example of the usage of I²C while having three slave devices communicating with one master device (microcontroller).

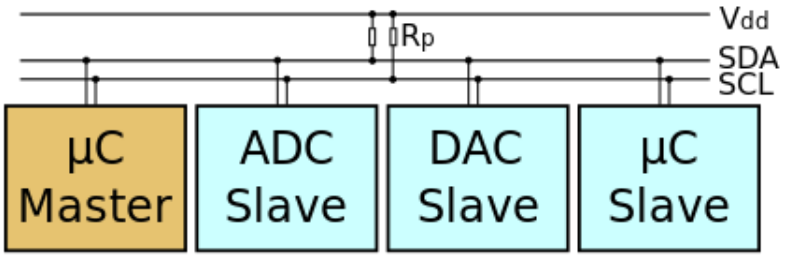


FIGURE E.6: A design of I²C where there is one master and three slave nodes

The I²C reference design has a 7-bit or a 10-bit address space depending on the device used. Usual I²C bus speeds are 100 kbit / s for standard mode and 10 kbit / s for low-speed mode, but random low clock frequencies are also permitted. Current changes of I²C can mass more nodes and run at faster speeds like 400 kbit/s Fast mode, 1 Mbit/s Fast mode plus (Fm+), and 3.4 Mbit/s High Speed mode. These speeds are more extensively used on embedded systems than on PCs. There are also other features, such as 16-bit addressing.

Note the bit rates are estimated for the connections between master and slave without clock stretching or other hardware overhead as can be seen from the figure above. Figure E.6. The real transfer rate of user data is lower than the peak bit rates. For instance, if each communication with a slave device incompetently permits only 1 byte of data to be transmitted, the data rate will be less than half the peak bit rate. The maximum number of nodes is limited by the address space, and also by the total bus capacitance of 400 pF, which restricts practical communication distances to a few meters.

Appendix F

Zigbee Technology

In this appendix, Zigbee technology will be discussed in detail.

ZigBee is a global, open-standard wireless-communications technology that operates on the IEEE 802.15.4 physical radio specification, with unlicensed industrial, scientific, and medical bands (ISM bands). The bands include 868-868.8 MHz (operated in Europe), 902-928 MHz (operated in the USA and Australia), and 2.400-2.4835 GHz (operated in most jurisdictions worldwide)[12][1] [2]. ZigBee's specification is a packet-based radio protocol intended for low-cost and low-power wireless networks [8]. The protocol allows devices to communicate in a variety of network topologies and can have battery life lasting several years. Being formally created by member companies of the ZigBee Alliance, the ZigBee protocol was designed to provide an easy-to-use wireless data solution characterized by secure, reliable wireless network architectures [12].

ZigBee, which communicates data through hostile radio frequency environments common in industrial and commercial applications, has several advantages distinguish it from other protocols. First, ZigBee supports multiple network topologies, such as point-to-point (which is used in this project), point-to-multipoint, and mesh networks, with up to 65,000 nodes per network. In addition, by having low duty cycle, ZigBee as mentioned earlier, provides low power consumption (hence longer battery life). Moreover, the data connection is secure due to supporting 128-bit AES encryption. Furthermore, data transmission through ZigBee is highly reliable through its collision avoidance, retries and acknowledgments.

A member of the ZigBee Alliance, Digi, has developed a wide range of networking solutions based on the ZigBee protocol. XBee, XBee-PRO 802.15.4, and other XBee products are examples of the modules that provide an easy-to-implement solution, which provides functionality to connect to a wide variety of devices. Particularly in this project, the XBee 802.15.4, formerly known as XBee Series 1, will be used.