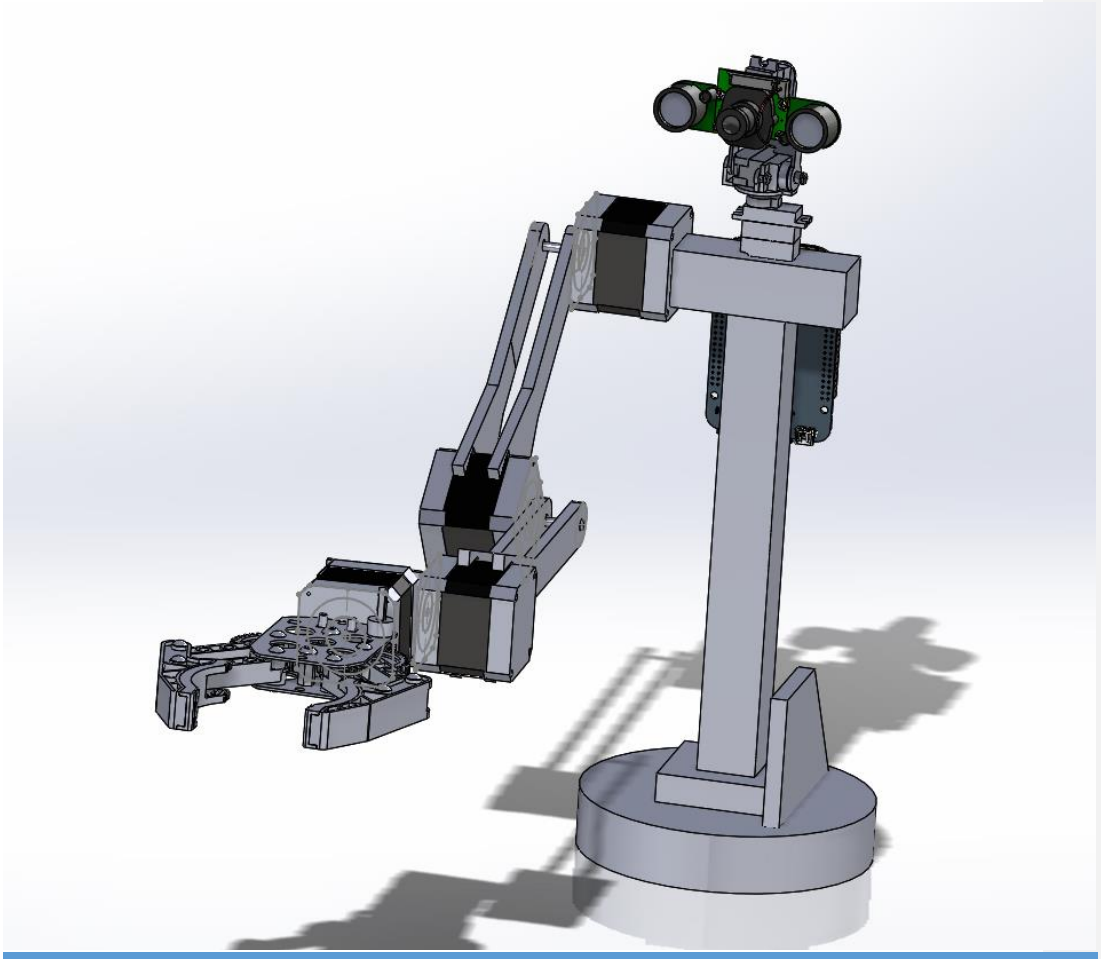


VR Telepresence Robot Arm

Project Report

Klin Rothenberger Aaron Ingram Salvador Cuevas Eduardo Calderon

Table of Contents

Commented [RR1]: Don't worry about the table of contents for now, the formatting is just messed up in the online version. Whenever you download it as a word doc its fixed

VR Telepresence Robotic Arm

Aaron Ingram

Salvador Cuevas

Eduardo Calderon

Klin Rothenberger

CONCEPT OF OPERATIONS

REVISION – 2

Dec 2018

CONCEPT OF OPERATIONS

FOR

VR Telepresence Robotic Arm

TEAM TARMU

APPROVED BY:

_____

Project Leader Date

_____

T/A Date

_____

John Lusher II, P.E. Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2/21/2018	Team 8		Draft Release
1	2/5/2018	Team 8		1st Semester Final Draft
2	12/6/2018	Team 8		Post-Completion Update

Table of Contents

Executive Summary

The Virtual Reality Telepresence Robotic Arm is a telepresence system that will allow a user to see to control a robotic arm from a distance while also being able to observe the environment in which the arm is located. There are many applications that the Robotic Arm can provide such as assisting rescue missions where the conditions of the environment could prove to be too hazardous for human intervention. The system will have two main parts, the Virtual Reality controller that the user will be using and then the robotic arm which would be in a different location than the user. The Virtual Reality headset will provide the display to the user to which the cameras on the robotic arm will capture. The motion of the arm will be dictated by an InfraRed sensor located on the headset that will track the user's arm movement and send these movements to the robotic arm to be replicated.

Introduction

We were motivated to create a device to simulate the human arm in movement so that control would be intuitive to a typical user. Another motivation was to provide the concept of telepresence in the product to allow for remote control. Thus, we chose to develop a robotic arm that could be controlled using common hand motions and producing video image of the arm to the user. This would allow a user to pick up items by moving their arm in a motion of picking up an object, and the robotic arm would emulate the movement. The video image would reflect the environment where the arm is so that the user can see objects around the arm. The product would allow the user remote control over a robotic arm with aided remote vision to perform actions such as picking up items, pushing items around, or releasing items.

Background

Currently, most robotic arms on the market are catered to producing strict and precise movement with the goal for manufacturing. Areas like the automotive industry, CNC machining and such utilize the arms exceptionally with the repetitive motions. The push for automation in the industry results in an increasing demand for robotic arms. The downside is that most development is focused on robots that can be pre-programmed to do tasks at a large scale.

Thus, a desire to build a robot arm that has the capability of being controlled through human motion is warranted. In addition, telepresence to the arm will bring the capability of a user to see the environment through the internet and make any necessary motions needed dependent on the scenario.

Overview

The system we will develop will consist of the user's space and the robotic arm's space.

The user space will involve the use of a Virtual Reality headset and Infrared sensor (Leap Motion™ controller) that will be wired into a Personal Computer. The headset and sensor will be connected into the Computer through USB and the computer will be sending the information from the user to the arm. The user will be seeing the robotic arm's space through the headset and the Infrared sensor on the headset will track the user's arm and send movement data to the robotic arm.

The robotic arm space will include a robot arm with servos, a camera mounted system with servos, and a battery pack to allow power to be sent throughout the system. The arm will be hardwired through ethernet to reduce latency between the user and the arm. When the user moves their head around to see, the camera system will move accordingly to allow the user to view the environment. When the user moves their arm, the arm will receive the information and mimic the user's movement.

Referenced Documents and Standards

802.3bv-2017 - IEEE Standard for Ethernet
https://static.oculus.com/documentation/pdfs/intro-vr/latest/bp.pdf

http://blog.lemoneerlabs.com/FILES%2f2013%2f10%2fBBB_30fps.pdf.axdx

https://cdn-shop.adafruit.com/datasheets/BBB_SRM.pdf

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield-v2-for-arduino.pdf

Operating Concept

Scope

The objective of this project is to implement the VR Headset with the Leap Motion™ controller (LMC) as input that uses hand-tracking to track the human hand simultaneously to control a physical robot arm replica to a human arm.

Current goals of the project include:

- Developing program that will translate data from LMC to robotic arm
- Design microcontroller system to rotate stepper motors
- Implementing microcontroller to exchange data from camera to motors
- Design independent power system for the microcontrollers and step motors

This project will be limited as a proof-of-concept and not a feature complete prototype.

Some of these limitations include:

- Constrained axis of movement, the arm will only be able to move in two planes of movement
- No finger tracking, the infrared camera will only be able to detect a binary change of whether the user's hand is open or closed
- Power source dependent on wall outlet.
- Wired connection from the Beaglebone to the computer
- Limited field of view from the stereoscopic camera
- Half-scale size compared to a normal human arm.

Operational Description and Constraints

To use the VR Telepresence Robotic Arm, the user will need to wear a VR headset combined with the Leap Motion™ controller. The Leap Motion™ Controller consists of two cameras and three infrared LEDs to track infrared light. The infrared LEDs are used to track the hand position of the user and the headset will track the head position of the user. A program will be written to read all data being transmitted by the LMC. The program will then begin by first receiving input from the user through the LMC. Due to large amounts of data being taken, commands will then be sent via Ethernet to the BeagleBone. The Beaglebone will be in control of the Camera and Camera Gimble. The Beaglebone will process the commands from the VR Headset for the head position and rotate the Camera Gimble to the desired angle. The video information from the camera will be processed by BeagleBone and sent to the VR Headset over Ethernet. The VR headset will display the video information that was given by the BeagleBone. The BeagleBone will be connected in serial to the Arduino microcontroller. This connection will be responsible for sending the information of hand sensors that the BeagleBone received to the Arduino.

The Arduino will then process the hand position information and rotate the respective motors needed to emulate the user's hand position. The BeagleBone, Arduino, and Stepper motors would be powered by an external power supply. The VR Headset and LMC will be powered by a Personal Computer that is hooked into a wall outlet.

System Description

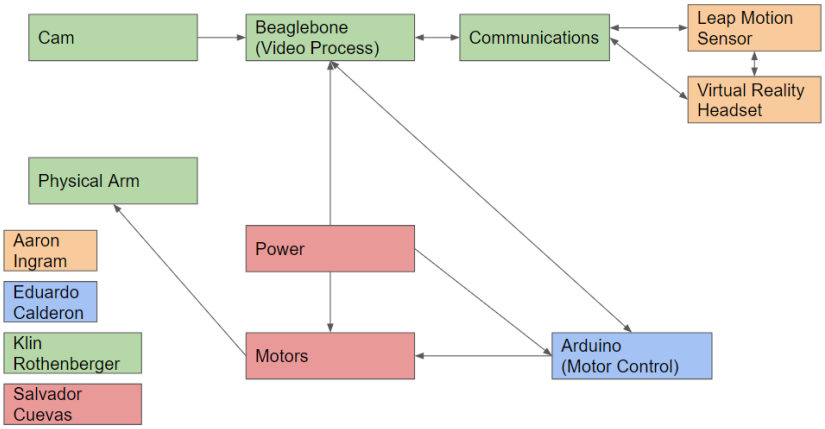


Figure 1. Block Diagram of System

Camera: This component will receive optical data from the stereoscopic camera and send it to the BeagleBone™ microcontroller.

BeagleBone: This microcontroller will receive optical data from the camera and send it to the computer. Additionally, it will receive head and hand position tracking data from the Leap Motion™ Sensor and the VR Headset. It will then send this data to the Arduino for translation.

Communications: This subsystem will account for the data transfer from the camera to the VR Headset. Additionally, it will send the head and hand position tracking data to the BeagleBone™.

Power: This component will manage and supply the electrical power provided to the robotic arm system.

Motors: This component will receive power from the Arduino and actuate the sections of the arm accordingly.

Physical Arm: This component encompasses the physical design of the arm and will move from the motors.

Arduino: This component will translate the head and hand position tracking data from the BeagleBone™ into degrees of movement. It will then move the stepper motors to the desired angle for the robotic arm.

Leap Motion Sensor: This component is physically attached to the VR headset and will track the operator's hands using an infrared camera. Reports hand pitch/roll/yaw, wrist position, and elbow position.

Virtual Reality Headset: This component will display the video feed received from the camera and will track the operator's head position. It will get two separate camera feeds, one for each lens, then stitch them together to form a 3d render. Head orientation will be reported using a pitch roll yaw format

Modes of Operations

The robotic arm will consist of three different modes of operation all dependent on the user: stand-by, tracking, and out-of-range:

- **Standby:** user is not wearing headset and the system is at rest.
- **Tracking:** User activates headset, and microcontroller begins transmitting data from user to robotic arm.
- **Out of Range:** due to the Leap Motion™ controller we are limited to only being able to track at distances in-between 25 to 600 millimeters, anything beyond this will result in improper communication between robot arm and user.

During normal operation (Tracking), the user will be immersed in the environment around the robot arm. The user will be able to identify objects near the robot arm and reach out with their own arm to grab them. The Leap Motion™ will track the movements of the user's arm and send those instructions to the Beaglebone to translate into movement of the stepper motors. The stepper motors will move the arm to mimic the movements of the user's arm, allowing for intuitive control over the robotic arm. If the user decides to look around the environment near the robot arm, the headset will measure the rotational movement of the user's head and send it to the Beaglebone to translate into movement of the gimbal. The gimbal will then roll, pan and tilt the camera to match the motion of the user's head.

Users

The VR Telepresence Robotic Arm is intended to assist users in need of remote presence in situations that may not allow them to be present physically. For instance, our arm would help assist remote medical attention by doctors for when they cannot be there in person either due to distance or danger of infection. Since the Robotic Arm is being developed with the ease of use in mind, we hope many users would be able to control the arm without difficulty. Therefore the user should not need any basic understanding of

the robotic arm components, and should instead just be able to perform the actions of moving their arm or head.

Support

Support for this system will be provided through the form of user manuals and online technical support. Each user manual will provide illustrations of each component along with descriptions on both the hardware and how they each interact with each other. A step by step instruction will be also included in the manual to provide user reference on how the system is connected, beginning with making sure they have the proper software needed to run the system. An online video will be developed demonstrating the system in use, to insure the user the system is performing in the proper manner.

Scenario(s)

Interacting with Hazardous Material

Safely interacting with hazardous material is a task that requires the person to either be in bulky protective gear or to operate a robot to interact with it for them. Our robotic arm would allow for safe interaction with these hazardous materials while still providing the operator with visual information similar to what they would have if they were actually there. Additionally, because it mimics the human arm motion, instructions that would otherwise be difficult to convey using a traditional controller are now easy to perform.

Performing Remote Medical Procedures

Having a doctor available at a moment's notice would be helpful, but currently there are no systems in place. Our arm would allow for a doctor to see everything around the arm, and then to perform a procedure with a full range of motion.

Dangerous or Remote Research Experiments

A scientist or researcher may wish to work on a project where the subject is either too dangerous to be near or located someplace that the researcher can't easily access. It could also allow a researcher to directly participate in multiple experiments that they would otherwise not be able to because they were working elsewhere.

Analysis

Summary of Proposed Improvements

The remote-sensing robotic arm will offer several major improvements over currently existing systems. The first is the ease of use in operating the arm due to the hands-free infrared controls. A user would need minimal training if any in order to use the arm effectively as the system is designed to mimic the natural way of picking up an object. The arm and claw are designed to follow a one-to- one motion of the user's hand and arm so it is extremely intuitive to use.

The second major improvement over existing remote-sensing systems is the immersion provided by the stereoscopic virtual-reality telepresence. Rather than only seeing a flat screen, the user will be able to see depth right next to the object being manipulated. The ability to easily distinguish objects in three dimensions is crucial for fine spatial navigation and only serves to further resemble the actual experience of picking up an object.

Disadvantages and Limitations

The arm currently has several disadvantages in its current implementation. The first of which is the reduced number of axis of motion when compared to the human arm or more advanced robotic arms. The arm is currently limited in its range of motion and will not be able to completely mimic the intricacies of all the joints of a human arm. Furthermore, the

claw being used is drastically simplified compared to the extreme dexterity of a human hand.

The second major drawback in our arm is the current design of the data transmission. Currently, the arm is connected to a computer via an Ethernet cable. Due to the high data transmitting requirements of using two high-definition cameras, an Ethernet cable was the most viable option for transmitting the video stream for this project. The drawback from this choice is that the distance at which the arm can be used from the remote operator is limited entirely by the Ethernet cable length in its current form. This disadvantage also impacts the mobility of the device due to the wired cable.

Alternatives

There are several alternatives for accomplishing the same task that the arm is able to achieve. Current remote sensing arms use a traditional joystick or controller in combination with a monitor in order to manipulate the arm remotely. These robotic arms are able to use less wirelessly transmitted data since the requirements of viewing the robot on a single screen are less intensive than the requirements for 60 FPS, stereoscopic video needed to avoid nausea in a HMD. On the other end of the spectrum, high-end robotic arms that mimic the human arm with all of its joints are considerably more dexterous than our implementation. Our project seeks to show that the novel use of infrared hand tracking and virtual reality observance can be integrated with more complex robotic arms and ultimately be part of a combined system that controls a remote human-resembling robot with one-to-one movement and capabilities.

VR Telepresence Robotic Arm

Aaron Ingram

Salvador Cuevas

Eduardo Calderon

Klin Rothenberger

INTERFACE CONTROL DOCUMENT

REVISION – Draft

25 January 2018

INTERFACE CONTROL DOCUMENT

FOR

VR Telepresence Robotic Arm

PREPARED BY:

_____

Author Date

APPROVED BY:

_____

Project Leader Date

_____

John Lusher II, P.E. Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2/23/2018	Team 8		Draft Release
1	12/6/2018	Team 8		Post-Completion Update

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

No table of figures entries found.

Overview

This ICD document will cover in detail the connections and systems that will be required for the VR telepresence arm. The systems will be broken down into physical, thermal, electrical, and communications. The VR telepresence arm will include a series of stepper motors, a camera, microcontrollers. The user will be using a Virtual Reality headset with a Leap Motion Sensor attached to it. The headset will be strapped onto the user and will be powered by a computer. The headset and sensor will receive information from the user's input and send the data to the microcontrollers on the arm to be processed and translated to move the arm and camera.

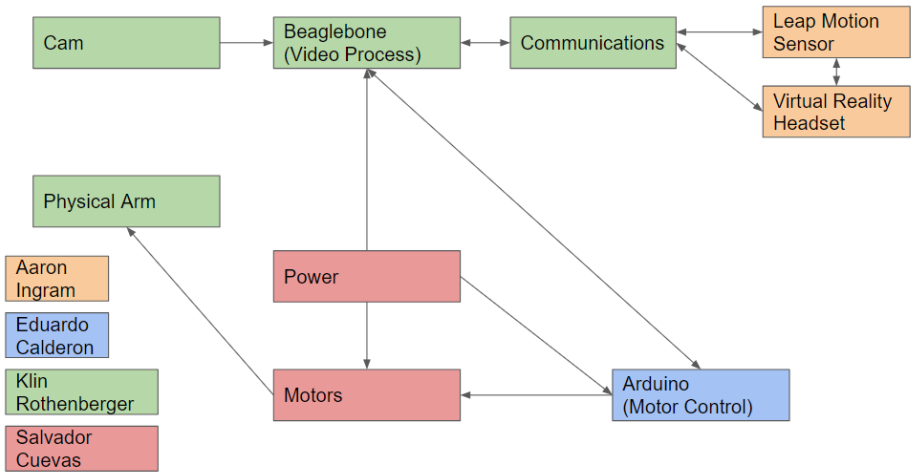


Figure 1. Block Diagram of System

References and Definitions

References

Refer to the Reference Documents section of the Functional System Requirements document.

Definitions

LMC Leap Motion Controller

mA	Milliamp
mW	Milliwatt
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determinet

VRVirtual Reality

W Watts

Physical Interface

Weight

Component Name	Weight	Quantity	Total Weight
Beagle Bone Black	39.68 g	1	39 g
Arduino Due	36 g	1	36 g
Virtual Reality Headset (Oculus Rift)	470 g	1	470 g
Leap Motion Sensor	32 g	1	32 g
Personal Computer	TBD	1	TBD
Stepper Motors	390 g	5	1950 g
Robotic Claw	181 g	1	181 g
Camera	396 g	1	396 g
Camera Servo Motors w mount	40 g	1	40 g
Power Supply	1623 g	1	1623 g
Robotic Arm Total Weight	N/A	1	2642 g

Table 1: Physical Weights

Dimensions

Dimensions of User Space Devices

Component	Length	Width	Height
Virtual Reality Headset	240 mm	190 mm	95 mm
Leap Motion Sensor	75 mm	25 mm	6.2 mm
Computer	800 mm	300 mm	500 mm

Table 2: User Dimensions

Dimensions of Robotic Arm Attachments

Component	Length	Width	Height
Camera	40 mm	80 mm	30 mm
Robot Claw	203 mm	79 mm	38.1 mm

Table 3: Arm Dimensions

Dimensions of motors

Component	Length	Width	Height
Stepper Motors	42 mm	42 mm	48 mm
Arm Servos	60 mm	25 mm	36 mm
Camera Servos	31 mm	36 mm	31 mm

Table 4: Motor Dimensions

Dimension of microcontrollers

Component	Length	Width	Height
Arduino Due	101.52 mm	53.3 mm	N/A
BeagleBone Black	86.40 mm	53.3 mm	N/A

Table 5: Microcontroller Dimensions

Mounting Locations

The Camera will be mounted on top of the robotic arm. This will allow for the user to see the arm as if they were looking down at their own arm. The Microcontrollers will be mounted on the back as specified in the figures below. Each stepper motor is depicted as a joint on the arm with the robot claw joint being depicted in an additional figure to show the mounting for it.

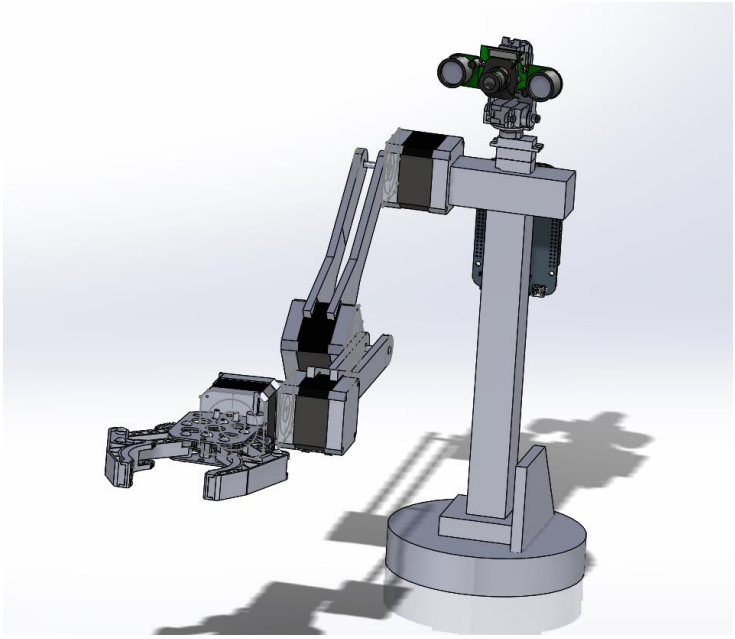


Figure 2. Robotic Arm Mounted with Motors and Camera

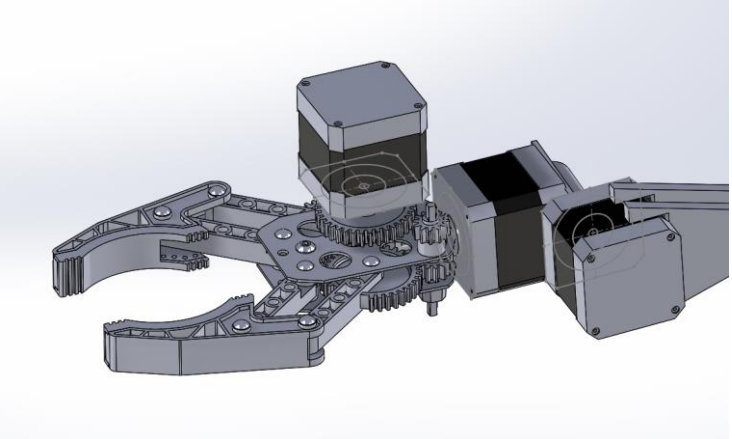


Figure 2. Motor Mount for Robot Claw

Electrical Interface

Primary Input Power

There are two main primary sources of power for the VR telepresence robot.

The first input power will be through the Computer's power supply which will be plugged into an outlet. The power supply will power the headset and Leap Motion through the computer's USB ports.

The second input power source will be an external power supply on the robotic arm. This will power the microcontrollers, motors, and camera to send and receive information from the headset.

Electrical Input for Arm Components

Component	Quantity	Voltage	Current	Power
Arduino Due	1	3.3 V	800 mA	2.64 W
BeagleBone Black	1	5 V	2 A	10 W
Stepper Motor	5	12 V	350 mA	21 W
Servo Motor	TBD	TBD	TBD	TBD

Table 6: Electrical Draw of Components

Camera Feedback to Headset

The video feed from the camera will be sent through USB to the computer. This is to allow the transfer of video quickly to the user's headset. The user will see the camera feed through

the headset and the headset tracking motion will transmit movement information through the beagle board to control the camera servo motors.

Communications / Device Interface Protocols

Communications (Ethernet)

The BeagleBone microcontroller will use ethernet to obtain information from the computer.

Host Device

The VR Headset and Leap Motion will communicate to the computer via USB 3.0 and USB 2.0 ports.

Video Interface

The camera will send the video feed to the computer through USB Cable and the Headset will display the video through an HDMI connection between the headset and computer.

Environment Interface

The robotic arm will be interacting with the physical space around it. This will be done by the user moving their arm and the robotic arm emulating this motion. The arm will be able to physically touch and lift objects while the user will not be physically touching said object.

VR Telepresence Robotic Arm

Aaron Ingram
Salvador Cuevas
Eduardo Calderon
Klin Rothenberger

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – Draft

25 January 2018

FUNCTIONAL SYSTEM REQUIREMENTS

FOR

VR Telepresence Robotic Arm

PREPARED BY:

_____

Author Date

APPROVED BY:

_____

Project Leader Date

_____

John Lusher, P.E. Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2/21/2018	Team 8		Draft Release
1	2/5/2018	Team 8		1st Semester Final Draft
2	12/6/2018	Team 8		Post-Completion Update

1. Table of Contents

Commented [GU2]: should some page numbers be bolded?

2. List of Tables

No table of figures entries found.

3. List of Figures

4. Introduction

4.1. Purpose and Scope

The Virtual Reality Controller Robotic Arm is a solution for those who want the ability to control a robot arm by moving their own arm as well as the ability to see the environment in which the robot is located. The arm is intended to be a system than can be mounted on various surfaces so that it can function as an additional piece. For demonstration purposes, the robotic arm will be a scaled down version. The arm and camera will be controlled through a series of servos and microcontrollers and will communicate with the user through an ethernet connection. The user will be wearing a headset and infrared sensor to allow for data of movement to be sent to the microcontrollers so that the arm will move accordingly. The arm will be powered independently on its own whereas the headset will be powered through USB connection to a computer.

Figure 1. Conceptual Image for Robotic Arm

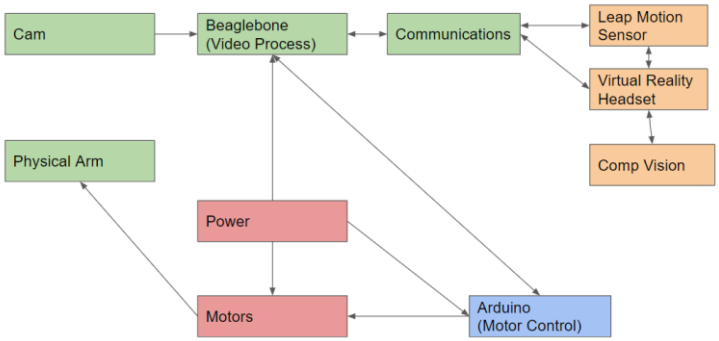


Figure 2. Subsystem Overview

The following definitions differentiate between requirements and other statements.

Shall:	This is the only verb used for the binding requirements.
Should/May:	These verbs are used for stating non-mandatory goals.
Will:	This verb is used for stating facts or declaration of purpose.

4.2. Responsibility and Change Authority

Eduardo Calderon will ensure that the requirements for the project are met. The requirements can only be changed with the approval of the team leader and Pranav.

Sub-System	Responsibility
Power and Motors	Salvador Cuevas
Arm and Camera Communications	Klin Rothenberger
Motor Control	Eduardo Calderon
Virtual Reality Headset and Camera Vision	Aaron Ingram

Table 1: Table of Subsystems

5. Applicable and Reference Documents

5.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
DOD-HDBK-791	3/17/1998	IEEE Standard for Ethernet

5.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
310-30000-02	2017	**Oculus Best Practices**

5.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

6. Requirements

This section defines the minimum requirements that the development item(s) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

6.1. System Definition

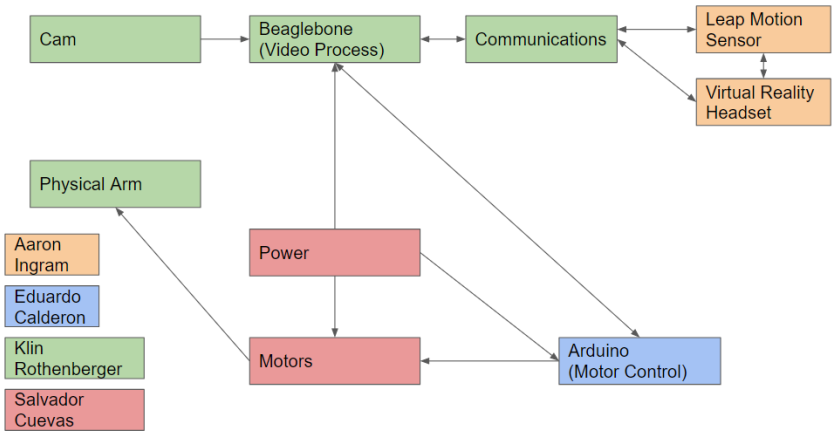


Figure 2. Block Diagram of System

The input will be comprised of the Virtual Reality Headset and Leap Motion which is powered through a computer via USB port. This subsystem will track the user's arm movement and show the user the video feed from the camera. Afterwards, the user's arm movement will be transmitted over an ethernet connection and then translated in the microcontroller for movement instructions. From there the arm's servo motors will move alongside the user's arm to replicate movement on the robotic arm. Additionally, the robot arm will feature a camera system to allow the user to view the environment in which the arm is located. This will be done by using a camera and processing it through a microcontroller to then send the information to the headset.

6.2. Characteristics

6.2.1. Functional / Performance Requirements

6.2.1.1. Maximum Latency

The maximum latency of the system shall be no more than 200 milliseconds.

Rationale: Any latency more than this will jar the user's experience and make it difficult for them to use the headset without getting motion sick from the delayed movements.

6.2.1.2. Power Supply Lifespan

Bench Power Supply used will supply power through wall outlet.

Rationale: Ensure robot arm is consistently receiving power.

6.2.1.3. Lifting Capabilities

The arm shall lift an item no more than 250 grams.

Rationale: With the scaled down version of the robotic arm, it should be able to pick object with this mass. At this mass, there should be no jerky movements from the motors being unable to support the torque needed.

6.2.2. Physical Characteristics

6.2.2.1. Mass

The mass of the Robotic Arm system shall be less than or equal to 10 kilograms.

Rationale: The robotic arm was calculated to be 2.6 kg without adding in the metal bar components to act as the skeleton. This gives a leeway of 7.4 kg to add onto the arm to complete the skeleton with estimations of 6.8 kg as the weight of the metal bar components.

6.2.2.2. Physical Dimension

The area that the robotic arm takes up shall be no more than 30 inches in length, 5 inches in width, 15 inches in height.

Rationale: The Arm was designed in AutoCAD, and the maximum length and height was found through extension of the arm. The width was also determined through the AutoCAD mockup.

6.2.2.3. Volume of External Power Supply

The volume of the bench power supply used in VR Telepresence System is equal to 3 inches in height, 4.9 inches in width, and 7.5 inches in length.

Rationale: These are the dimensions of the power supply we will be utilizing for the arm.

6.2.3. Electrical Characteristics

6.2.3.1. Inputs

a. The VR Telepresence with Arm system is designed to only take inputs provided by the user that is then read by the leap motion controller. Input data sent by the Leap Motion Controller will consist of digital signals governed by the data sheet provided by the given controller. A microcontroller will also be integrated in system to translate LMC data into a usable form of data to control robotic arm.
Rationale: By design, should limit the chance of damage or malfunction by user/technician error.

6.2.3.1.1 POWER CONSUMPTION

a. The maximum peak power for the whole system shall not exceed the overall power from each individual component in the system. microcontrollers, stepper motors, and camera.

Rationale: This is so that there is no damage to any of the devices by using too much power than the recommended specifications.

6.2.3.1.2 INPUT VOLTAGE LEVEL

The input voltage level for the Arm component of the system shall be between +8 VDC to +12 VDC. While the microcontrollers will be supplied with a voltage supply of 3.3V-5V.

Rationale: Due to the stepper Motor and Arduino recommended rating in the datasheet.

6.2.3.1.3 COMPUTER SPECIFICATIONS

The computer will need to have at least the minimum specifications denoted in the table.

Graphics Card	**NVIDIA GTX 1050 Ti / AMD Radeon RX 470 or greater**
Alternative Graphics Card	NVIDIA GTX 960 4GB / AMD Radeon R9 290 or greater
CPU	Intel i3-6100 / AMD Ryzen 3 1200, FX4350 or greater
Memory	8GB+ RAM
Video Output	Compatible HDMI 1.3 video output
USB Ports	1x USB 3.0 port, plus 2x USB 2.0 ports
OS	Windows 8.1 or newer

Rationale: Due to the VR headset's needed processing power, this is the minimum specifications needed according to the Oculus Rift's Documentation in order to be able to use the VR Headset.

6.2.3.2. Outputs

6.2.3.2.1 RAW VIDEO OUTPUT

The Telepresence arm will output a live video stream to the VR headset worn by the operator

Rationale: For the operator to be able to see what the arm is doing in a remote location they need a live video feed to the interface device.

6.2.4. Environmental Requirements

The VR Telepresence with Arm System shall be designed to withstand an operate in the environments specified in the following section.

Rationale: This is a requirement specified by our customer due to constraints of their system in which the VR Telepresence Arm is integrating.

6.2.4.1. Pressure (Altitude)

The VR Telepresence Arm System, will be tested and verified operate at an altitude of 328 ft., the elevation of College Station

Rationale: The arm will be tested and modified in the location of College Station and therefore will have the specifications to perform at this altitude.

6.2.4.2. Thermal

The Arm component of the system will operate between 10°C – 40°C.

Rationale: With an annual range of 16.1°C - 35.7°C in College Station, the robotic arm should be able to perform within the stated temperatures due to the area of performance.

6.2.4.3. Humidity

The system operates in low to moderate humidity conditions.

Rationale: Due to the integrated components of the system, we must take into account for which the specs are specified.

Appendix A: Acronyms and Abbreviations

CCA	Circuit Card Assembly
CPU	Computer Processing Unit
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EO/IR	Electro-optical Infrared
FOR	Field of Regard
FOV	Field of View
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
KHz	Kilohertz (1,000 Hz)
LCD	Liquid Crystal Display
LED	Light-emitting Diode
LMC	Leap Motion Controller
mA	Milliamp
MCU	Microcontroller
MHz	Megahertz (1,000,000 Hz)
mW	Milliwatt
OS	Operating System
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
VME	VERSA-Module Europe
VR	Virtual Reality

VR Telepresence Robotic Arm

Aaron Ingram

Salvador Cuevas

Eduardo Calderon

Klin Rothenberger

SCHEDULE AND VALIDATION

REVISION – 3

6 Dec 2018

Schedule:

Task	Date Expected	Person	Status	Date Complete
Concept of Operations	2/15	All	Completed	2/15
Functional Systems of Requirements	2/21	All	Completed	
Interface Control Document	2/21	All	Completed	
Project Parts Ordered	2/21	All	Completed	
Midterm Presentation	2/22	All	Completed	
Progress Update #1	3/8	All		
The leap motion sensor will be able to track and report hand positions while it is stationed on a desk	3/8	Aaron	Completed	3/8
Set up Arduino Due to control a single motor	3/8	Eduardo	Completed	3/7
Show progress on the solidworks model (50% complete)	3/8	Klin	Completed	3/6
Develop the external power supply for the entire system	3/8	Salvador		
Wire the Arduino to the Beagle bone to allow the transmission of data between the microcontrollers	3/15	Eduardo	Completed (Late)	4/16
Progress Update #2	3/22	All		
Receive head tracking using the oculus rift positional system and send positional data to camera motors over Ethernet		Aaron	Completed	3/26
Get hand tracking information from the computer fed through the beagle bone	3/22	Eduardo	Completed (Late)	4/16
Finish solidworks model, Show MCU receiving camera data.	3/22	Klin	Completed	3/22
Establish power to the motors and the camera	3/22	Salvador		
Transform hand tracking information into movement for motors	3/29	Eduardo	Completed	3/28
Progress Update #3	4/5	All		
The Oculus Rift headset will display a live video feed that come into the computer via USB port	4/5	Aaron	Completed (late)	4/25
Use the Arduino to control the full set of motors on the arm	4/5	Eduardo	Completed (Early)	3/28

Show the Beaglebone Black controlling each servo individually	4/5	Klin	Completed	4/27
Ensure that the motors are receiving proper power based on the microcontrollers	4/5	Salvador		
Progress Update #4	4/19	All		
Fully integrate head and hand tracking with the VR headset and send to MCUs. Model arm position based off hand position	4/19	Aaron	Partially Completed	4/25
Handle jittery movement of the arm and take care of movement issues and bugs	4/19	Eduardo	Completed	4/25
Improve the framerate received by the camera to the computer	4/19	Klin	Incomplete	N/A
Make final adjustments to ensure that power is constant throughout the entire system	4/19	Salvador		
Finish Final Presentation Slides	4/20	All	Completed	4/22
Final Presentation	4/23	All	Completed	4/23
Finish Subsystems for the Demonstration	4/26	All	Completed	4/30
Project Subsystem Demo	4/30	All	Completed	4/30
Finish Final Report	5/1	All	In-progress	5/3
Final Report Submission	5/3	All	In-progress	5/3

Validation Plan:

Task	Specification	Results	Person
Robot arm should move with precision given test input	<5°	Expected: 45° Actual: 44.6 ° Within <5° Specification Error: 0.88%	Eduardo
Serial Communication for BeagleBone Black and Arduino Due	<100ms	Sent 64-byte String Actual: 67 ms Within <100 ms specification	Eduardo
Lift and maintain control of weighted object	>250g		Salvador
Total weight of the TARMU system	<10kg	System (minus the computer) weighs 16lbs = 7.27 kg	Klin
Translate head position to gimbal movement	<100ms	Gimbal moved camera ~900ms after reciving instructions	Klin
Display Video Feed to Rift	>=20FPS	Scene rendered at ~80 FPS Camera feed loaded at 20FPS	Aaron
Send updated head position	<100ms	Head position was calculated once per render Actual ~12.5ms Within <100ms Range	Aaron

Send updated hand position	<100ms	Actual 20ms Within <100ms	Aaron

VR Telepresence Robotic Arm

Aaron Ingram

Salvador Cuevas

Eduardo Calderon

Klin Rothenberger

SUBSYSTEM REPORTS

REVISION – 1

2 May 2018

Table of Contents

Ignore the messed-up format for now

Virtual Reality Headset / Hand Tracking Subsystem

2.1 Subsystem Introduction

The VR Headset and Hand Tracking subsystem has been developed to interface between the human operator and the robotic components. The subsystem is comprised of the Oculus Rift VR Headset, the Oculus Rift motion tracking sensors, and the LeapMotion Hand position sensor. The Oculus Rift Headset was tested for head orientation tracking and video feed display. The LeapMotion sensor was tested for hand and arm positioning. The consistency of the systems was tested by performing the same motion repeatedly to see if the values were consistent.

2.2 Subsystem Details

The Oculus Rift Headset was used because it is one of two high quality commercially available headset, and one of our group members already owned it before the project began. We used three tracking sensors because it allows for more accurate head tracking when the headset isn't pointed at its front orientation.

Power and connections for the headset are handled by a VR capable desktop computer. This is because the Headset has strict connection requirements for data transmission and doesn't have a dedicated power line so integration with the power subsystem wasn't feasible.

The LeapMotion sensor was chosen because it is a commercially available sensor that handles hand tracking using an IR camera. For all testing and data collection up to this point it has been mounted on the front of the headset. It also is powered by the computer because its only power/data cable is USB.

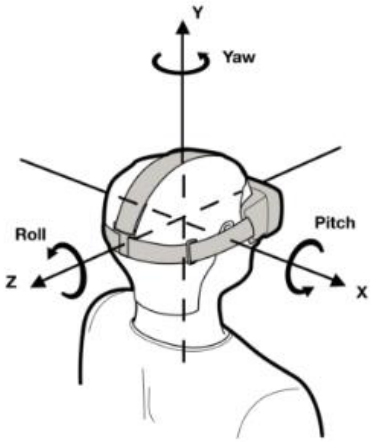
For Integration, All the components will be connected to a desktop computer, which will be tethered to the BeagleBone Black using an ethernet connection.

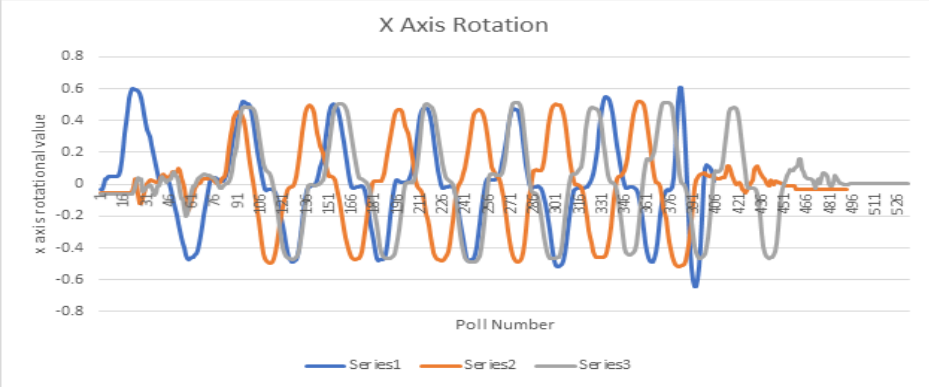
2.3 Subsystem Validation

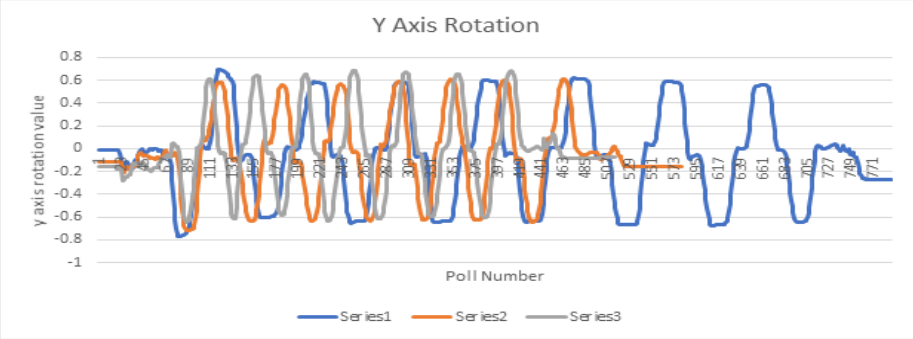
For Validation of the Subsystem there were three main tasks to achieve. The first task was simply to ensure the headset could pass a video feed for the user to view. This was simply validated by showing that the image was displayed for each of the lenses of the headset. The left picture shows the image displayed for the left lens and the right for the right lens.

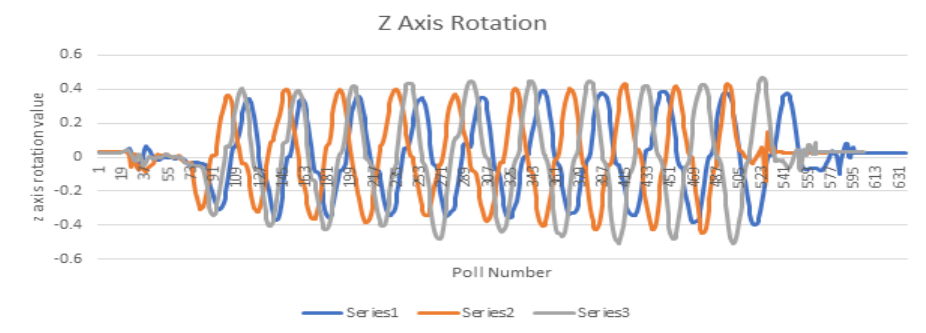
The second validation was that the headset positioning was consistent across multiple trials of the same motions.

For each of the three rotational axis I conducted multiple tests by repeatedly performing the same motion. Each test was designed to the axis for normal head movements. The rotational value is on a scale from -1 to 1.




X Axis Rotation
x axis rotational value
Poll Number
Series1
Series2
Series3

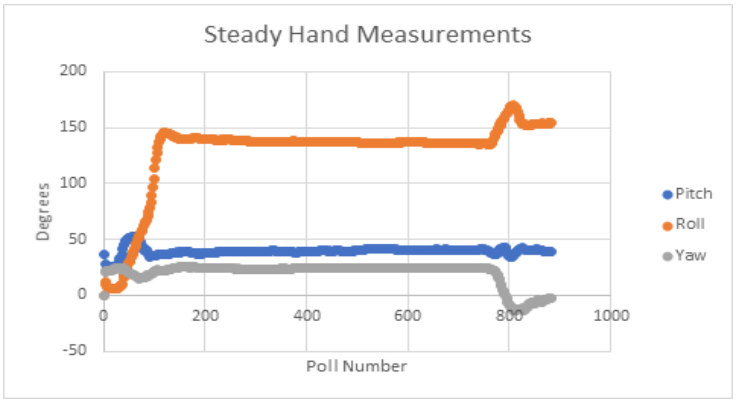

Y Axis Rotation
y axis rotation value
Poll Number
Series1
Series2
Series3

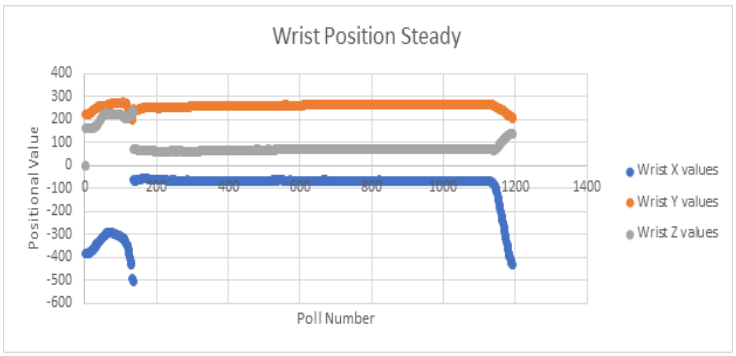

Z Axis Rotation
z axis rotation value
Poll Number
Series1
Series2
Series3

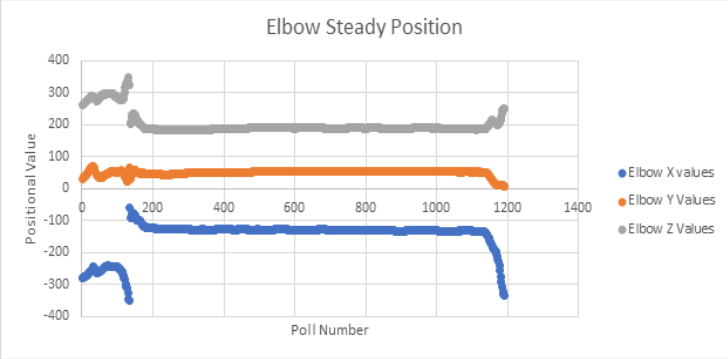
The graphs show that for each run the maximum and minimum rotational values are very close to one another. This consistency means we should be able to reliably use the positional data to control the camera gimbal.

The last major validation test was for the LeapMotion sensor. For these tests a specific gesture was repeatedly performed similar to the headset orientation test. For this sensor we also tested the resting data for when a hand isn't moving but the headset is.

When a hand is held still without moving the headset it yield the following graphs:

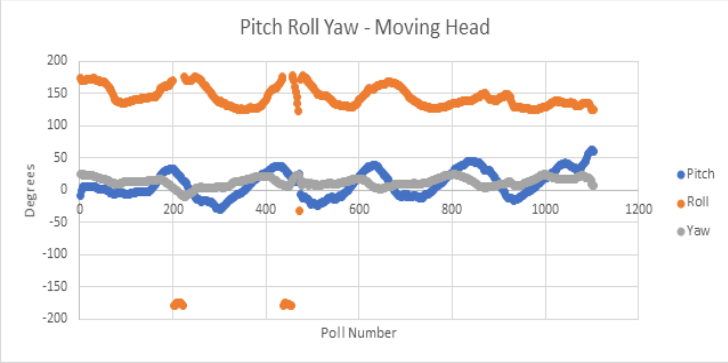


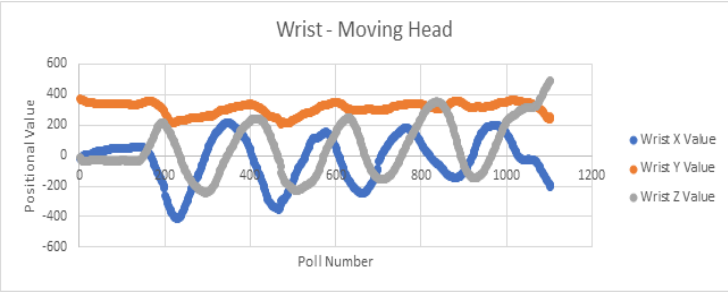


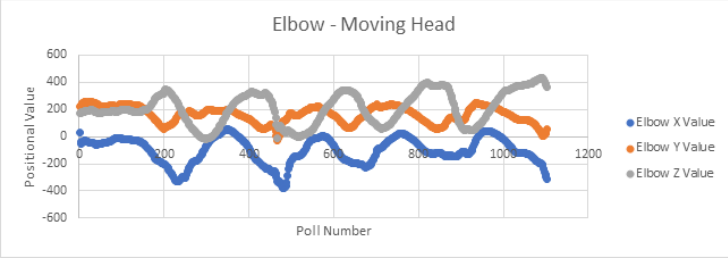


From theses graphs we can see that the leap motion is consistent in keeping a steady value when both the sensor and hand are not moving

Next, I tested to see how consistent the values were when the hand was stationary but the headset was moving.



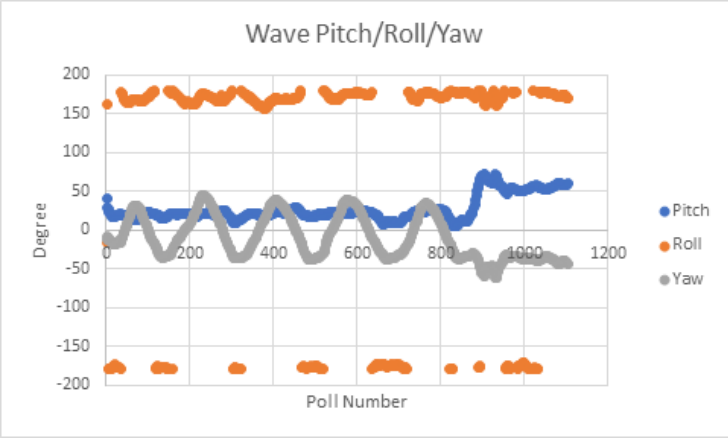


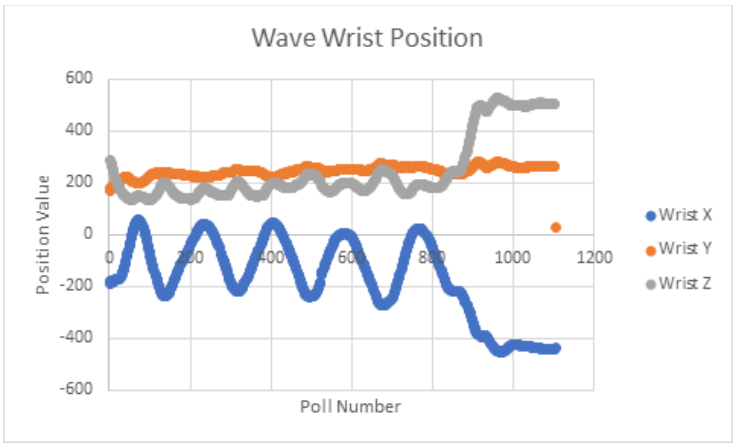


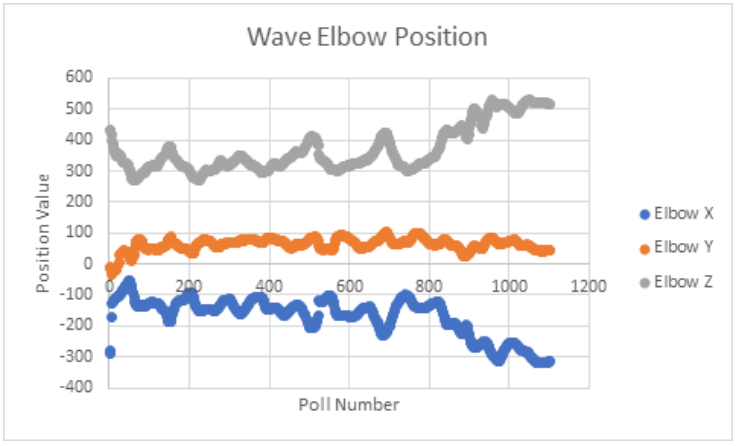
These graphs paint a very different picture. Here it is visible that the raw values fluctuate a good deal when the headset/sensor move but the hand does not. This is especially prevalent in the wrist position, which is a big factor in our arm.

To test gesture consistency, I repeatedly performed the same action. The headset was stationary for these tests.

Side to Side Wave:

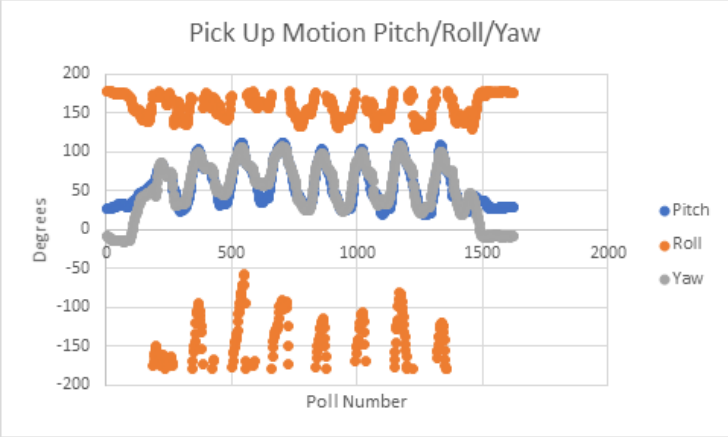


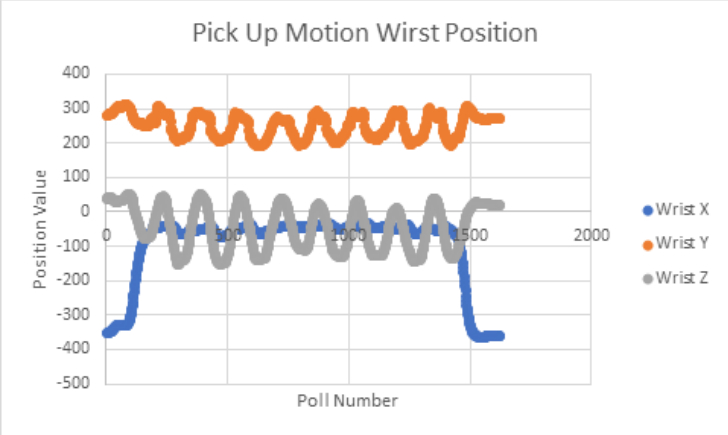


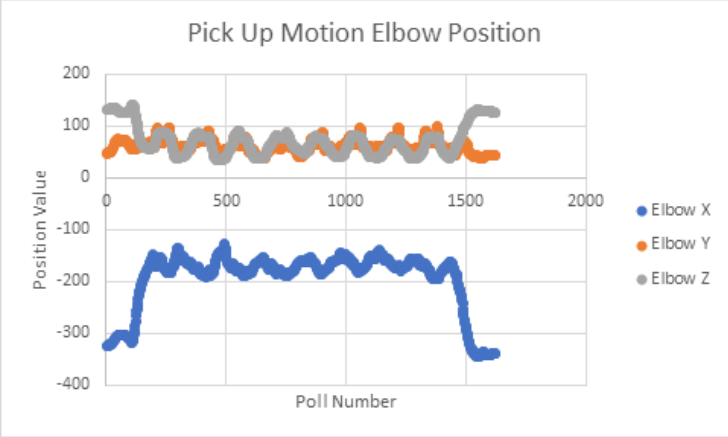


These Graphs are consistent with what you would expect from a wave motion. The hand pitch and roll are fairly constant, though the roll does seem to be on a measurement wrap around edge. The yaw oscillates with the wave. The elbow position is mostly, but not perfectly steady. The wrist y and z positions are stable and the x oscillates with the wave.

The other Gesture I tested was the motion of picking something up. Once again, the headset/sensor will be stationary so only the hand is moving.


Pick Up Motion Pitch/Roll/Yaw
Degrees
Poll Number
Pitch
Roll
Yaw


Pick Up Motion Wirst Position
Position Value
Poll Number
Wrist X
Wrist Y
Wrist Z



These motion graphs are consistent with the pickup motion performed. The primary motion was lifting the wrist up and changing the angle of it. This is demonstrated will by the data, because the elbow is stable, as is the x position of the wrist.

2.4 ECEN 403 Subsystem Validation Conclusion

The subsystem components were mostly successful in validation for their functionality. The primary function of the VR Headset/Hand tracking subsystem is to Report the location and orientation of the head and hand, while streaming video to the operator. Two of these functions were validated without any issue. The video was streamed as needed at a rate where the operator could process his/her surroundings. The Headset accurately reported its orientation through multiple cycles of the test. The LeapMotion controller was only partially successfully verified. While it behaved correctly and as expected while the headset/sensor was stationary, the moving sensor test shows that the values aren't consistent while the operator is moving their head. As this is a common action for our project this poses a problem. There are two possible solutions for the issue. The motion sensor can be mounted on the desk rather than on the headset, or I could try to implement a correction algorithm that adjusts for the moving headset. As we have already shown that the sensor works correctly when stationary it seems most reasonable to me to simply mount the sensor on the desk.

2.5 ECEN 404 Subsystem Accomplishments

Tasks Accomplished:

- Full video passthrough from camera to headset
- Stabilized video feed in headset so head orientation doesn't affect user's ability to see video feed

- Combined LeapMotion functionality, headset rendering, and headset position tracking into one program
- Achieved hand and head tracking and value transformation to send useful motor values to microcontrollers instead of raw positions

2.6 Conclusion

The virtual reality headset and hand tracking subsystem acted as the user input and output for the project. The subsystem was completed and was successfully integrated with the other subsystems to form multiple communication pipelines. The functionality of this subsystem was based around two primary pieces of hardware and communicating values to and from them. The Oculus Rift headset served as the output to the user. It displayed the camera feed to the user in near real time. It also served as an input to the system. It delivered the head pitch roll and yaw to the camera gimbal so that it could mirror the user's head motions. This was mostly successful, as there was a case when the Oculus Rift would invert the yaw value without any clear warning or trigger action. To counter this we added a toggle switch on the keyboard. While this solution worked it was less desirable than it automatically working without any user input. The LeapMotion sensor served as our user input. Functionality for this system was based on taking the user's hand and generating values. First the sensor would report raw values like the hand normal, wrist distance from the sensor, and distance between fingers. We would then take the hard sensor data and turn it into motor controls. The sensor was able to accurately report values associated with the hand and wrist but struggled to get the accurate value for elbow and beyond. Future modifications for this subsystem would be to add more sensors in addition to the LeapMotion. While it does a good job at capturing hand data if we wanted to implement larger arm control it would do a poor job of that. Another improvement would be to switch to the HTC Vive for the headset used by the operator. Due to changes made to the SDK by the owners of oculus it is no longer easily possible to achieve stereoscopic video passthrough, while on the Vive this is still achievable. By creating and implementing this system the developer gained knowledge about 3D rendering and scene development using OpenGL, TCP/UDP server programming in C and C++, using algorithms to generate motor controls from raw data, and team communication.

Arm Control Microcontroller Subsystem

2.1 Subsystem Introduction

The microcontroller subsystem has been developed to control stepper motors to recreate the movement of the human arm. The subsystem is comprised of 5 stepper motors and their respective micro step drivers and an Arduino Due. The Arduino Due was tested for controlling these 5 stepper motors, and to accept incoming data from the Beagle Bone Black microcontroller over serial communication. The subsystem was tested for accuracy of stepper rotation and the transmission speed between Beagle Bone Black and the Arduino Due.

2.2 Subsystem Details

The Arduino Due was chosen to control the motors due to its Arm Cortex-M3 which allows for powerful and sophisticated operations. It's 12 PWM pins with its multitude of digital pins allow for easy and precise control of the stepper motors.

The Arduino Due can be powered by a 5 – 12 Volt DC power supply which will be provided by the Power Subsystem. Each Micro step Driver can be powered by a 9 – 42 Volt DC power supply which will also be provided by the Power Subsystem.

This subsystem will be responsible for controlling each stepper motor to rotate to a desired position to emulate the human arm. Thus, the Arduino Due will contain all the logic necessary to provide the control. The code was tested and validated for the criteria specified.

For Integration, each component will be mounted to the Robot Arm. Each stepper Motor will be mounted as a joint of an arm with each Micro Step Driver being mounted to a base with wires attaching to each respective motor. The Arduino Due will also be mounted on the base with wires going to the Micro Step Driver to control the Stepper Motors and wires going to the Beagle Bone Black to establish Serial Communication.

2.3 Subsystem Validation

For Validation of the Subsystem there were two main tasks to achieve. The first task comprised of moving all five stepper motors to a desired rotation angle and then evaluate the motor's accuracy by comparing the actual angle of the final movement to the theorized angle.

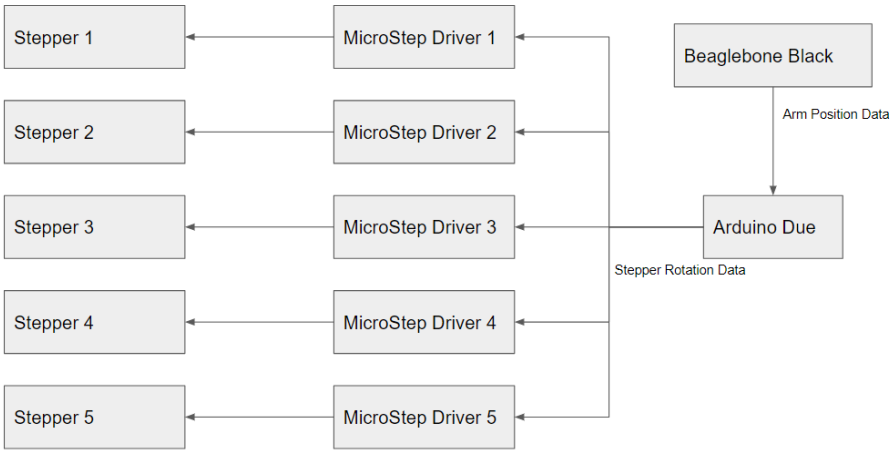


FIGURE X: FLOW DIAGRAM OF SUBSYSTEM

The flowchart above shows the connections within the microcontroller subsystem. The Arduino Due has a connection with the Beagle Bone Black that will allow the two microcontrollers to communicate with each other. From here the Arduino Due can process the information given by the Beagle Bone Black and then output stepper motor commands. These commands by the Arduino Due come in the form of PWM (Pulse Width Modulation) waveforms and will be the key component in generating the rotation for the stepper motors. The Micro Step Drivers exist to allow more power to be safely driven to the Stepper motor as well as allow for more precise stepping with the motor. These Drivers change the step size of the motor from 200 to 1600 to allow for an angle of .225° for each step as opposed to the default 1.8° for each step.

Each stepper motor designates a different part of the arm to move, and thus the code will have to break down the information coming from the Beagle Bone Black to each stepper motor. Since Arduino Due can only do one task at a time, the code will have to allow for positions to be updated and allow each motor to move to the place before re-updating the position and repeating the process.

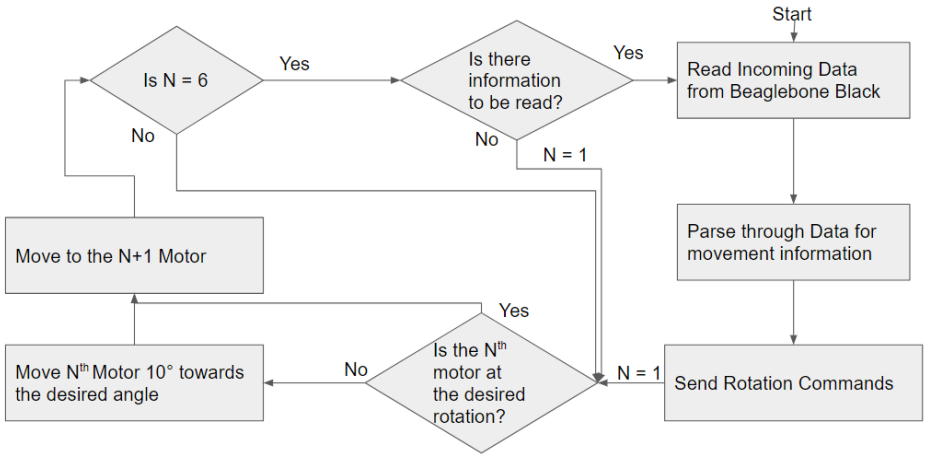


FIGURE X: FLOW DIAGRAM OF MICROCONTROLLER CODE

The flowchart above shows the methodology of the stepper control. Since the Arduino Due can only process one task at a time, we must command each motor sequentially. We define an order for each motor and then sequentially move through each motor rotating at most 10 degrees to desired rotation. We choose 10 degrees to accommodate for the speed of motor vs the speed of updated data from the Beagle Bone Black. Once we hit the last motor, then we can check if there is an updated position of the arm. And then repeat the process of parsing through new data or moving the motors to the desired position of the previous data input.

The validation for the stepper motor was done by rotating the stepper motors a certain amount of degrees and checking how close the actual result was. The method of testing was to spin the motor a full 360° multiple times before finally rotating a specified amount. Then we can observe the final results with the expected to validate the accuracy and movement of the stepper motors. The system was also tested while being powered by the external power supply given by the power subsystem which yielded positive results for the future when integration will occur next semester.

The second task was to establish Serial Communication between the Beagle Bone Black and the Arduino Due. The table below shows the reserved pins for communication with the beagle bone black and stepper motor control.

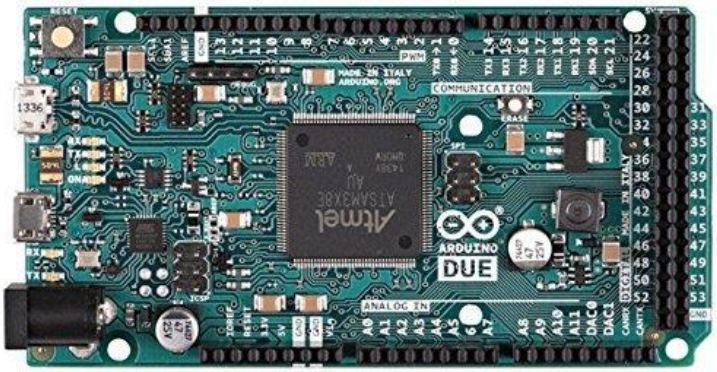


FIGURE X: ARDUINO DUE

TABLE X: ARDUINO DUE PINOUT SCHEME

Arduino Due Pins	Functionality
0	UART RX to Beagle Bone Black
1	UART TX to Beagle Bone Black
GND	Ground for Beagle Bone Black
2, 3, 4, 5	Stepper 1, Micro Step Driver 1
6, 7, 8, 9	Stepper 2, Micro Step Driver 2
10, 11, 12, 13	Stepper 3, Micro Step Driver 3
22, 24, 26, 28	Stepper 4, Micro Step Driver 4
23, 25, 27, 29	Stepper 5, Micro Step Driver 5

The Serial Communication was established by utilizing Arduino Pins 0, 1 and GND. These Wires went to the Beagle Bone Black's respective pins 24, 26, 2. This established TX -> RX, RX -> TX, and GND <-> GND connections to allow for serial communication.

The validation for Serial Communcation was performed by printing statements on Arduino Due and observing the statements on the Beagle Bone Black's terminal. We can measure the time through recording videos of the screen and finding out what frames the message appears from being sent to being received. To measure more precisely we either need a camera that can record more frames per second, or an additional device that can detect the sending and receiving events of each device.

These two functions were validated separately and integration of beagle bone communication to stepper motor movement will be developed in the next semester when all subsystems will be integrated together. Integration will involve combining these statements that are coming from the Beagle Bone Black with the Stepper Motor Control Program.

2.4 ECEN 404 Changes

During the semester of ECEN 404, major changes occurred for the Motor Control System. These changes were implemented to make integration simpler. For instance, we changed the communication microcontroller, the Beagle Bone Black, to the Odroid XU4 for reasons I will explain below. Additionally, the Arduino Due that we utilized had to be replaced with an Arduino Mega 2560.

2.4.1 Communication with the Odroid XU4

We made the change from the Beagle Bone Black to the Odroid XU4 so we could better implement video passthrough. The stereoscopic camera that we were utilizing required USB 3.0 to have enough data transfer for both high definition images to be sent. There was no USB 3.0 option on the Beagle Bone Black, and video processing was difficult to implement at a level we were satisfied with. With the Odroid XU4, we were able to utilize USB 3.0 and the appropriate video transmission libraries to allow for stable video to be captured and sent over a TCP Server. This allowed us to maintain better video quality on the origin's user's end through the VR headset.

When we utilized the Beagle Bone Black and Arduino Due in the last semester, serial communication required little externals because both pins ran at the same voltage, 3.3V. Thus we ran into an issue when we upgraded to the Odroid XU4, because the pin voltages of the Odroid are 1.8V. This meant that if we were to directly connect them in a similar fashion from before, we risk shorting the boards. To solve this, we obtained the Odroid XU4 shifter shield. This shifter shield allows for the Odroid pins to be set to accept 1.8V, 3.3V, and even 5V. This allowed for us to connect the two microcontrollers into serial again.

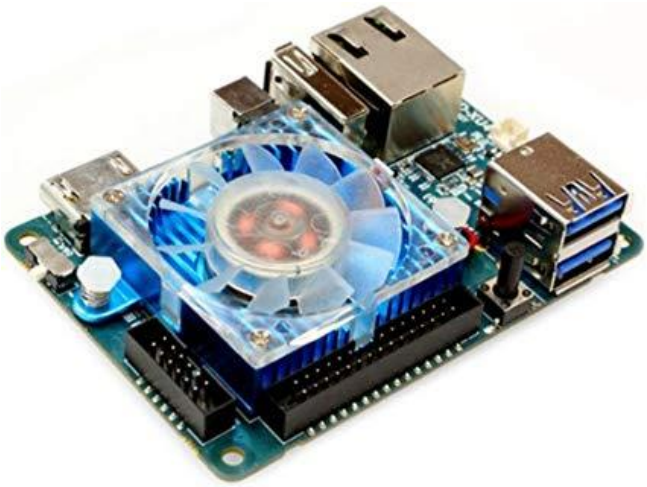


FIGURE X: ODROID XU4

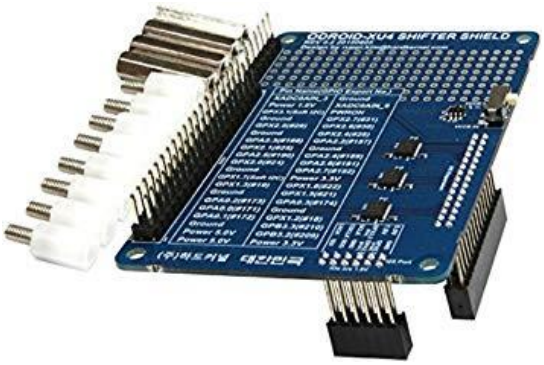


FIGURE X: ODROID XU4 SHIFTER SHIELD

2.4.2 Motor Control with the Arduino Mega 2560

Motor Control for the last semester involved using the Arduino Due to control the motors, but this semester we had to switch over to the Arduino Mega to do our control. There is an issue with the Arduino Due's hardware that can stop the Due from connecting to a device. This became apparent later in the semester when we were attempting to upload code to run motors, but the Arduino Due became unresponsive and there were no solutions we found that solved our issue. Thus, we swapped to the Arduino Mega 2560, which featured the exact pins as the Arduino Due. This allowed us to continue to proceed with our motor control.

Lastly, even though the pins are the exact same as the Due, there exists one important hardware difference we'd like to say. The pin voltage on the Due is 3.3V whereas the pin voltage on the Mega is 5V. Now, if we were attempting to do serial communication, we would need to make sure that the other device, the Odroid XU4, is also running at 5V. Thankfully, even though the Odroid XU4 is running at 3.3V on their pins, the Shifter Shield allows us to switch the Odroid to accept and operate at 5V pin voltage.

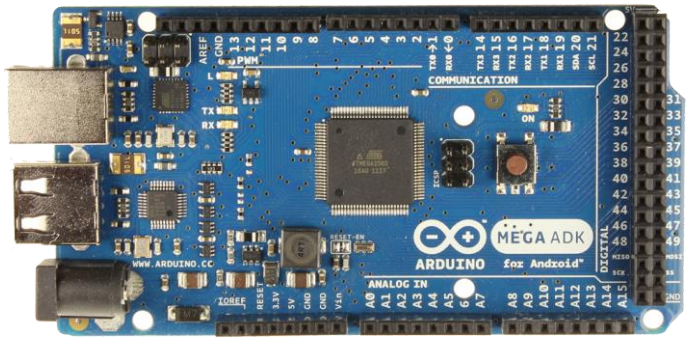


FIGURE X: ARDUINO MEGA 2560

2.4.3 Camera Gimbal Servo Control

Last semester's plan was to utilize the Beagle Bone Black's Pins to drive the servo motors on the Camera Gimbal. This would be possible because the Beagle Bone Black is capable of driving PWM signals to move the servo motors. However, we have replaced the Beagle Bone Black with the Odroid XU4 in the project, which has no PWM Pins to drive the servo motors like we had planned last semester.

This means that to control the Gimbal, we pass the values over to the Arduino Mega with the other motor controls as opposed to using the Odroid for it.

2.4.4 Update to Tables and Diagrams

With the changes to the microcontrollers, as well as the changes to the motors we are utilized on the arm, we found it necessary to display the new pinout scheme of the Arduino Mega

TABLE X: ARDUINO MEGA PINOUT SCHEME

Arduino Mega Pins	Functionality
19	UART RX to Odroid XU4
18	UART TX to Odroid XU4
GND	Ground for Odroid XU4
22, 24, 26, 28	Wrist Pitch Stepper Motor
9, 10, 11, 12	Wrist Roll Stepper Motor
4, 5, 6, 7	Elbow Rotation Stepper Motor
8	Pinch Servo Motor
3	Camera Gimbal Lateral Movement
13	Camerea Gimbal Vertical Movement

2.4.5 Overall Accomplishments

In the semester of ECEN 404 we accomplished

- Video transmission from the Odroid in the form of a UDP stream. The Odroid sent UDP packets of compressed images, as a client, to the origin user, who was acting as a server.
- Non-Blocking Serial Tokenization. The Arduino has a method to read in serial integer values, but these methods require time to process, so we utilized a more efficient method to get the values to move motors.
- TCP Server/Client between the Odroid and the Origin User. We connected the Computer and Odroid in TCP so that we could reliably send motor commands over the network, and then relay that information over serial to the Arduino Mega.

2.5 Subsystem Conclusion

The subsystem was successfully validated for their functionality. The primary function of the microcontroller subsystem is to rotate the stepper motors accurately and communicate with the beagle bone black to get updates of position information. The Arduino Due was successful in generating functionality of stepper motors and establishing communication with beagle bone black. The subsystem has also proved to be able to integrate well with the power system and the camera/gimbal subsystem which contains the Beagle Bone Black. The success of integration with the other subsystems is important for generating arm movement in the future of the project.

Camera/Gimbal Subsystem

3.1 Subsystem Introduction

The camera/gimbal subsystem was responsible for sending a video stream to the headset to provide useful visual information to the user as well as translating the head motions of the user into gimbal movement. This was done by encoding a raw video stream from the microcontroller and sending it to the headset over a wired connection. Additionally, whenever the user turned their head in a different direction, the headset detected the change and sent the updated rotational values along the same cable. Here the microcontroller simultaneously translated these rotational instructions into movements for the 3-axis gimbal and powered the servos to move accordingly.

3.2 Subsystem Details

This subsystem used a Beaglebone Black as the microcontroller to encode and translate the video stream and gyroscope instructions. The Beaglebone Black was chosen because it had the I/O necessary to allow for simultaneous transfer of high-bandwidth data, as well as pins necessary to control the gimbal.

The Beaglebone Black may be powered from a 5 Volt DC power supply or from power provided by a usb connection. The three servos that comprise the gimbal also require a 5V power supply. The camera may only be powered from its usb connection to the Beaglebone Black. All three components are powered from the power subsystem.

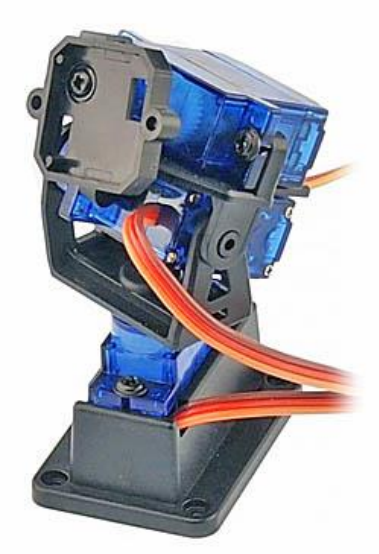


FIGURE X: MAGNFIED IMAGE OF THE GIMBAL USED

The gimbal is comprised of three individual servos, each of which rotate along a coordinate axis. Each servo has can rotate within a 180° range, allowing the mounted camera to match the user's head rotation within the front half of a sphere. The brown, red and orange wires connected to each servo connect to the ground, +5 VDC power supply and PWM pulse respectively. These wires are connected to a simple circuit that provides power to each of the servos, while also providing PWM control from the Beaglebone Black.

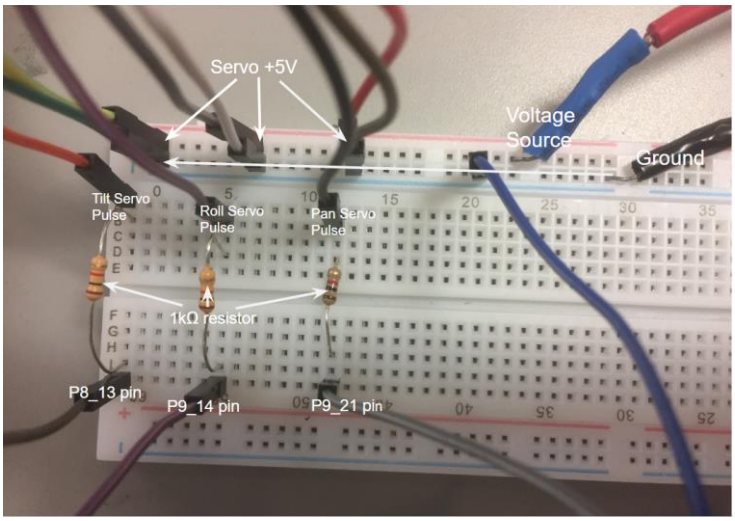


FIGURE X: GIMBAL CIRCUIT WITH DESCRIPTIONS ON EACH CONNECTION

The camera will be mounted on top of the arm mounting bracket such that the placement between the camera and the arm is the same proportion as the location of the user's arm to their eyes. The Beaglebone Black will be mounted on the back of the supporting frame.

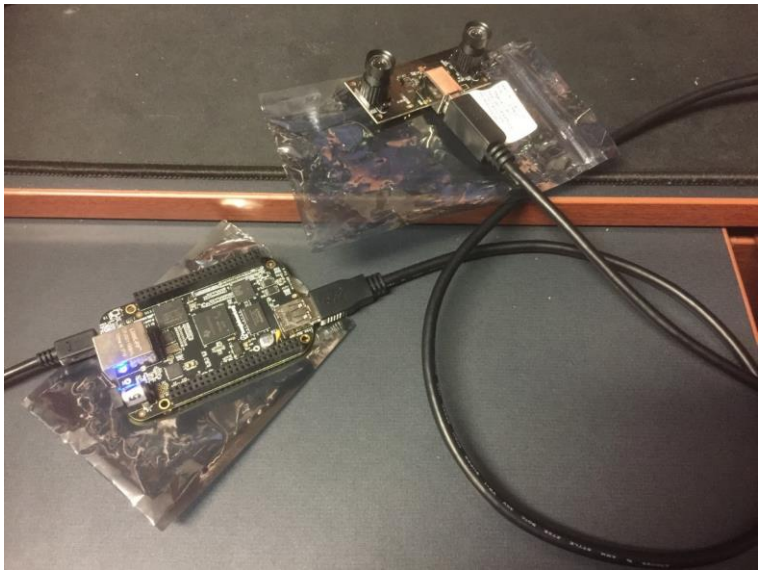


FIGURE X: STEREOSCOPIC CAMERA CONNECTED TO BEAGLEBONE BLACK

The stereoscopic camera natively outputs a 2490x960 resolution video at 30 fps. When the Beaglebone Black receives the uncompressed video stream from the camera, it compresses the image into the motion JPEG (MJPEG) file format. It then transfers the image to the computer using a Cat 6 ethernet cable. Due to the data transfer limits of the usb 2.0 port on the Beaglebone Black, the image is reduced in resolution during compression to a 1280x640 video at 30 fps. To compress and encode the video stream, a special data compression algorithm unique to ARM processors called NEON had to be used. NEON improves the data encoding and compression rates, improving the resolution at 30 fps from a 640x320 video to the current 1280x640 video stream.

FIGURE X: Without NEON: 640x240 resolution, 310x120 per eye at 30fps

FIGURE X: With NEON: 1240x480 resolution, 640x240 per eye at 30fps

3.3 Subsystem Validation

To validate these subsystems, a simple piece of test code should show that control can be established over each of the servos on the gimbal and that video data can be sent over a wired connection to the computer.

The validation requirement for the camera subsystem was to stream video from the camera to the computer through the Beaglebone Black given certain parameters to meet. Due to the camera providing

the only visual information to the user while wearing the HMD, the camera should send the video at as high of a framerate as possible to headset. The resolution determined was to stream a 1280x640 video to the computer over a wired connection.

The validation for the gimbal was to use a piece of test code to show controlled movement along each servo axis. In addition to the test code, a simple circuit needed to be constructed that provided power to each of the servos and connected the Beaglebone Black's PWM pins to the correct wires. This circuit values were simulated and measured, below is the comparison between the two.

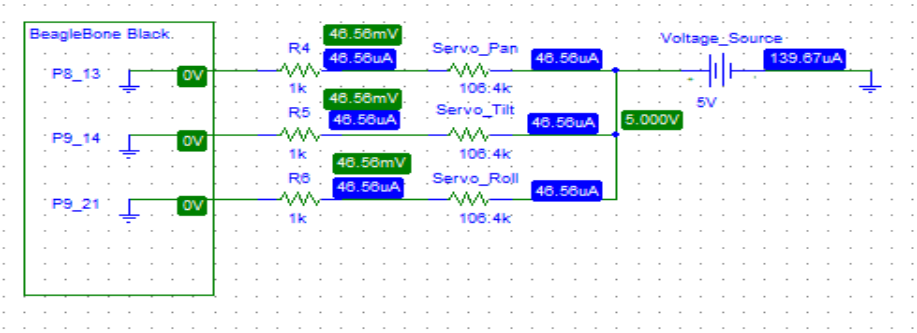


FIGURE X: GIMBAL CIRCUIT SIMULATED USING PSPICE

Specification	Simulated	Measured
Servo Input Voltage (V)	1.67	1.75
Servo Output Voltage (mV)	46.56	60
Voltage Source Current (uA)	46.56	0.012
Servo Current (uA)	139.7	0.012

As an additional step, the PWM waveforms emitted from the Beaglebone Black's pins were measured using an oscilloscope. The values measured are listed below.

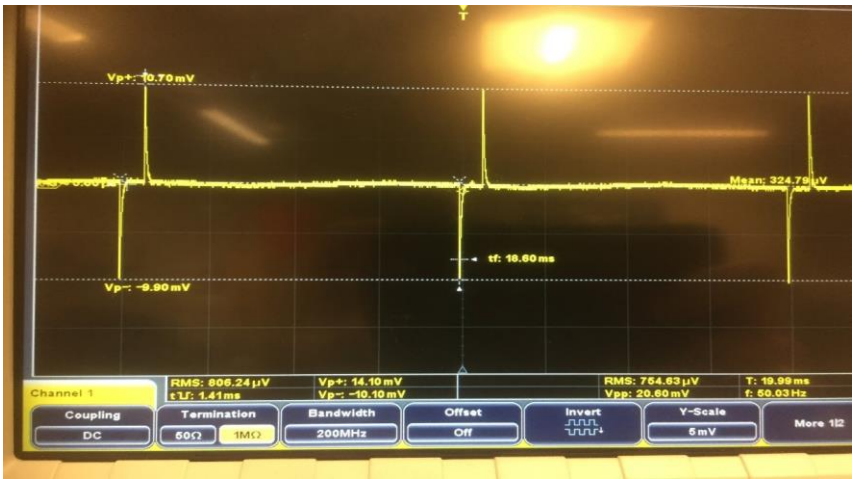


FIGURE X: PWM WAVEFORMS MEASURED DURING SERVO MOVEMENT

Specification	Measurement
Servo Pulse Duration (positive) (ms)	1.41
Servo Pulse Duration (negative) (ms)	1.62
Servo Pulse Magnitude (mV)	10.70
Period (ms)	18.60
Servo Pulse Frequency (Hz)	50.03

The gimbal's test code caused it to move in short steps, akin to a stepper motor at certain angles. This movement was based off of the pulse frequency provided from the Beaglebone Black. While this code was able to control the movement of each individual servo, additional code will have to be written that can translate the rotational values provided by the gyroscope in the headset to spherical coordinates. This will allow for more precise head movement and allow the gimbal to 'reset' to a default head position should the head tracking lose 1 to 1 movement with the servos.

3.4 Subsystem Changes - ECEN 404

The changes made in the second semester were substantial to the development of the project. One of the first major changes was the decision to use a pre-existing design in the open-source BCN3D Moveo.

FIGURE X: OPEN-SOURCE ROBOT ARM BCN3D MOVEO

Using a pre-existing design saved considerable time that would've been spent in designing, modelling and testing the arm for the project. Without this change, it is unlikely that this project would've been completed. Switching to this design both increased the scope of movement allowed by the arm without compromising the strict limit on time. The updated arm design has seven axes of movement although only three were used in the final design.

FIGURE X: ARM PARTIALLY CONSTRUCTED

Despite the time saved from needing to design the arm, it required substantial time in constructing it. All structural components for the arm were 3D printed, needing only specific stepper motors, servos, screws, and timing belts in order to construct.

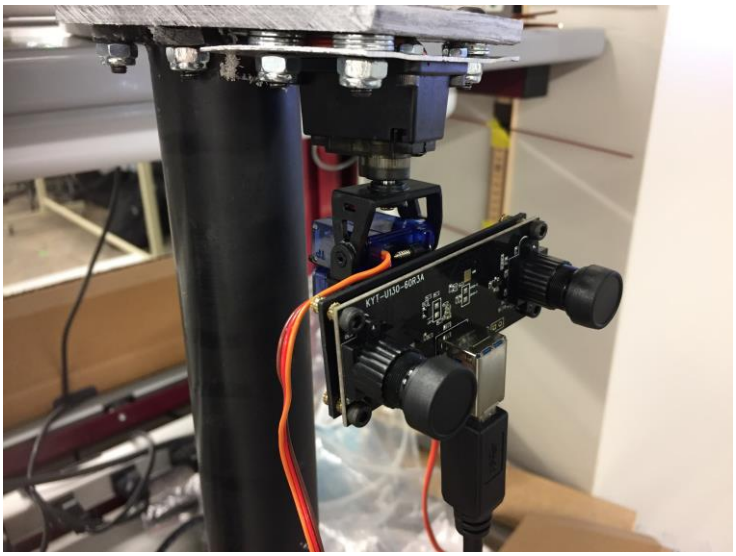


FIGURE X: FINAL CAMERA + GIMBAL DESIGN

The camera and gimbal that were used in the final design were successfully able to capture the surrounding environment and translate the head-movement into camera movement with the use of the gimbal. The camera that was used here is a stereoscopic 3D camera that technically outputted 1080p video, but because of compression restraints, had to be downscaled to 320p. Despite this limitation, the camera fulfilled its purpose, and with improved data transmission technologies, would be able to transmit a clearer image. As a side note, the gimbal had to be inverted in the final design because the cable connecting to the camera was too bulky. Inverting the camera would have added more latency in flipping the image itself in software, so a simple inversion of the values being sent to the gimbal was implemented instead, fixing the issue.

3.4 Subsystem Conclusion

The decision to switch to the open-source design was a boon for the project. Having a complete idea about what the capabilities of the arm would be allowed everyone in the group to use their time more effectively, even while the arm was still being constructed. The build quality of the arm was quite good, but late into the project, issues with certain rotation axes hampered the final arm demonstration. Despite this setback, the arm went above and beyond the original project parameters. Its potential capabilities make it a great candidate for similar projects. The final camera and gimbal implementation

were similarly ambitious, offering additional capabilities if certain problems are fixed. As it stands, the design that was chosen fulfilled its purpose and provided remote vision to the user.

POWER SUBSYSTEM

4.1 Subsystem Introduction

The power subsystem is split between two main power designs the first consisting of an AC-DC converter designed to supply DC-regular power to all the components of the robotic arm system consisting of 5 “NEMA 17” stepper motors. The second design will consist of a DC-DC buck boost converter that will power all small electronics for both the robotic arm and virtual reality consisting of Beaglebone black, Arduino Due, and Gimbal. Each power system design was both simulated and tested to ensure its stability, capacity, and consistency. Each test helped verify that the power subsystem design would support the components of the system that will be powered by it.

4.2 Subsystem Details

A block diagram of the subsystem is shown below

FIGURE 1: Functional block diagram of the AC-DC converter design.

4.3 Subsystem Validation

The AC-DC converter was first simulated using pspice to ensure that the AC signal was first turned into a usable DC signal. Once this was verified the circuit was constructed to maximize its efficiency being able to produce a constant DC signal with as little noise as possible to ensure no harm to the components being powered.

FIGURE 2: STIMULATED ARM OUTPUT

Once we were able to reach a consistent 12V output I was able to redo the simulations to improve design under extreme conditions while the converter will receive 120VAC most of the time in order to ensure the converter will in other conditions its was tested from 110-130VAC. The outputs where then plotted to see efficiency of the system as shown in the figure below.

FIGURE 3: POWER EFFICIENCY

Once efficiency of design was verified, next step was the production phase. The printed circuit board was designed to reduce noise as much as possible, for this reason dedicated planes were established for input, output, and ground. Final pcb design is shown below

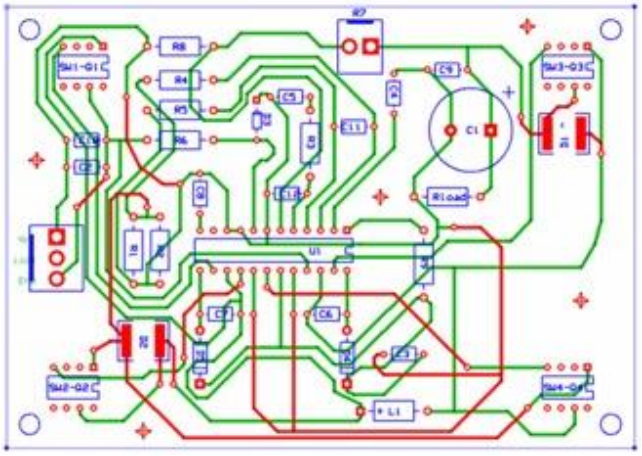


FIGURE 4: PCB LAYOUT

Figure 5 shows printed pcb courtesy of Sanmima Corporation and having all components soldered into place.

FIGURE 5: FINAL AC-DC CONVERTER

Once everything was soldered, we ensured the converter's output by connecting it to the oscilloscope and ensuring that we received not only a 12V output but a fully rectified DC signal. As displayed in Figure 6, we end up with a 12 V output and DC signal with very minimal noise in the circuit.

Figure 6: Oscilloscope Results

FIGURE 6: OSCILLOSCOPE RESULTS

	Input (Wall Outlet)	Oscilloscope (output)	Multimeter
AC-to-DC	**120.23VAC**	**12.1VDC**	**12.043VDC**

TABLE 1: VERIFIED CONVERTER OUTPUT

The table below shows both the current and voltage draw of each of the different components being powered by both power supplies. The first design AC-to-DC converter will be operating at 12V meeting the specifications set on the NEMA 17 datasheet. While the rest of the components will be powered through the DC-DC converter operating at 5V.

Component	Rated Current	Rated Voltage
Arduino Due	800 mA	7V
Beagle bone Black	1200 mA	5V
Gimbal-Camera	900 mA	5V
Nema 17 Stepper Motor	350 mA	12V

TABLE 2: RATED CURRENT AND VOLTAGE FOR ELECTRONIC COMPONENTS

4.4 ECEN 404 Changes

One of the most significant changes that occurred during ECEN 404, was we were able to begin construction on a 3D printed open source robotic arm. Changing of the robotic arm design meant that power calculations would need to be redone to ensure each stepper motor incorporated in the overall design received enough power to lift its load in this case each stepper was a assigned to a different joint of a human arm. With this new design certain motors that were previously thought to be incorporated in the final design ended up being replaced with new steppers. In our case we had replaced some of our NEMA 17's for NEMA 20's still using same micro step drivers, giving us the availability to push anywhere from 0-42 VDC to the driver itself. From there we could configure the drivers to output current to match our power calculations. Due to the new found NEMA motors and the overall design of the robotic arm a rated power supply was needed to drive the steppers. After seeing which steppers were not receiving adequate power to lift their portion of the arm we need to step up the voltage using the potentiometer incorporated in the power supply. Power supply that ended being chosen for this specific arm was the LM YN DC 0-60V 8A Adjustable Switching Power Supply. Able to output 0-36VDC, 13 A. Falling well below our margins for our micro step drivers while still allowing for enough current to flow through steppers to lift weight.

4.5 Subsystem Conclusion

Each part of the subsystem was shown to work correctly. The power subsystem effectively generates and stores the needed power for each stepper motor included in the robotic arm. When interfaced with the other subsystems, it will enable the robotic arm system to continuously and autonomously consume required power based on the data transmitted through the Virtual Reality subsystem.

VR Telepresence Robotic Arm

Aaron Ingram

Salvador Cuevas

Eduardo Calderon

Klin Rothenberger

SYSTEM DESCRIPTION AND DEVELOPMENT

REVISION – 1

Dec 2018

Table of Contents

OBJ

1 Overview

The Virtual Telepresence Robotic Arm is a remote-controlled robot arm designed to be operated with the movement of your own arm. In addition to this intuitive movement, camera feedback is sent to the operator's headset to allow remote vision of the arm. The system consists of several different subsystems: Robot Arm and Camera, Motor Control, Motion Control, and Power. Integration of the four subsystems was crucial to allow for the intended design of the final product. Once completed, any user would be able to control the robot arm to do various tasks while from afar.

2 Development plan and execution

The Virtual Telepresence Robotic Arm project has been under development starting from last semester and finishing up at the end of this current semester. Last semester was focused on individual subsystems and validation for those individuals. This current semester has been focused on integrating the 4 subsystems and testing for a full system validation. This report will be focused on the current semester and validations for the full system

2.1 Design Plan

To begin the second semester, we reevaluated the current physical robotic arm that we were planning to make. One of the main design issues from last semester is that we had no guarantee that the robot arm designs our group mocked up would work for the loads we were using for testing. Thus, we decided to go with an open source tried and tested robotic arm that could be 3D printed. Once we made the decision with this, we then found extra considerations we would need to take for our design.

With a new robot arm, our power systems would need to be reevaluated to allow for the motors to be driven optimally. The previous semester, our power system was a fully designed power supply which was not as optimal and reliable as many affordable consumers power supply. We decided it was best to use the power supply we bought to allow for safe and constant power to drive the arm.

Next, we looked to our designs for camera pass through from the robotic arm to the user's computer. We deemed that the Beagle Bone Black did not have a fast-enough data transmission, so we switched to using the Odroid Xu4 which sported USB 3.0 for our camera passthrough. Thus, these improvements towards the design allows us to more easily complete the full system integration for the project.

2.2 Execution

After making the design improvements, our next step was to go through each subsystem and verify that each subsystem is still meeting their goals. Since we replaced the Beagle Bone Black with the Odroid XU4, we had to ensure that serial communication could be re-established so that the Arduino Mega could still receive commands to move the motors.

From here, the communications side had to become fully integrated to allow for motor movement to occur when hand input is given. The Computer and Odroid XU4 would need to be connected over ethernet to allow for a TCP server to be created which would allow for the Odroid to receive values from the Leap

motion. Then, we can utilize serial communication to allow the Arduino Mega to receive the same information from the Odroid, and then the Mega can write to each motor accordingly how much to move.

Execution Plan:

Task	Deadline	Member
Final design for Robot Arm	September 19	Klin
Update Power for Arm	Septemver 19	Salvador
Validate motor subsystem	Septemver 19	Eduardo
Validate motion subsystem	September 19	Aaron
September 19 Checkpoint		All
Submit Robot Arm for Printing	October 3	Klin
Evaluate Camera Gimbal Power	October 3	Salvador
Re-establish Serial Communication	October 3	Eduardo
Establish TCP Server	October 3	Aaron
October 3 Checkpoint		All
Final design for Robot Arm	October 17	Klin
Update Power for Arm	October 17	Salvador
Camera UDP Stream on Odroid	October 17	Eduardo
Refine headset image processing	October 17	Aaron
October 17 Checkpoint		All
Start Arm Construction and mount Camera	October 31	Klin
Wire up motors and power	October 31	Salvador
Finish full communication pathway	October 31	Eduardo
Refine values for motion control	October 31	Aaron
October 31 Checkpoint		All
Validate and test range of arm movement	November 14	Klin
Validate and test power supply to arm	November 14	Salvador
Validate and test motor precision	November 14	Eduardo
Validate and test communication delay	November 14	Aaron
November 14 Checkpoint		All
Validate and test control of robot arm for demo	December 5	All
Finish VPS video submission	December 5	All
December 5 Demo		All
Finish final report	December 6	All

FIGURE X: EXECUTION SCHEDULE

2.3 Validation Plan

The validation plan for the full robotic arm system can be found below. The Virtual Telepresence Robotic Arm is intended to be operated while the user is indoors. The functionalities validated include both TCP

and UDP stream response time, weight of object lifted, video feed frame rate, and the consistency of motor accuracy. These values were validated first as individual components where applicable and then again as part of the complete system. To validate the data stream, we measured how long it took for a value to make it from point a to point b. for UDP it was a camera image, for TCP it was in int representing motor controls. To validate the lift strength, we just got an object we knew the weight of and lifted it. For the Video feed framerate, we checked the refresh rate of the animation loop. The following table shows the results of these systems after they have been integrated into the whole design.

TABLE X: VALIDATION RESULTS

	Specification	Result
TCP Stream Response	initially <200ms, revised <500ms	200ms-400ms
UDP Stream Response	<200ms	100ms – 200ms
Lift Object	Minimum 50g	100g
Video Feed Frame Rate	>= 20 FPS	Stable 30 FPS
Motor Consistency	<5 Degrees given 200 commands	<1 degree

3 Data

3.1 Headset Data

The Oculus headset's primary data came from analyzing the pitch roll and yaw of the headset. The position was taken each time a new frame was rendered to the device. This led to roughly 50 samples/second. (for the full readout graphs see the Headset subsystem report). Based off these samples we were able to generate the following table:

TABLE X: HEADSET RANGE VALUES

	Angle Range (0 being staring straight ahead)	Raw Values
Pitch	-50° to 80°	-.4 to .74
Roll	-30° to 30°	-.27 to .34
Yaw	-90° to 90°	-.55 to .6

Because we map these values to servo motors that have a very specific range as well this table allowed us to generate our equations to match angles between the servos and the headset.

3.2 LeapMotion Data

The LeapMotion sensor has many different values that one can get, but for this project we only used a few. We focused on the pitch and rotation of the palm, the pinch distance, and the height of the wrist. Like the headset, we generated a table based off of min and max values of physical position and the raw data.

TABLE X: LEAPMOTION RANGE VALUES

	Physical min/max	Raw Sensor Values
Palm Pitch	-80° to 80°	-1.45 to 1.36
Palm Roll	-90° to 90°	-1.67 to 1.72
Pinch Distance	2 inches (open) to 0 inches	0 to 1
Wrist height	0 inches to 18 inches	0 to 463

These values weren't mapped directly, but rather had a scaling factor applied to them so that it would correspond the correct number of steps taken by the corresponding stepper motor.

3.3 Serial Data

The serial communication between the Arduino Mega and Odroid XU4 perform at a baud rate of 19200. The information being sent consists of a string of data, a character at the beginning followed by 7 integer values. We parse the data on the Arduino to write or step to the necessary value at the position.

Position	Example	Function
1	'L'	Validation Check
2	90	Write to Pinch Servo
3	2000	Step Wrist Pitch Motor
4	500	Step Wrist Roll Motor
5	0	Step Elbow Motor
6	130	Write to Gimbal Lateral
7	40	Write to Gimbal Vertical
8	50	Write to Gimbal Roll

3.4 Stepper Motor Data

The three Stepper motors we utilized each step at a different rate for a full rotation. This is due to the step sizes we set for the micro step drivers attached to each motor as well as the physical implementation of the motor. Two of the three stepper motors use elastic belts and gears to rotate the arm components, which means that the full rotation is also based off of a gear multiplier. For simplicity, we have determined the stepper values needed for a full rotation. Granted, the elbow and wrist pitch cannot perform a full 360 degrees rotation due to only allowing 180 degrees movement on the arm.

TABLE X: STEPPER MOTOR RANGE

Movement Type	Steps for Full Rotation	Range of Physical Rotation
Wrist Pitch	4000	Roughly 4 Rotations
Wrist Roll	32000	½ Rotation
Elbow Rotate	128000	½ Rotation

3.5 Servo Motor Data

We used four servo motors for the VR Telepresence Robotic Arm. One servo is utilized for the pinch motion on the arm. The other three are servo motors dedicate to controlling the camera's lateral, vertical, and roll movement. In theory, all four of the servo motors can move from a position of zero degrees to a position of 180 degrees. In actual implementation, the servo motors cannot utilize the full range of motion due to physical limitations of the arm. The pinch servo is limited by how far the robot claw can open and close. The camera gimbal servo motors are limited because they increase the risk of jamming the motors when they reach the critical end points of their rotation. Due to this issue, we chose to limit the range of rotation within acceptable bounds where we see a decrease of jamming.

TABLE X: SERVO MOTOR RANGE

Movement Type	Range of Motion (Degrees)	Actual Range (Degrees)
Pinch	0 - 180	0 - 90
Camera Lateral	0 - 180	30 - 150
Camera Vertical	0 - 180	30 - 150
Camera Roll	0 - 180	30 - 150

4 Conclusion

At the beginning of ECEN 403, we set out to develop an intuitive way to control a robotic arm whilst having telepresence to allow a user to perform remote control more efficiently. Now that we are at the end of ECEN 404, believe the VR Telepresence Robotic Arm has proven to be successful as a novel way to approach intuitive remote control over a robotic arm.

4.1 Key Decisions

One of the first major decisions of this semester was to go with an open-source arm design as opposed to an original design. The design chosen was more ambitious and complicated than one that could be designed specifically for the project parameters. Despite the additional complexity, the choice saved an enormous amount of time that would be spent in designing and testing the arm itself. By using a proven design, it was known not only that the arm would work, but how it would work. This allowed team-members to develop subsystems with a more complete understanding about what the final capabilities of the arm would be.

We decided on a few constraints for the VR Telepresence Robotic Arm. One such constraint is the amount of motors we chose to utilize in the arm design. For instance, the robot arm has 7 motor joints that can influence movement. However, since our design involves using a human arm to intuitively control the robot arm, we chose to not implement some of the motor joints. In the end, we chose to use the 4 joints that consist of pinching, wrist vertical movement, wrist rotation, and elbow rotation. We determined that using the last 3 motor joints would not contribute to allowing an intuitive method to control the robotic arm. The decisions to limit the amount of motors utilized was also influenced by the power being driven to the robot arm. We concluded that the extra power needed to drive the bottom

motors would be far greater than the benefits of being able to move those motors, thus we excluded them.

Another key decision we made was to limit the camera feed to only a single camera rather than the original two camera feeds. There were a multitude of reasons to make this decision, but the biggest reason was that true stereoscopic vision wasn't possible with the version of headset we were using. We could display both camera feeds, but it wouldn't feel immersive. Instead it would feel like you were pressed up against a TV or else both feeds would be visible to both eyes. Though even if we had been able to use both feeds it's still possible that we wouldn't have gone that route. We had to compress our video feed using the JPEG format. Using two camera feeds doubled the size of the file to compress and ended up with two very pixelated feeds. This could have been less desirable than a clear mono-scopic window into the world.

4.2 Learnings

Throughout the course of this project, all team-members developed valuable skills in bringing a project from initial conception to validation and conclusion. Initially, many decisions had to be made without the knowledge or experience to know whether they would be feasible within the given time-frame. The lesson learned from this period was not necessarily being a better predictor, but understanding that given the ambition of the project, strict self-imposed limitations on the scope had to be made. This discipline was difficult considering how easy it seemed to add new features, but it ultimately proved to be the deciding factor in the success of the project.

As the project progressed, team-members learned valuable skills in TCP/IP and UDP communication, microcontroller programming, OpenGL Rendering, power system management, and troubleshooting every step of the way. Ultimately, the project progressed more organically as team-members learned what skills were most useful and sought to balance the workload for everyone involved. Communication was key throughout this process as team-members relied heavily on the combination of their subsystems.

4.3 Suggested Improvements

A suggested improvement for this project is to use a different VR headset. The oculus rift is good for game production as well as using existing products as a consumer. However, as an augmented reality platform it has many short comings. The headset I would recommend in an ideal scenario is the HTV Vive. It allows the developer more freedom and control over the hardware than the Oculus Rift does.

Commented [RR3]: boy whatchu saying bout my headset I swear on me mum Ill smack ya right in the gobber if ya keep rippin on me pride n joy, ya don't think tat valve doesn't have their system locked down ya crazy

Another area of improvement is the sensor coverage. The LeapMotion sensor is meant to detect the hand. It does that incredibly well. But our arm has joins for the elbow and could be expanded to incorporate the shoulder as well. If we wanted to do that then we would be best off to get an additional sensor that is designed specifically to target those joints.

The Arduino code has a check system to see if the string being sent from the LeapMotion starts in the correct position. For instance, the string always starts with the character 'L', and so if the Arduino reads the serial in the middle of the string, it won't process the string unless it reads it from the start. This method still allows for invalid strings consisting of all zeroes to pass through, even if the LeapMotion

was sending non-zero values. Thus, a suggested improvement to this issue is to use a checksum to check for validation. If we do this, then when the checksum does not match the sum of the movement data, we can simply discard the string and check for another incoming string.