Beaulier, Alex J.

Portland State university

ECE 506 VR Project

# Acknowledgements

Professor Roy Kravitz for approving of this project.

Libraries

Arduino Serial Library

SteamVR Asset Library

[Unity Serial Asset Script](https://sirwilliam.hashnode.dev/serial-communication-between-arduino-and-unity-3d)

Contents

[Acknowledgements 1](#_Toc111212040)

[Summary 3](#_Toc111212041)

[Overview 3](#_Toc111212042)

[Background and Motivation 3](#_Toc111212043)

[Deliverables 3](#_Toc111212044)

[Theory of Operation 3](#_Toc111212045)

[Architecture 4](#_Toc111212046)

[Skeleton Code Arduino 5](#_Toc111212047)

[Installation Instructions 5](#_Toc111212048)

[VR – See Unity Installation Instructions Website 5](#_Toc111212049)

[Arduino – See Arduino Website for Details 5](#_Toc111212050)

[Results 5](#_Toc111212051)

[Discussion 6](#_Toc111212052)

[Current Errors 6](#_Toc111212053)

[Future Improvements 6](#_Toc111212054)

[Conclusions 6](#_Toc111212055)

[Appendix A – Result Pictures 6](#_Toc111212056)

[Appendix B – Result Videos 6](#_Toc111212057)

[Appendix C – Github Repository 6](#_Toc111212058)

# Summary

The proposal is to build an IOT Virtual Reality Environment with a camera live feed hooked up to a microcontroller, computer or GPU. The goal of this project is to learn how to interface a camera and eventually other components from a physical system to a virtual reality one. The inspiration for the project is to build a future work environment for blue collar workers to maintain a fleet of autonomous vehicles or drones. One could observe each vehicle similar to a security room full of CCTV’s and the cost would be significantly less. Various stats could be displayed around an instantiated camera such as safety, operation level, critical warnings and feedback from vehicle peripherals such as maintenance status, engine status ect…

# Overview

## Background and Motivation

A combination of VR and ECE courses provided a baseline to integrate a game engine into a telerobotics platform. As the fields are maturing a project involving both microcontrollers and VR appeared of interest to combine both fields.

## Deliverables

Functioning VR environment with a camera display and one status item. Will include a brief one page or longer report and other pages for instruction steps to build the environment as well as connection for the device.

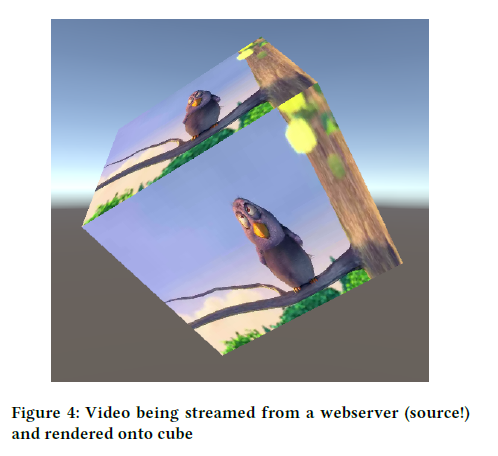
Grading

Instructor Crouse Grading: Final deliverables shall be graded on A-F basis. Scale and background of grading TBD.

# Theory of Operation

The operation of the device is to get messages to a microcontroller and take in the user’s physical movements, keypresses and view to a separate platform which can control a robot. The user can move in VR space which will be tracked by the game engine, press using the built-in game controller which can signal virtual buttons. The virtual buttons are linked to C# scripts which can run on Windows or Linux machines and send messages out to the host PC’s COM ports through serial communication. The messages sent across are read by a microcontroller which in our case is an Arduino. The Arduino parses the message, takes appropriate programmed actions such as turning on the LED and sends the message back. The bidirectional communication allows Unity’s scripts to parse the received message and by linking the classes public members to UI elements such as a textbox, the received message is displayed to the user. The VR space has preprogrammed assets linked to the scripts and buttons which can be seen in the appendix for accessing and running the scripts class members such as ping, refresh and connect. The user first selects a com port in the VR UI element, then clicks the UI connect element and then pings(sends message) to the Arduino to turn on the LED.

Another aspect in the theory of operation is the VR camera for virtual operator live feeds. I have attached a file regarding all of the various methods explored and tried such as webcam, integrated laptop camera using Vuforia and wireless webserver rendering. Currently, the Vuforia asset is the only plugin which was operable and is only run in a 2D game ie the VR camera rigs are disabled. For this to run the user selects the Vuforia assets as enabled, starts the game, and the integrated laptop camera is displayed on an auto generated canvas from the plugin. In theory the operator will have a non-auto rendered asset that is usable in VR to render as a texture onto any asset such as the cube found in the Melle’s research paper. This would allow for scaled screens as originally desired in a CCTV like room. One could also display the view as the main camera instead and attach servo motors for 3DoF controls in a submodule that could be exited. An operator could access a submodule vehicle for direct robot view as a sort of menu navigation and exit to a regular VR viewing setting in the future. It would be analogous to driving a car versus walking where the view and controls methods separate for where the user is.



# Architecture

The architecture is shown below. Note the RaspberryPi could also be used instead of the Arduino. It was chosen for it’s webcam feasibility(project did explore ov7670 as well) and built in WIFI connectivity however due to serial communication the Arduino was preferred for reference code. The Webserver portion connected to the RaspberryPi worked however the client side to the Unity Game Engine required additional scripts to pull the webserver data. The user also gets haptic feedback and visual input from the game engine. The laptop hosting the engine could be used to host multiple devices and have additional clients as a server model rather than a single user. The only webcam in the diagram working in the demo provided is the webcam directly connected to the laptop.

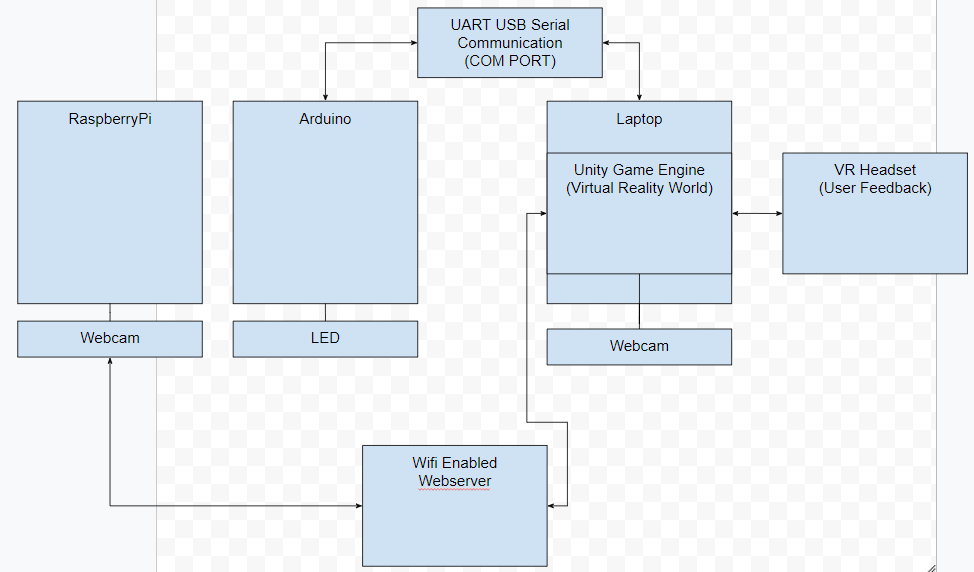


Figure 1: System Architecture Telerobotics Platform

# Skeleton Code Arduino

Setup Pins

Establish Connection

While loop until value is read

Print to Terminal

Repeat Message Back to Connection

# Installation Instructions

## VR – See Unity Installation Instructions Website

-See additional attachments within Github for specific issues

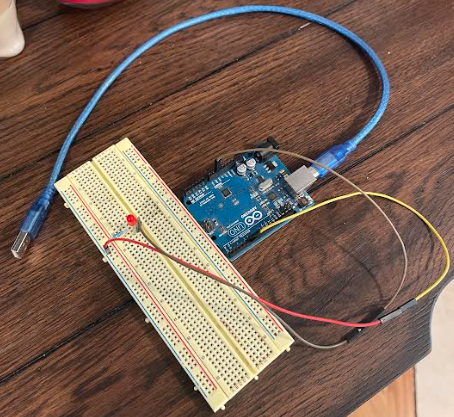
## Arduino – See Arduino Website for Details

-See github for code

# Results

The platform worked for the deliverables requested. The serial communication worked flawlessly for the LED to display when reading a message. The status of the message was sent back and received for fully bidirectional communication between a VR user and a microcontroller. A major drawback in the project was the webcam however the deliverable was met just not as one may expect. The webcam was visible and it was shown an asset is capable of taking in webcam data however a wifi enabled remote webcam is desired as well as scalable assets. The RaspberryPi camera was broadcasting to a remote server however the Unity client side to pull the data needs additional work.

The operational LED circuit is shown below.



The specific rig used is a Valve Index for it’s great FOV, reliable detection(blind spots almost non-existant) and industry leading haptic feedback for gripping and signaling. No other setup comes near the quality and benefits as this rig currently provides.



The VR world is shown below with significant assets shown. The operational status display message is shown below with the VR world. The response text is overwritten with the last message received from the Arduino. The dropbox shows available COM ports and the UI boxes are connected to the C# script written in Unity.



# Discussion

## Current Errors

The webcam in Unity uses a pre-scaled asset as a full canvas image. Though not an error it is not desired for a VR platform and is only available for a 2D VR user rather than a VR operator rig.

## Future Improvements

Based on discussions and explorations the next steps in this project should be video transmission protocols based on webRTC. There are beta exploration packages in which a user can connect the server setup from the RaspberryPi and turn the video data into a render texture within Unity. The texture can be applied to any object as a material and the video can be displayed. Note this is very similar to some of the webcam tutorials found only this would require setting up the webRTC for it’s data rather than the webcam.

Second, development of wifi communication should be done for tcp/ip communication. As the webRTC sends video audio and some signaling it lacks in an ack/knack response for emergency signals. TCP/IP can guarantee a signal and the developer should also consider a TLS or certificate authority for authorized access of the system.

## Conclusions

*“Functioning VR environment with a camera display and one status item. Will include a brief one page or longer report and other pages for instruction steps to build the environment as well as connection for the device.*”

The deliverables requested were met and all sources were documented. The camera display, bidirectional feedback message and led all worked as scoped in the project deliverables. Further development should be applied for emergency signals, wifi channels and video streams using webRTC as this project lays a groundwork for further implementations. The learning curve of Unity development was met while taking the VR course and the deliverable is accomplished.

# Appendix A – Result Pictures

# Appendix B – Result Videos

See Ehsan Aryafar for recorded demo video

# Appendix C – Github Repository

<https://github.com/ajbeaulier/ECE_506_Telerobotics>