Capstone, Fall 2015

Intuitive Telepresence for the Disabled

Prof. Kimani

Andrew Pereira, Dominic J. Catalano, Nasir al Shubbar, Russell Walker, Sean Goodwin

Contents

Abstract

Our Capstone project takes the concept of Telepresence and applies it to assisted living applications for handicapped individuals. Telepresence is the feeling of being elsewhere as emulated with virtual reality technology. For this project, we plan on designing a remote-controlled robot that is equipped with a robotic manipulator arm, a lift system, and a camera gimbal system to get a better view of the environment. The camera system will stream video live to an Oculus Rift VR headset which, when worn, will create the feeling of telepresence as if the user was travelling along with the robot. The head tracking capability provided by the Rift will correspond to the camera gimbal's movement, allowing the user to freely look around the surroundings of the robot to provide an unparalleled field of view. Finally, by using the Nintendo Wii Remote to provide users with an intuitive motion control device to manipulate the robotic arm, the final product will be very simple to use for accessibility purposes. Our hope is that the final product will be modular and extensible, allowing our concept to be used for a variety of other applications as well.

Introduction

In the past few years, advancements in virtual reality technology have opened the door to a new and unexplored field of science and engineering. They have enabled the engineering community to redefine how people interact with their environment—giving more control and access than ever before. Virtual reality allows people to experience a world outside of the current reality, and now is the time to enable interactions with this augmented world view.

Currently there exists remote operated vehicles (ROVs) that can offer a view into a different part of the world and interact with it. However these devices come with some issues. A technical report from Defense R&D Canada lists common issues as "having to remotely operate the vehicle", "limited spatial awareness because of impoverished visual cues", and the camera's limited field of view[1]. Commercial aquatic ROVs suffer from this shortcoming while also being expensive. With virtual reality, issues pertaining to the visual difficulties can be resolved. Through the use of a commonly known control device that is intuitive to use, difficulties in viewing the ROV's surroundings can be mitigated.

In past decades, the capabilities of robots have increased dramatically—giving a newfound level of control and precision to the operator. However as these capabilities grow, so do complications around the control. Advanced precision devices, like bomb disposal robots, offer the user very delicate and exact control, but are limited by the requirement of training and the complexity of those controls. Devices such as Microsoft's Kinect, Leap Motion's controller, along with Nintendo's Wii remote, offer possible alternatives to a panel of knobs and buttons. Thus, complicated devices can be simplified and made available for large numbers of end users without requiring special training.

The combination of virtual reality and intuitive motion control offers a wide array of possibilities in the fields of scientific exploration, handling situations too dangerous for direct human contact, and even assisted living. By enabling handicapped individuals to experience and interact with an environment that they are otherwise unable to reach through a simple and intuitive interface, the quality of life can be significantly increased.

Problem Formulation

Over the course of our project's development, we faced numerous problems and came up with solutions to match them in order to have the best design possible that meets all of our requirements.

Brainstorming

This was the most challenging stage of development wherein we came up with a number of problems that needed to be solved and the solutions for each one. At the very beginning, we knew we wanted to work with VR technology and incorporate it somehow into our project. In this way, our brainstorming stage was somewhat unorthodox as we had solutions (the Oculus Rift) before we had the problem that they solved. Therefore, we started looking into different fields to think of problems that this technology could solve. By looking at other projects involved with the Oculus VR headset, we decided that we could use the concept of Telepresence and a robot as well. From there, we researched more into what possible needs telepresence with a VR headset and a remote controlled robot could solve. After discussing with our advisor and with some student groups on campus, we came to the conclusion that handicap assistance might possibly be the best use for this technology.

Once we had decided upon this, we came up with a concrete list of features that our project must meet. Our Product must have:

1. Cameras that stream to an Oculus Rift
2. Telepresence via a mobile platform
3. A camera gimbal system to match the Oculus Rift head tracking
4. A robotic arm to manipulate the environment with motion control

Each of these four parts required more details considering the cost and the benefits of using it in our project. Based on this, we elaborated on the above by breaking them down into tasks as follows:

1. Cameras that stream to an Oculus Rift
    a. Wireless connection from camera to Rift
        i. Bluetooth
        ii. Wi-Fi
2. Telepresence via a mobile platform
    a. Build robot from scratch
    b. Buy robot kit and modify
3. A camera gimbal system to match the Oculus Rift head tracking
    a. Oculus head tracking
    b. Gimbal design
        i. 3D Printed
        ii. Hand-crafted
4. A robotic arm to manipulate the environment with motion control
    a. Motion control
        i. Microsoft Kinect
        ii. Leap Motion
        iii. Wii Remote
    b. Robotic arm
        i. Borrow from ECE lab
        ii. Buy off-the-shelf arm
    c. Lift system

Design

Physical Components

Gimbal

After a careful evaluation of our initial gimbal design shown in Figures 2 and 3, we decided to increase the stability of the components. This was accomplished by adding a second stabilizing bar on the base link to have the secondary linker stabilized by more than the servo that powered it. We also decided to encapsulate the two servos in the secondary linker to provide a more rigid structure. Because of the precision needed in these parts, we had the parts 3D printed using the 3D printing lab in the library.

Figure 1.1 - Gimbal Substructure

The eyepiece was constructed out of balsa wood as a light, low-cost material. We decided this piece should minimize cost in the event multiple iterations of this piece was necessary. Adding the eyeplate for the cameras, and the bottom servo, we obtained our final gimbal.

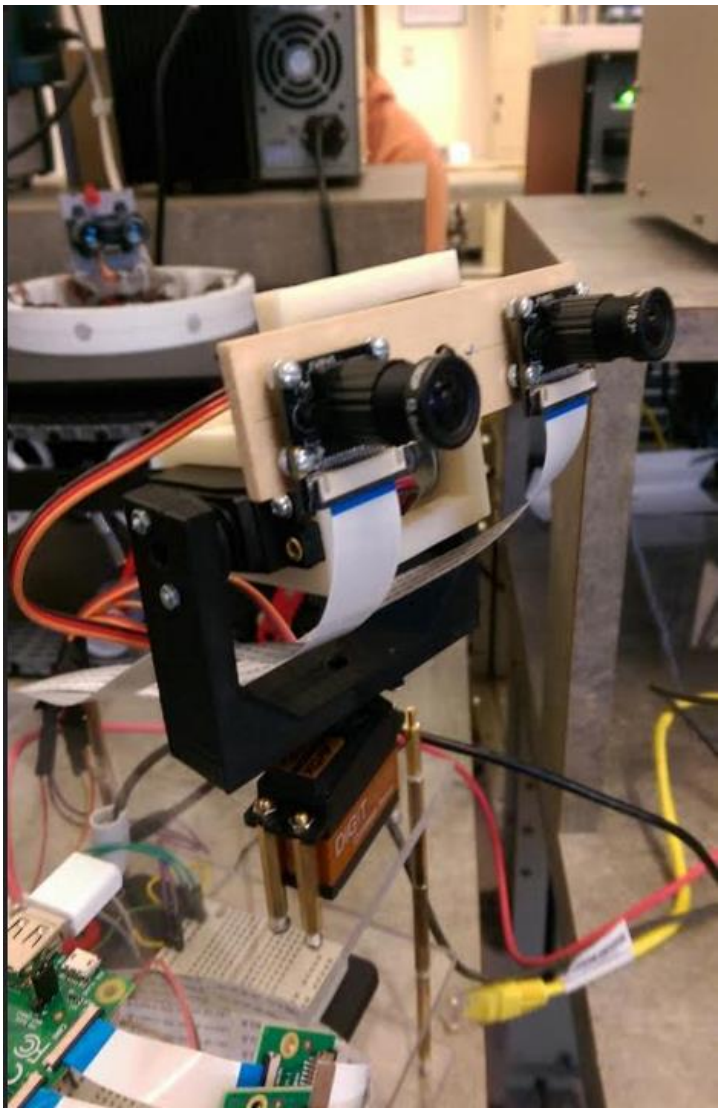


Figure 1.2 - Final Gimbal Structure.

Mobile Platform

The mobile platform was comprised of three components. First, the robot designed to give the robot mobility in an environment, the arm provided by the ECE lab, and the platform one which the other components would be mounted.

We opted to purchase a kit for the robot which was assembled easily, and no modifications were necessary to the arm. From there, we used acrylic to create mounting space for our components. A base plate in acrylic allowed us to mount the robotic arm and its attached plate to the built kit base. We then cut the acrylic with the aid of the mechanical engineering lab to create an adjustable, multiplatform hardware stack. This stack allowed plenty of room to mount components in a place where we did not have to be concerned with the arm hitting it.

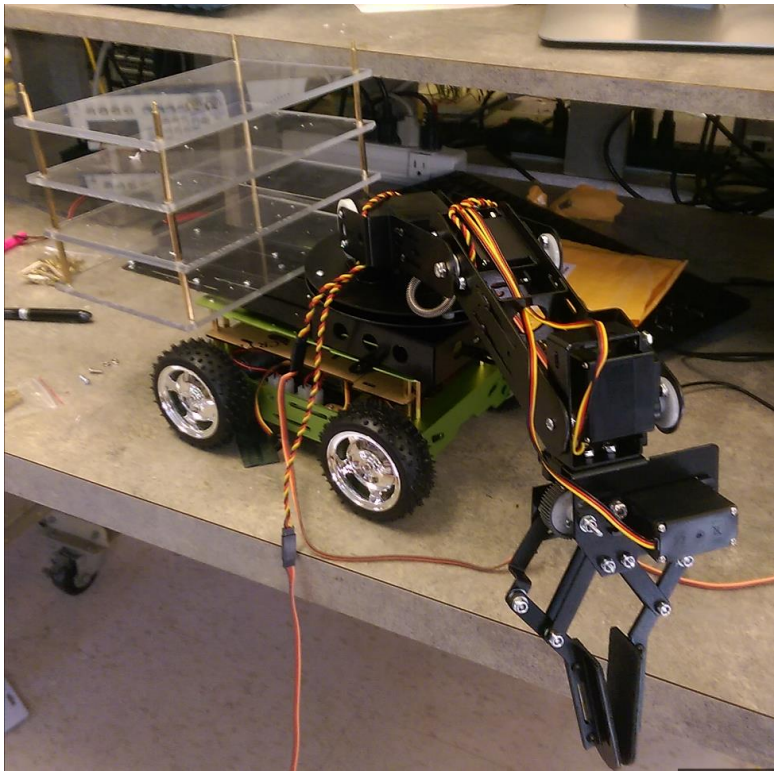


Figure 2.1 - Robot with base, arm, and hardware stack

Mounting the components onto the hardware stack was a fairly painless and straightforward process. We had also labelled all of the PWM cables to their associated servos to make debugging easier.

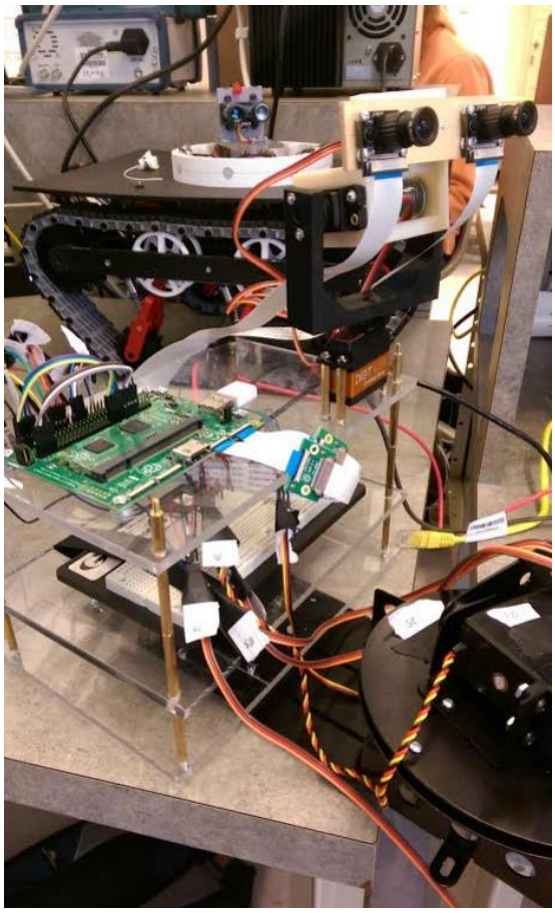


Figure 2.2 - Hardware stack with labelled components.

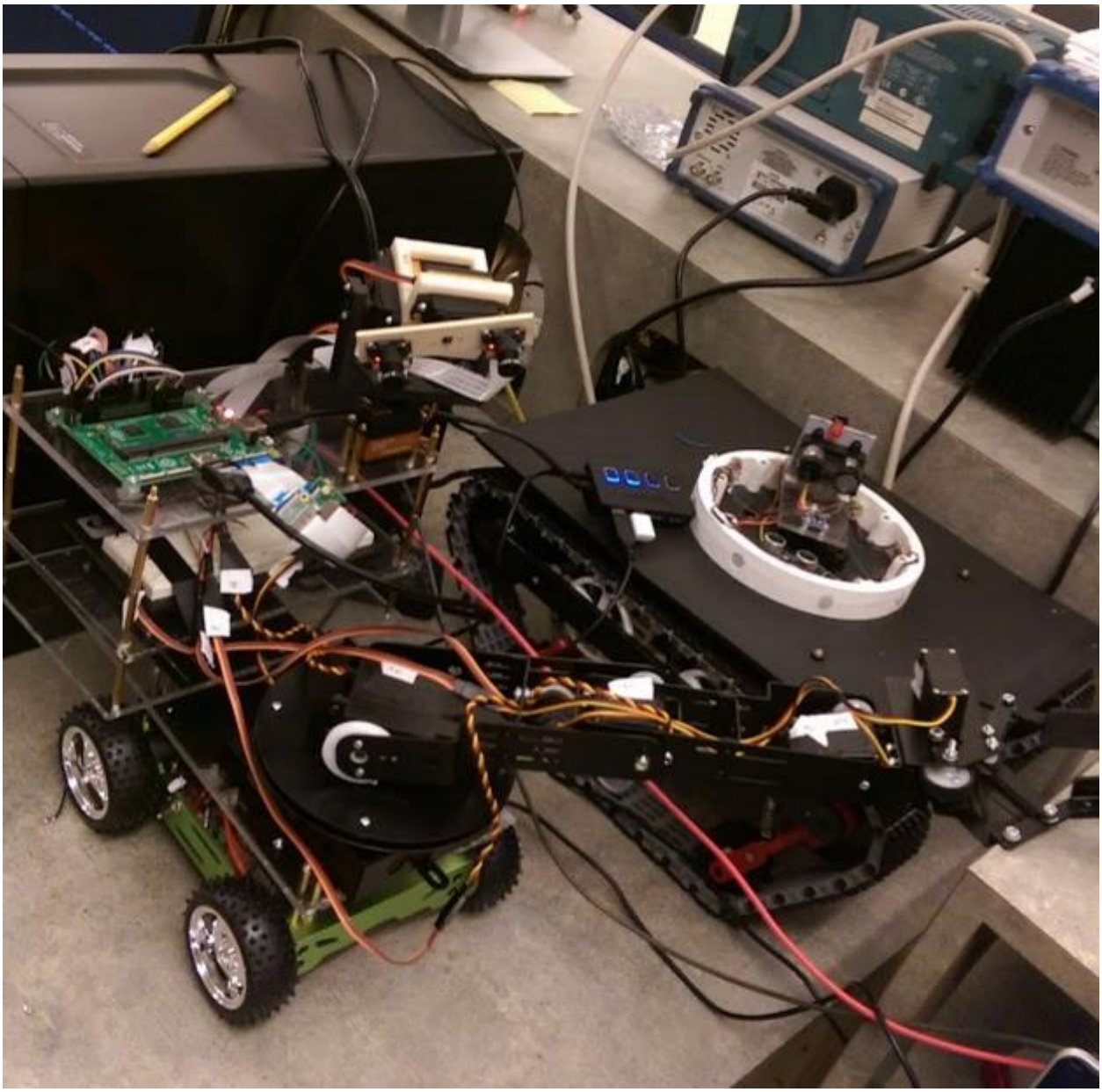


Figure 2.3- Final Assembled Robot

Software

Rest Server

In order to achieve communication between the various systems created as part of this project, we decided to design and implement a REST server. The server is written in Python and utilizes the Flask library to implement the networking backend. Various endpoints are provided to the modules to allow them to communicate with one another. We choose to design our communication model in a task-centric manner. Each task consists of a "TaskName" which details the nature of the task, as well as a "Values" field, which contains the associated task data. There are three URL endpoints that can be used to achieve this communicate are: "create_task/", "get_task/<TASK_NAME>", and "mark_task_as_completed/<TASK_ID>". Together these three enable simple, asynchronous communication between all of the discrete devices.

Each input device (Wiimote, Oculus Rift) has one or more "tasks" that they are in control of. Each task consists of a "TaskName" which details the nature of the task and is static across all similar tasks, as well as a "Values" field, which contains the associated task data. When a user action is detected (such as an arm swing or head motion), the script in charge of the particular device reads the raw data from the devices' sensors. This data is then compared with the previously collected state data; if there is not enough difference between the recorded values (usually requires 1-3% or so difference), the data is discarded and the script resumes waiting/polling. If the data does differ enough, then the previous state data is replaced with the newly read values, and a new task is created. This task is then stored on the REST server until it is retrieved and marked as completed by the corresponding consumer device. These consumer devices poll the REST server for new tasks, attempt to apply the task to the related hardware (e.g. moving servos), and mark the task as completed on the REST server upon success. However one downside to this approach is that in the event that there are network delays or problems with the consumers receiving the tasks, there may be a backlog of tasks to run.

After implementing and beginning the integration testing phase of the project, we found that the delay between the base delay between a task being created and competed was quite reasonable and sufficient for the project. However, we spent much time working on finding appropriate polling times to prevent overrunning either the RPi or the REST server. Given additional time, we would likely transition to a WebSockets based approach to remove this concern. As each task is only used by two devices (producer/consumer), there is no real need for a centralized server to handle all communication. This would be a relatively minor enhancement, but if support for additional devices is added, it will become a greater concern.

Streaming

We chose the Raspberry Pi Compute Module primarily because it had two camera ports, which would allow us to easily get camera streams from two sources at once. We quickly decided against using USB webcams, due to the poor performance and low quality of the video compared to using the Raspberry Pi's Camera Modules. Our first challenge was to get the video from both cameras at the same time and combine them into one video stream. Luckily the capability to do this was recently added to Raspbian, the Linux distribution that the Pi runs. Our next challenge was to get the video streamed efficiently over a network. Sending the raw video data via netcat worked, but resulted in a delay of several seconds, which was unacceptable for our purposes. We wound up using gstreamer, which encoded and compressed the video enough to give us an acceptable amount of delay.

After applying the distortion effect to the video via the Virtual Desktop program, the video gives the desired 3D effect, and the frame-rate and latency are acceptable. If we were able to use a separate board for the servo control, that would give us the processing power necessary to make further improvements to the video quality and field-of-view.

Oculus

The Oculus Rift is a proprietary piece of hardware with its own Software Development Kit (SDK) to allow a user to create software based around it. It provided us with a multitude of capability, though we only needed a select few methods from it. In our case, our interaction with the Rift was twofold: displaying the camera stream *to* the Rift and retrieving the head-tracking data *from* the Rift. For the first problem, we used a number of third-party programs, and for the latter, we worked directly with the Oculus SDK.

Due to the nature of current VR technology, it was necessary to apply a distortion effect to any image that should be viewed through the Oculus Head-Mounted Display (HMD). We spent some time debating on how best to display images to the Rift which included programs like VLC, MPlayer, VRPlayer, and several others. Some of them had features where we could program our own distortion filter to achieve the effect we needed. However, the solution became much simpler in our final prototype, where we found a program titled Virtual Desktop that simply warped the entire desktop when viewed through a VR headset. Since we already had a solution for streaming in the form of gstreamer, it was simple to just maximize the window that gstreamer creates and then use Virtual Desktop to transform that image.

One of the most powerful features of the Oculus Rift is the head-tracking, which allows a developer of a VR application to move the camera according to the user's head movements. It does this by keeping track of a number of IR LEDs located on the Oculus HMD with the small Oculus webcam-like Tracker. By keeping the front of the HMD facing the front of the Tracker, the Oculus reports its position as facing forward. From that one point, the Oculus can detect the full range of motion available to a user's head, including rotation angles and distance in a 3D space. While we never used the measure of distance in our application, we deliberately designed the gimbal to support the three Euler Angles of rotation: yaw, pitch, and roll.

The Oculus SDK provides a method that returns the tracker data in the form of four angles in radians. Those four angles correspond to orientation, pitch, yaw, and roll. In this example, orientation refers to the direction the user is facing (forward/backward), rotation along the X axis (pitch), rotation along the Y axis (yaw), and rotation along the Z axis (roll). For our final value sent to the gimbal's servos, we mapped the angles to PWM signals by range - smallest angle to smallest PWM signal.

Power

Powering the system that consists of a number of servos, for the arm and the gimbal, the mobile platform and the cameras has gone through different stages. In the building stage, we considered using the constant DC power supply in the lab to power the board and the Raspberry Pi with a 6V DC. Then, in the testing stage, the team has come up with a variety of choices to power the robot such as building a portable power supply with AA batteries or using pre- existed power supplies. Connecting, in parallel, two sets of four 1.2V AA batteries in series, gave the required voltage and power to power the servos

and we succeeded. However, this was not very useful for long-time use, so we decided to try another option that can handle most of the equipment, and that would be rechargeable.

The platform has four DC motors powered by a rechargeable 2200 mA/h Lithium Battery, which also can handle more than the four motors since there is no heavy load on the platform, so we decided to use it to power the board and the servos. For powering the pi, we considered using a portable charger that provide 5V and capable of producing 3000 mA/h and a maximum of 11.1 W/h, and it has a USB output which makes it easy to connect it to the pi.

Figure 3.1: Energizer and Lithium battery

Impact of the Project

When designing this project, the primary application in mind was as a way to assist those that have limited mobility, including the elderly and disabled. Products like the one designed would allow them to have a larger degree of independence by not requiring them to have a physical caretaker present to do certain tasks, such as retrieving a television remote or flipping a light switch. The proposed control scheme is intuitive and easy-to-use, which is important for users to quickly obtain mastery of the device.

The proposed system is modular and could therefore be easily modified to suit other applications as well. Such a system would be ideal for any situation where direct human involvement is hazardous, but still requires an amount of situational awareness and intuitive control that current ROVs cannot provide. It could perform reconnaissance for firefighters or the military, assist in deep-sea or outer space scientific exploration, or investigate contaminated areas.

Budget

Line item	Price	Quanitity	Expanded Price	Receipt
Compute Module	100	1	100	compute module
Raspberry Pi 2	35	1	35	pi2
Camera	42.99	2	85.98	cameras
Servos	59.9	3	179.7	servos
Oculus Rift Dev Kit		1		On Loan
WiiMote		2		On Loan
Dimension Print	40	1	40	dimension_receipt
Wifi Adapter	9.95	1	9.95	wifi_adapter
Robot	169	1	169	Robot
Camera Adapter	24.27	1	24.27	camera_adapter
Cables	12.95	1	12.95	cables
Acrylic	21.37	1	21.37	HD
Electical Tape	1.97	1	1.97	HD
Acrylic Cutter	4.78	1	4.78	HD
Foam Tape	3.48	2	6.96	HD
Long spacers	7.77	1	7.77	spacers
Short spacers	12.87	1	12.87	spacers
Servo Replacement	15.19	1	15.19	Astor USPS
Total Shipping Costs:			46.72	
Total:			774.48	

Conclusion

Our final product met our proposed primary goal and provides a feeling of telepresence for users of the system. The Oculus Rift HMD displays the video stream captured by the Raspberry Pi cameras which will move according to the head movement detected by the HMD. As discussed, the applications for this kind of technology extend far beyond home use and we can see a product similar to our own being used in a variety of fields. Indeed, the VR technology that we only utilize for telepresence has nearly limitless potential in terms of what we can simulate. However, for our purposes on this project, our project provides a basic level of telepresence that is sufficient in this prototype phase.

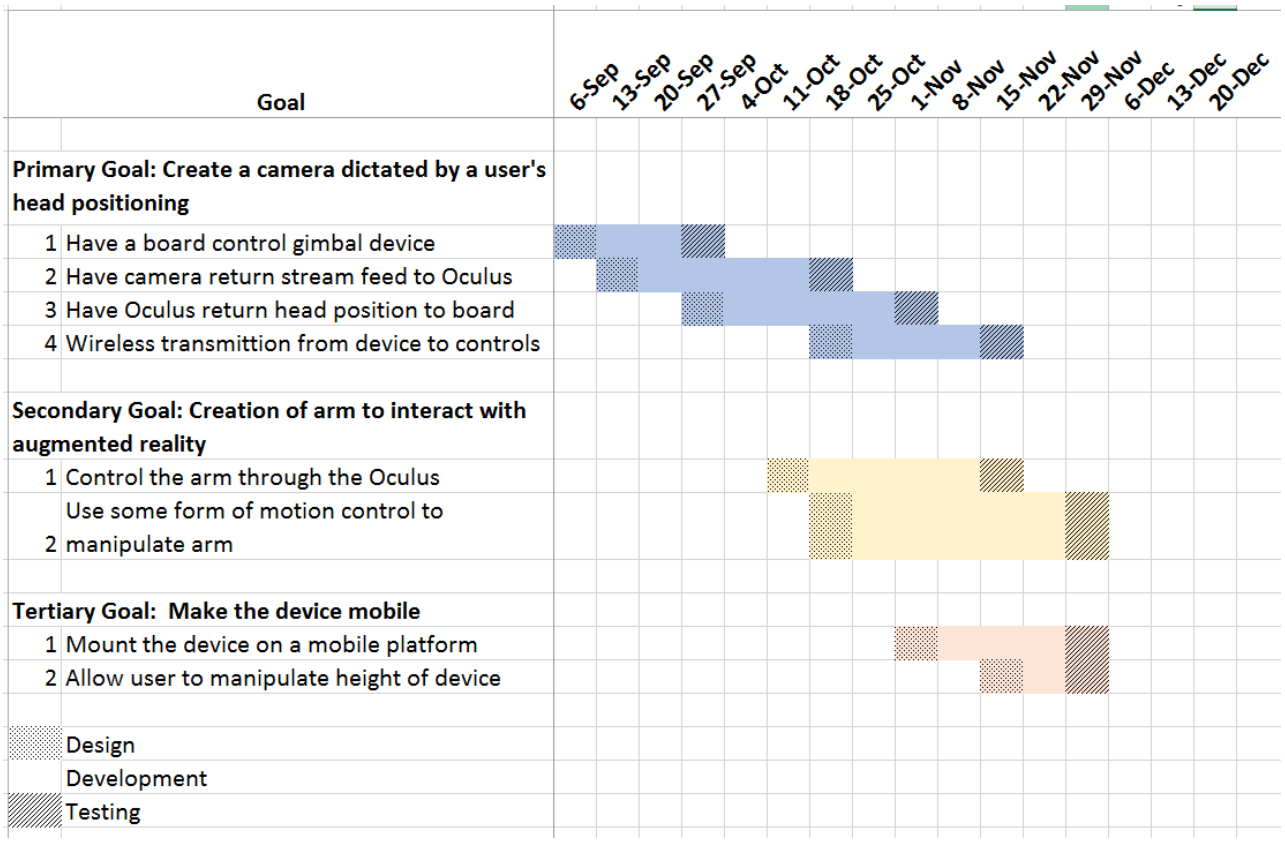
This project turned out to be a much larger undertaking than we anticipated during the design process. Our overall architecture proved to be mostly sufficient for the project but we ended up having several difficulties with details we hadn't considered. Because of these unforeseen factors, we were unable to meet the secondary and tertiary goals we set for ourselves at the beginning. The foundation for meeting all of those goals is in place, but we lacked the time to implement all of them.

Because of this, we believe that future Capstone groups who perhaps lack a concrete idea of what to do can take our project as a template for their own. With another four to six months of development, the secondary and tertiary goals can be easily reached and it's possible to even add more functionality on top of what is currently planned! While there would be a few key pieces that they would have to acquire on their own, much of the project itself is still intact in the ECE Capstone lab. We hope that the design that we outlined in this report can be of use to any prospective groups looking for inspiration.

Division of Tasks

	Andrew	Sean	Russell	Dom	Nasir
Board/Gimbal Control	X			X	X
Camera Stream		X	X		
Oculus Head Tracking		X	X	X	
Wireless Transmission		X		X	
Arm Control	X		X		X
Motion Controls		X	X		
Mount Device on Mobile	X				X
Lift Control	X			X	X

Timeline



Appendices

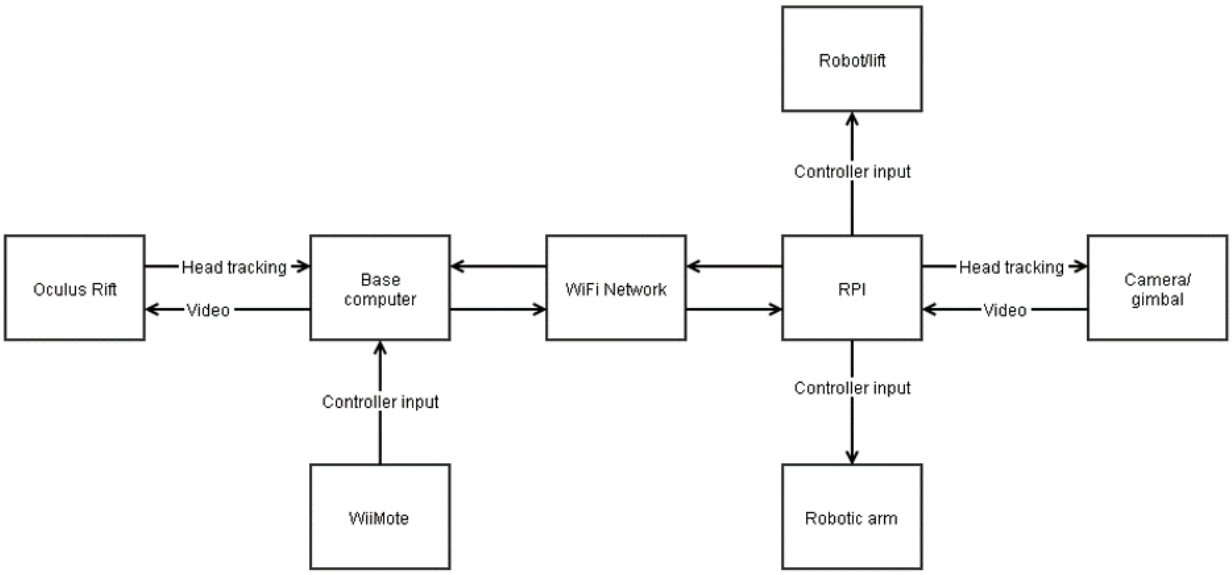


Figure One

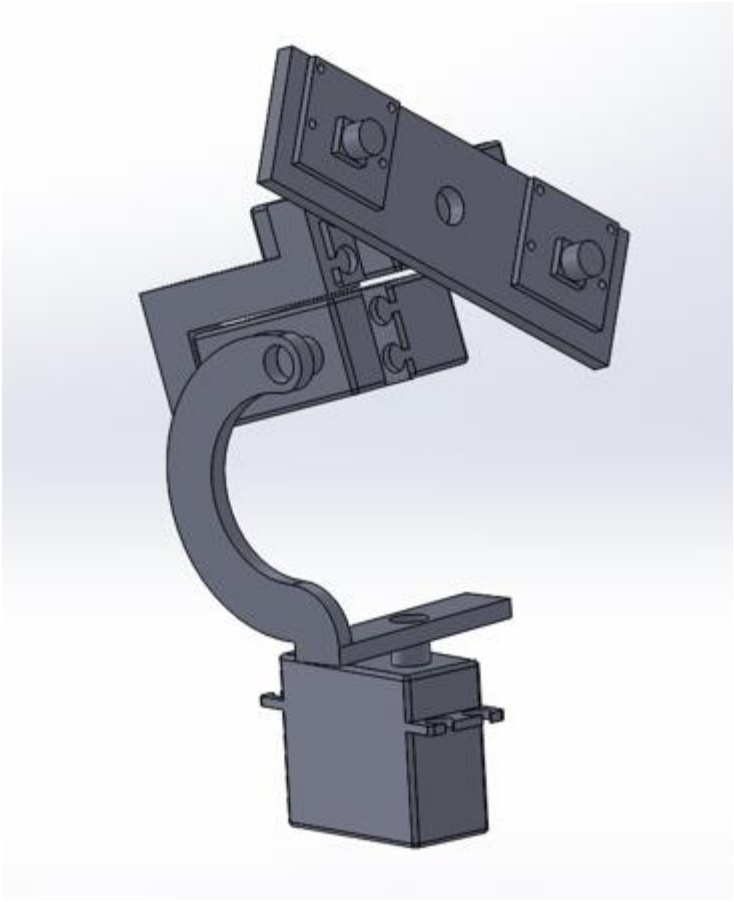


Figure Two

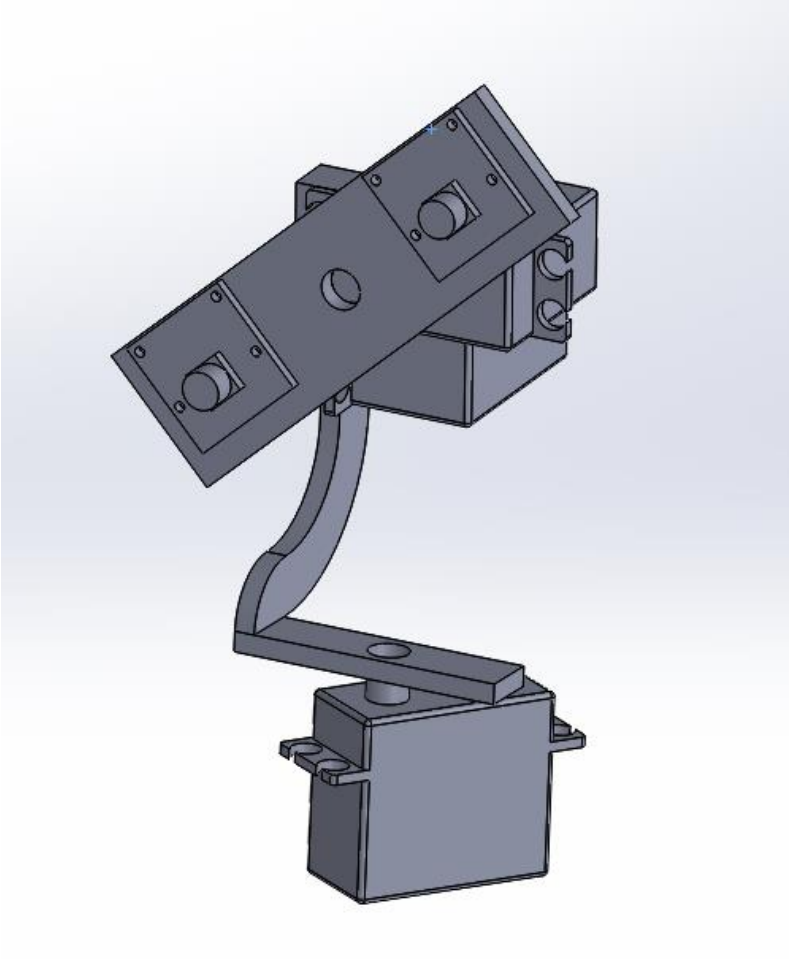


Figure Three

References

[1] G. Ho, N. Pavlovic, R. Arrabito, R. Abdalla, *Human Factors Issues When Operating Unmanned Underwater Vehicles.* Toronto, Canada: Defence R&D Canada, March 2011, pp 3, www.dtic.mil/dtic/tr/fulltext/u2/a555588.pdf