**Team: ECE 409**

**ELECTRICAL & COMPUTER ENGINEERING**

School of Engineering 

**Interim Project Design Report**

**Modular Wireless Xbox Kinect Data Transfer Device**

**Date: December 8, 2017**

**Advisor: Yuichi Motai -** [**ymotai@vcu.edu**](mailto:ymotai@vcu.edu)

**Names: Millad Nooristani, Eric Olson, Ayush Rohatgi, Dat Doan**

**Signatures**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table of Contents

**Executive Summary2**

**Project Brief/Scope3**

**Engineering Design Specifications9**

**Design Research10**

**Design Concepts13**

**Proposed Design13**

**Plan of Work17**

**References19**

**Addendums20**

**Executive Summary**

This project seeks to alleviate issues a user may encounter when trying to set up multiple Microsoft Kinects with their PC. For example, a previous computer vision project encountered the problem of having to install multiple USB hub PCI cards; instead of having to install multiple cards in a computer, a user would rather have an easy, modular solution. In our project, we will virtualize a Microsoft Kinect on a host PC by transmitting its data from a Raspberry Pi. The Kinect is connected to the Raspberry Pi, and the data is received through a program interface. The Pi will then transmit the data to the host computer via WiFi connection to a router on the same local network as the host computer.  
 Our project sponsor is Dr. Yuichi Motai. His specialty is in robotics and computer vision, and therefore, is helping us form realistic and informed project goals. The project is being made to help our professor and his researchers use and set up projects involving multiple kinects, wirelessly, and without having the obstacle of installing a PCI card for each Kinect.

Our final deliverable will be a system that allows a Microsoft Kinect to be plugged into a Raspberry Pi that transmits Kinect data to the host computer via high speed wireless connection. The host computer then separates the information it receives, and virtualizes the Kinect. Through this, a user should be able to connect, within memory limitation, as many Kinects as they need using multiple Raspberry Pis. The project sub-deliverables include: a software which parses the raspberry pi data, a software which virtualizes the kinect using this data, an API which sends the information from the Pi to the computer, and an apparatus built with the Raspberry Pi that allows for each connection of the kinect.

Our final deliverable will be demonstrated by means of a people counter using facial recognition software. The cameras will be set up in key locations around the expo. The facial recognition software will tag individuals as they enter the field of view and catalog them. The software will be demonstrated to an individual by having the individual look into a kinect at the home station and be able to see when the individual entered the expo center. This will demonstrate the power of being able to use multiple Kinects on the same system, while also showing the versatility of having a wireless network of communication from the Kinects to the host computer.

**Project Brief / Scope**

Description:

Our project is looking to solve the issue of the difficulty in setting up multiple Microsoft Kinects with a single system (PC). In its current implementation, a computer requires a separate USB hub/card per Kinect, due to the high bandwidth required. This is generally inconvenient, and not a portable solution. Our goal is to send the Kinect data through a wireless connection, that would allow the Kinect to be virtualized on the host PC, and thus remove the need for extraneous USB hubs. The need for these would be replaced with a requirement of a Raspberry Pi per each Kinect. Though still not the most convenient, this solution is much cheaper, and much more portable and easy to setup.

As a consumer, if I want to use multiple Microsoft Kinects to collect data from different views, I have to pre-plan and ensure that I first have a computer capable of connecting multiple auxiliary USB hubs. Furthermore, I lose the aspect of portability in this case, as I then have no way to use my laptop, which is the most common type of computer in use by consumers. These constraints could be enough to dissuade me as a consumer from even experimenting with the Kinect, and instead finding different tools (which may be more expensive and overall unfriendly for me to access as a casual user). If instead, I could find a modular solution, I would be more inclined to deal with some minor inconveniences, as my overall workflow would not be mired. This is the problem our project is seeking to fix, and we hope that we can achieve a good, user-friendly solution.

Our project sponsor is Dr. Yuichi Motai, a professor in the Department of Electrical and Computer Engineering here at VCU. Dr. Motai’s research is in the computer vision, which is the general field related to the Microsoft Kinect. To further existing projects, he has found the need to set up multiple Kinects in a modular fashion. He ran into this need as he was switching labs and upgrading computers, causing the project to move between computers. The problem is, that not every computer has the multiple hubs required to deploy multiple Kinects. Dr. Motai is excited by our project scope and is helping us by providing insight into issues we may run into during the time of deployment (i.e. bandwidth constraints etc) and their solutions.

Though we do not have any other faculty directly working on this, we think that this project could be used positively by other researchers and students. It will allow for the cost and inconvenience of deployment to be diminished, and thus allow more interesting projects to be created. We look forward to seeing how others may creatively use our product.

Goals/Objective:

We reached the final scope of our project after a few iterations. We initially sought to create a hardware based USB hub that would actually split the signal on the computer side. This idea was criticized as being too lofty due to the lack of experience we had in electrical design. Our final idea was realized due to the availability of materials we had in our lab, as well as input from our sponsor regarding project feasibility.

Our goals and deliverables for the project follow as:

· Develop a software to interpret signal from the Raspberry Pi

· Adapt existing Raspberry Pi Microsoft Kinect API to allow for the transmission of data wirelessly

· Develop a software to virtualize Microsoft Kinect

· Develop a solution to decrease the signal information coming from the Microsoft Kinect/Raspberry Pi

· Ensure functionality by implementing this solution with an existing project that requires multiple Kinects

During this process we may need to change some components as we realize their constraints; maybe the Pi will not be able transmit the data fast enough, maybe the Kinect transmits too much information. We must, and are, prepared to make design changes as the project progresses. An example of this is that, in an earlier iteration, the wireless communication protocol that was used to communicate from the Pi to the host computer was Bluetooth 4.0. We decided to make a design change and use WiFi on a local network as our communication protocol due to its increased reliability and speed. We hope to keep the project flexible enough to allow this, so that late stage design changes do not negatively impact our workflow too much.

The end goal of the project is to create a system involving hardware (Raspberry Pi, Microsoft Kinect) and software (Interpreter software on Windows) that allows a Microsoft Kinect to be initialized virtually instead of through direct connection via USB. The project can be defined as complete when we can do this, and also allow for multiple such systems to be deployed on the same PC. Additionally, these virtualized Kinects should work well in existing projects.

Location/Environment of Use:

Our product will be used in various places. This includes homes, offices, hospitals, and schools/colleges. Each place has its own environment characteristics that our product must take into account. For instance, the characteristics of a home environment is consider low foot traffic in terms of people present. Has a medium or low interferences levels from appliances and connected devices in the household. A hospital is a fast pace with high amounts of foot traffic and susceptible to airborne illness. It has a high interference level from the several medical devices in each patient’s room. The office and schools/colleges also have high amounts of foot traffic and interferences coming each employee/student connected devices, ranging from their computer to their smartphones.

The Microsoft Kinect will only perform facial recognition and movements in the environments as stated above. Other products being used with our products is undetermined as of right now, but will be dependent on our environment.

Target Audience:

This product has a wide user base that includes research applications as well as commercial and personal use. The research applications are numerous, as many Microsoft Kinect sensor based projects have been completed in the past and are currently in development. The Kinect is an extremely powerful piece of hardware, so it is natural that many engineering institutions and universities have taken an interest in finding new ways to utilize its full capabilities. Most of these projects revolve around using and improving the Kinect’s interface with Windows 10. The product that we are creating will be of great interest to many future researchers in this area, as the Kinect does not have any innate wireless capabilities. Once we have a finished product, the WiFi attachment will lead the way for projects make use of both multiple Kinects, and projects that require the Kinect to not be hampered by physical cord attachment.

Commercial appeal for this product also exists. There are already many privately owned Microsoft Kinects in circulation, due to their video game entertainment capabilities with the Microsoft Xbox. This product will appeal to this group of users, as it will allow for home use of a wireless Kinect with a Windows 10 machine. The user will be able to implement several different, already existing, home projects that use wireless and multiple Kinects. This is also an economically viable option for consumers, as the kinect is a high resolution camera that can be cheaply converted to a wireless high resolution camera with Windows 10 interface with this product.

Target Market Information:

The number of potential users for this product is not easily calculated, due to the fact that Microsoft has declined to release the total number of Kinect sales. However, the number of potential users is numerous, as any person or institution with ownership over a Kinect is a potential user.

The potential uses of this product are too many to count, and new uses are sure to develope. This is due to the fact that any previously existing use of the Kinect with Windows 10 will now be able to be completed using WiFi. As an example, the most obvious use of this product is to use wireless Kinect cameras to set up a network of surveillance for any purpose that the user desires.

As stated above, the current market is difficult to nail down without significant assumptions. The earliest known data is that 24 million Kinects had been shipped out by February of 2013. Assuming that the product can be produced for $30, the maximum market size is $720 million dollars. Obviously, the mass majority of users will not purchase this product, but there is still a sizable market.

Product Constraints:

There are constraints for this product. The most prevalent of which is the wireless data streaming speed limitation. The Microsoft Kinect is an extremely high resolution camera with several other sensors such as motion tracking and facial recognition. All of this data is sent at an extremely high bitrate. WiFi capabilities are at a maximum 10 Mbps. Due to the massively slower capabilities of WiFi, much of the data will be dropped and will not be able to be transmitted to the Windows 10 machine.

Other constraints are the operating systems that can interface with this product. The specific product created is a physical attachment for the Microsoft Kinect, and will only interface with the Microsoft. The product also can only interface and transmit data to a Windows 10 operating system. There are no plans to develop support for other operating systems or devices at this time.

The cost limits for this product are more an issue of consumer prices. The physical components of the product are relatively cheap, but if the product is too expensive, certain parts of the market will opt to find another product to serve as a wireless camera.

The only specific material required is the Raspberry Pi Apparatus that will be used to limit data and transmit through WiFi to the target machine. The container for the prototype of this product will be 3D printed. A specific container manufacturing technique for the commercial product has not been determined at this time.

Current Products:

There are no products competing with us in a direct way. I would say that the current products available are other wireless capable cameras and using Microsoft Kinect with a wired connection.

The benefit of a wired connection with a Kinect is that there is no data limitations. The benefit of a different camera to use with WiFi is that it is significantly cheaper than purchasing a Kinect and our product together. Due to this, consumers who do not already own a Microsoft Kinect were not considered as part of our potential market. The shortcoming of wired connection is that it is difficult to use more than one Kinect with a single Windows 10 machine. Another shortcoming is the limitation of having to have an established wired connection. The current cost of a wireless camera depends on the quality that the user desires. A camera that our product will be able to compete with is priced at roughly $100-$150.

Value Created:

The user can measure the value that they received from the product through the projects that the user can now pursue, as well as the cost saved from buying an entirely new, compareable wireless camera. This will create an economic value for commercial buyers, and will create an academic value for buyers interested in using the product to conduct further research. We value above all the fact that this product will push the capabilities of the Microsoft Kinect in a newer direction that is mostly undiscovered. We believe that these new capabilities from this product will pave the way for more progress on projects relating to the Microsoft Kinect.

**Engineering Design Specifications**

Humans will interact with our product purely as a safe and legal manner if the user follows the specifications provided. Some factors that have been considered are: Safety, manufacturing, operation, and maintenance. Safety: the product serves as a relatively lightweight addition to an existing product. The safety concerns are minimal and are similar to other small electronics. Manufacturing: The manufacturing of the product is simplistic. The product is simply a microcontroller housed in a plastic shell. Operation: The operation of the product will be kept basic for the user’s sake, with most of the connecting being done automatically by the software. Maintenance: software updates can be provided for the OS side. Hardware maintenance will be up to the user.

Customer satisfaction will be based off of if the customer found as much value from the product that they perceived during purchase. The value that the user derives will be based on what use the user finds. The potential uses of this product are too many to count, and new uses are sure to develop. This is due to the fact that any previously existing use of the Kinect V1 with Windows 10 will now be able to be completed using WiFi. As an example, the most obvious use of this product is to use Wifi Kinect V1 cameras to set up a network of surveillance for any purpose that the user desires.

This project seeks to alleviate issues a user may encounter when trying to set up multiple Microsoft Kinects with their PC. Instead of having to install multiple USB hub PCI cards in each computer they want to use, a user would rather have an easy, modular solution. In our project, we will try to virtualize a Microsoft Kinect on a host PC by transmitting it’s data from a Raspberry Pi. The Kinect is connected to the Raspberry Pi, and the data is received through an API. The Pi will then transmit the data to the host computer via Wifi.

Video quality, connection range, data compression and decompression speed, gui quality and software responsiveness.

The product will be a rectangular prism. The product will be lightweight with dimensions of approximately of 9x6x3 cm. The product is of electronic nature, there will be no moving parts. The only materials used are going to be 3D printed rigid plastics. The Raspberry Pi 3 will have green, blue, and red LEDs flashing on the board. The green LEDs will represent when the Raspberry Pi is connected to the Kinect V2 or Kinect V1. The blue LEDs will represent when the Raspberry Pi is transmitting data via Wifi. The red LEDs will represent when the Raspberry Pi is disconnected from Kinect V1 or V2 and when it's disconnected from the Wifi enable computer. The only component of the prototype is the shell. The shell is going to be 3D printed plastics. The other components are purchased and programed. When the product goes to production, a more elegant solution for the manufacturing of the shell will be created. The shell must be able to effectively and safely secure the Pi microcontroller, while still remaining elegant in aesthetics. The shell must be able to effectively and safely secure the Pi microcontroller, while still remaining elegant in aesthetics. The product can be shipped in standard cardboard packaging with the standard safety methods.

Brief cost analysis

Microcontroller - $35.00

Shell - $2.50

Prototype cost: $57.50

Some constraints include: The product must deal with the constraints of two other products. The first being the Xbox Kinect. The constraint that this brings is the massive amounts of data that the Kinect collects and puts out. This constraint is solved by using compression and data mimicking techniques. The second constraint is the Windows 10 operating system. This constraint is worked around by designing the software from the ground up in Windows 10 to ensure compatibility.

**Design Research**

Effective Compression of Range Data Streams for Remote Robot Operations Using H.264

* This IEEE article on Effective compression of range data streams for remote robot operations using H.264 goes over how transmitting raw sensor data over a low bandwidth network is problematic as the data stream of sensor data is often large. This will aid us in learning how to limit/compress the stream of data originating from the Kinect. Since we will be using WiFi connectivity, which has a low bandwidth of up to 10 MBps and the Kinect data streams produces up to 48MBps, this article goes over the issue of transferring large data packets and compressing them without reducing the quality of the data stream. In addition, this will provide us reference on the compression algorithm and being able to determine what type of data we need to limit/compress.

Kinect API Overview

* This article is the application programming interface for the Microsoft Kinect. The main subsections of this article are video data and audio data. These are the two outputs of the Kinect sensor. Though the audio section does not detail much, the video section covers the bulk of the contents. The API basically describes the data sources, streams, stream readers, and various frames for the video output. In each section you can find the description and utilization of different functions, properties, and events that the Kinect has implemented in its software. The description of each type includes the syntax for JavaScript, C#, and C++. Just like any API, there are the input, return, and property values included in the description.

2013 IEEE Hot Chips 25 Symposium (HCS)

* This literature from Microsoft is the layout of the Main System on a Chip and Xbox One Kinect. The article shows the specific ways that the Xbox interfaces with the Kinect, as well as specific SoC components. The individual sensors on the Xbox One Kinect are also described in detail with how the information is physically gathered. This is relevant to our project, as a core part of our product is an Xbox One Kinect. This sheet will serve as valuable information on how the inner workings of the device are connected.

Raspberry Pi 2 Kinect 3D scanner

* This article provides us insight on how to initialize the Kinect with a Raspberry Pi and being able to stream/capture RGB and depth videos from the Kinect. In addition, this article provided us an introduction to Libfreenect, which is a software/library for accessing the Microsoft Kinect USB camera that includes all code necessary to activate, initialize, and communicate data with the Kinect hardware. Includes various drivers and a cross platform API that functions on Windows, Linux, and OS X.

Kinect RGB Demo V6

* This demo was developed by RoboticsLab and aims to provide a toolkit to start utilizing the Kinect data and develop standalone computer vision programs without the need of integrating existing libraries. The RGB demo features include, grab kinect images and visualize / replay them, support for libfreenect and OpenNI/Nite backends, extract skeleton data / hand point position (Nite backend), integration with OpenCV and PCL Multiple Kinect support and calibration, calibrate the camera to get point clouds in metric space (libfreenect), export to meshlab/blender using .ply files, demo of 3D scene reconstruction using a freehand Kinect, demo of people detection and localization, demo of gesture recognition and skeleton tracking using Nite, demo of 3D model estimation of objects lying on a table (based on PCL table top object detector), demo of multiple kinect calibration, and Linux, MacOSX and Windows support

**Design Concepts**

Some initial design concepts we brainstormed was applying our model as a 3D room

scanner, a facial recognition tracker that provided a live feed displaying its process, and lastly, a handheld camera/gun that could be applied to a video game, for example.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Design Concepts | Wide consumer usage | Cost effective (low cost) | Marketable | Hardware intensive | Feasible |
| 3D Room Scanner |  | **X** |  | **X (depends on # kinects)** |  |
| Live Feed Recognition | **X** |  | **X** | **X** | **X** |
| Handheld Camera |  | **X** |  |  | **X** |

Table1: Comparison Matrix

**Proposed Design**

As stated in the previous sections, the core of our design is based around putting together existing components, and building a software solution around them. The physical requirements to realize our solution include components that we already have and that we had to order. These include: A high-performance laptop, three Raspberry Pi 3 Micro-computers, a wireless router with port switch functionality, and three Microsoft Kinects V1. The goal of the project is to transmit data from the Kinect to our computer using the Pi as the middleman. In order to do so, we are also have certain software requirements. We require the use of frameworks such as OpenCV, and language support with Python and C#. Our solution seeks to have the Raspberry Pi receive the Kinect video feed, and strip out features that we may not be looking for, such as an Infrared map. Once captured, the video feed, along with other parameters such as depth, will be transferred over a Wi-Fi/Ethernet based connection. Once received by the host machine, we can feed this data into existing Kinect programs, and make use of multiple Kinects in a single program.

The key component that allows this to work is the Raspberry Pi 3. Fitted with a dual-core ARM processor, the Pi lets us send information between the PC and Kinect with high level code and software, and also lets us monitor performance through hardware. One metric to determine the quality of the feed being sent is the frames per second, or FPS. At a low FPS, 1-15, the video feed is choppy and is of no use to the user. The lack of fluidity in the video will not allow for the continuous capture of movements. After reaching around 20-30 fps, we can consider our output to be satisfactory. 24fps and 30fps are standard benchmarks for video speed, and in conjunction with the fact that other data is being transferred (thus consuming bandwidth), we expect these values as optimal maximums. We expect to keep, at a minimum, a stable 20fps.

Another metric for measuring the quality of content being sent is packet loss. Our system is sending over large amounts of data at a time. We initially chose Bluetooth as our protocol of choice, due to the perceived ease to set up a communication between two devices. While we found that the initial pairing is easy to do, the transfer of unsigned (raw) data is non-trivial. In addition, the speeds and latency suggested that we would not be able to reach the optimal results we were seeking (in regards to fps). We are instead using Wi-Fi/Ethernet to transfer our data. With this change comes the added benefit of easy to track packet loss issues. We can monitor the traffic on our router, and see what would cause a signal to no longer propagate without error. The rate of packet loss helps can be correlated to signal strength and speed, and provides an intuitive start for the fixes to our problems.

The Rapsberry Pi is a running a Debian based distro of Linux entitled “Raspbian”. Due to the nature of linux application compilation, we are able to compile most external libraries and applications on the device. This prevents the installation of incompatible frameworks (at an architectural level) that could mess up not only our program, but our hardware as well. Within Raspbian, we are planning to transfer output data which would be generated from a Kinect viewer application, such as RGBDemo. This output data is stored on the Pi, and can be stored in a cache folder for quick updating and erasing.

Our host machine will be running Windows 10, which provided native driver support for both the Kinect 1 and Kinect 2. Due to hardware limitation on the Pi 3, we are initially implementing Kinect 1, with the possibility of upgrading to the Kinect, pending how finicky the Raspberry Pi is with the Kinect, The Host machine will either virtualize drivers for the Kinect or provide a raw video feed with log data attached as well.

The Microsoft Kinect, which is the core feature and accessory being worked on in this project, is a cheap 3D and depth camera with applications in research, games, and virtual/augmented reality. The Kinect V1 makes use of USB-20 (with a speed limit of 480 Mbps), while the Kinect V2 uses USB-3.0 (with a theoretical max speed of 1 Gbps). Given the limitations of what kind of hardware the Pi support, we chose a solution that allows us to adapt as needed.

For the application of our solution, we are seeking to create an interactive experience, that will show the power of the Kinect. The core of our project leans towards Machine Learning, specifically Computer Vision. Using the recognition power of the Kinect, and the computers that handle them, we are seeking to craft a demo which can track individual users, or at least a group of users, throughout any event. At a event like the career fair, engaging people with an exciting project is part of the pleasure of making a senior design project. By creating a system where users can see when they enter and when they exit an environment, we engage users to come and interact with our setup. The key purpose of this solution, is to provide modularity to our Kinects, and this application could not prove it better.

Our software solution will be a mix of C++, C#, Java, and Python. We must use C#, as the goal is to make a product of a Microsoft branch work with another Microsoft product, but through an interface (the pi). Low and High Level Windows development is done in C++ and C#, so a portion of solution with be defined and compiled in C. The Pi is friendly to all languages, and compiling C++ programs on the Pi will allows us to have a uniform code base. Python and Java may enter our developer toolkit, as these provide great scripting support.

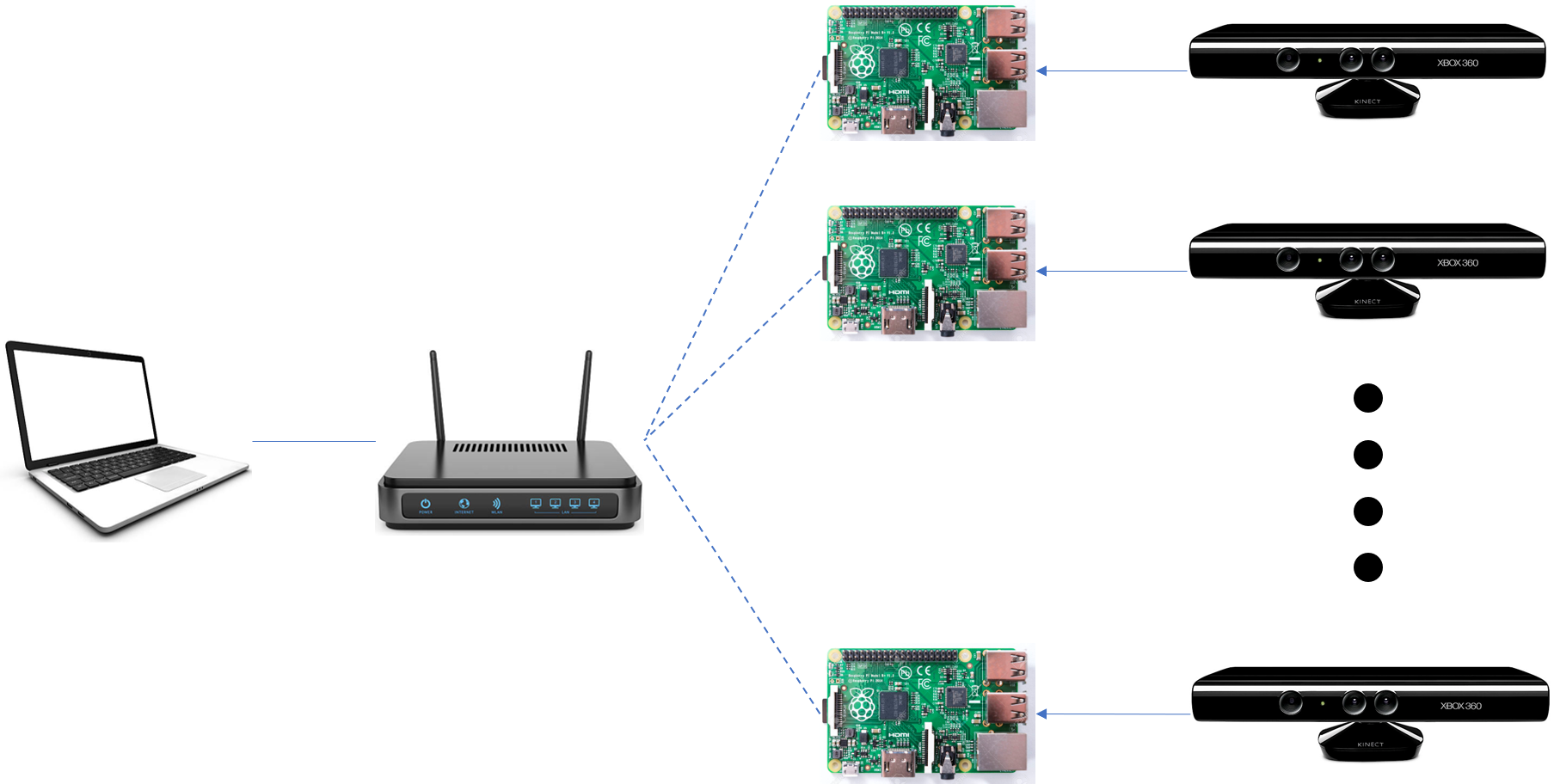


Figure 1: Representation of the completed network

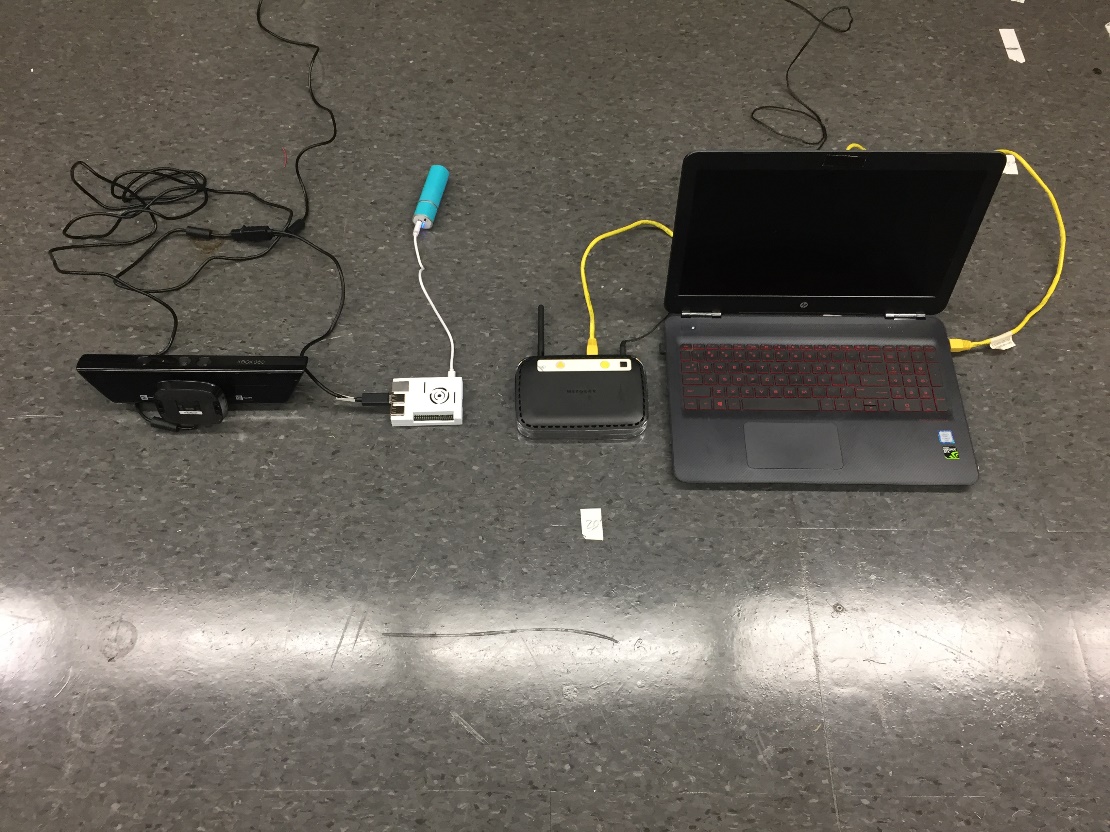


Figure 2: Picture of the system with one camera

**Plan of Work for ENGR 403**

In order to have successful completed prototype preview, we will follow our Gantt chart timetable and update as accordingly, refer to the Addendums. Our goal is to spend no more than two weeks or less per activity on the Gantt chart, unless stated otherwise. In addition, along with the Gantt chart, we have identify our two major issues that we must overcome for us to proceed to final production. We have decided to allocate two group members per issue. Below are the following issues we have determined to be our highest priority,

1. Capture/Locate Data Streams Info from the Kinect RGB Demo (Partners Ayush & Millad)

2. Transfer the Capture/Stream Data on the Kinect connected to the Raspberry Pi 3 to a Computer Over Wifi (Partners Eric & Dat)

Informing Technical Advisor:

- Our technical advisor will be informed by our weekly emailed progress memorandums along with handing in hard copies or emailing our progress memorandums to our instructor if necessary. In addition, we have agreed and planned to meet weekly on Thursday at 11 am with the advisor to inform any setbacks, recent milestone achievements or our general overall completion progress. Lastly, we will submit a regulatory report during the 12th week and present our final paper/presentation during the 15th week to our technical advisor for review.

Testing Protocol:

- Towards the completion of our prototype and the final project, we will follow the usability evaluation methods and base on the evaluation results, we will modified our final project accordingly. The usability evaluation method includes the following,

1. Reporting Usability Test results

a. Quantitative Data

i. Success rates

ii. Task time

iii. Error rates

iv. Satisfaction questionnaire ratings

b. Qualitative Data

i. Observations

ii. Problems experienced

iii. Comments/recommendations

iv. Answers to open-ended questions

**References**

Nenci, Fabrizio, et al. “Effective Compression of Range Data Streams for Remote Robot

Operations Using H.264.” *Effective Compression of Range Data Streams for Remote Robot Operations Using H.264 - IEEE Conference Publication*, IEEE, 6 Nov. 2014, ieeexplore.ieee.org/document/6943095/.

Microsoft. “Kinect API.” Kinect API Overview, Microsoft, 2017, msdn.microsoft.com/en-

us/library/dn782033.aspx#ID4E3.

J. Sell and P. O'Connor, "Main SoC and XBOX one kinect," *2013 IEEE Hot Chips 25*

*Symposium (HCS)*, Stanford, CA, 2013, pp. 1-18.

“Demo Software to Visualize, Calibrate and Process Kinect Cameras Output.” Kinect RGB

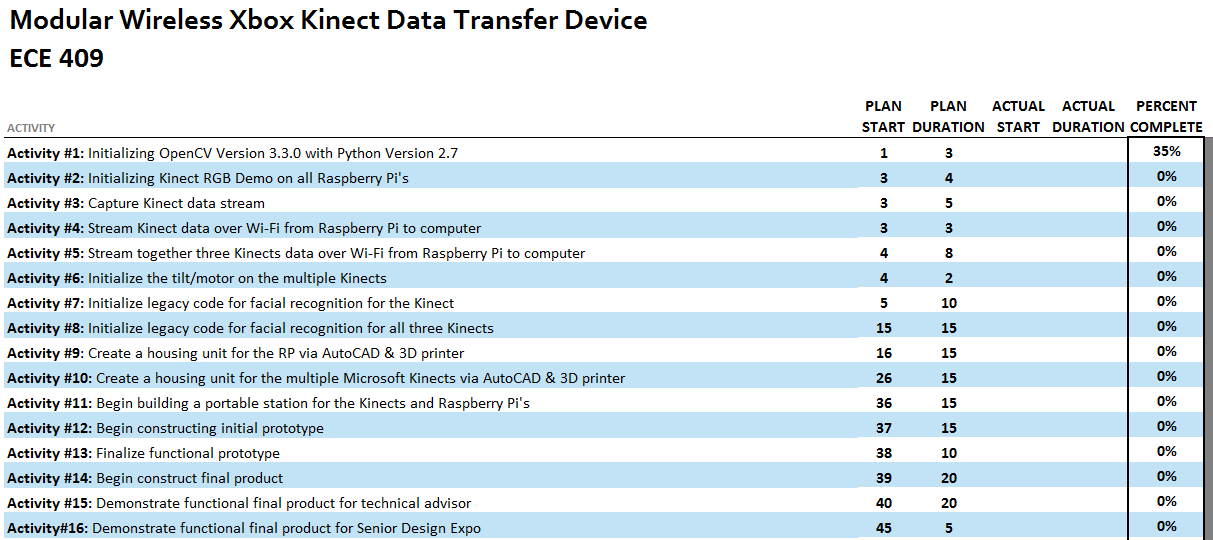
Demo V6, rgbdemo.org/index.php/Main/KinectRgbDemoV6.

Lukas, Mario. “Proof of Concept 3D Scanner with Kinect and Raspberry Pi2.” Blog Von Mario

Lukas, 11 Apr. 2015,

www.mariolukas.de/2015/04/proof-of-concept-3d-scanner-with-kinect-and-raspberry-pi/.

**Addendums**



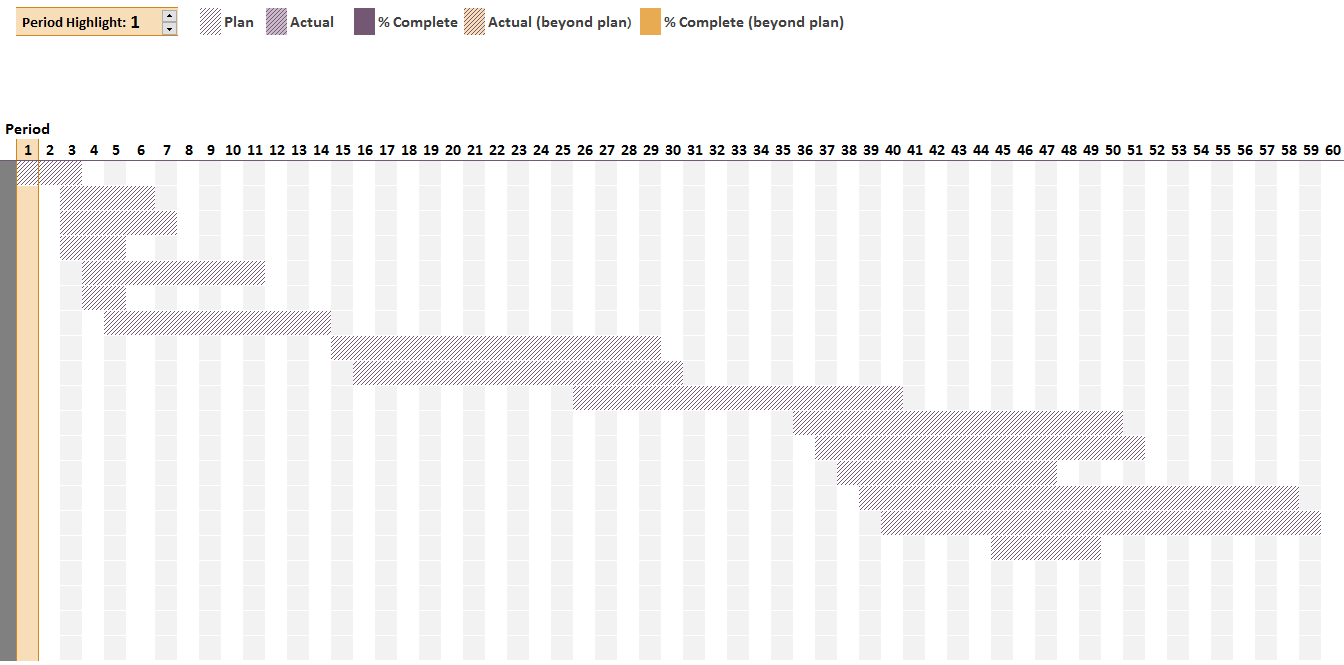


Figure3: Gantt Chart