**ENGR 403 Team: ECE 409**

**Final Project Report**



**Modular Wireless Xbox Kinect Data Transfer Device**

**Date: May 4, 2018**

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

**Names: Dat Doan, Millad Nooristani,**

**Eric Olson (Editor), Ayush Rohatgi**

**Signatures**

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

Table of Contents

**Executive Summary2**

**Project Description3**

**Engineering Design Specifications4**

**Prior Art7**

**Societal Impact9**

**Prototype Development14**

**Performance Testing15**

**Design Solution16**

**Financial-Regulatory Aspects19**

**Discussion, Conclusions, Recommendations23**

**References25**

**Executive Summary**

This report seeks to address all aspects of the Modular Wireless Xbox Kinect Data Transfer Device. The project description describes the project goals and overview of the experiment. This section will go in detail about the purpose and motivation behind the research of this device. Lastly, it will describe the software and hardware that is implemented in our solution at a very high level.

The engineering design specification section aims to summarize the project, background, target users, and target engineering design requirements along with their metrics. In other words, it is a technically competent project description to give the reader an in depth understanding of the goals, purpose, and utilization of the proposed research.

The prototype development section aims to elaborate upon the physical development of the model. This entails the software and hardware used to implement the system and goes into discussion about how each component came together to operate the device. This section is more visually oriented than other sections, therefore, the MWXKDTD system is depicted. All planning sketches, schematics, flowcharts, and photographs can be found in the prototype development section.

Performance testing seeks to describe how the implementation from the previous section was tested. Questions such as ‘how was the design verified? Did the system operate as expected? What steps were taken to ensure the functionality of the device?’ will be answered. This section is essentially an analysis of the results.

The design solution section provides a detailed description of the result of the research. It is a general discussion about the outcome and lessons that were learned from the prototype implementation and evaluation phase

The goal of the financial section is to convey the application of our design as well as financial considerations that went into building the device. Each component is evaluated in a cost vs effectiveness manner in relation to similar concepts of point cloud imaging devices.

Lastly, the discussion, conclusion, and recommendation section will approach the outcome of the research and interpret our results. The user satisfaction, functional requirements, and metrics that were specified in the EDS will also be addressed and recommendations will be drawn from the conclusions.

**Project Description**

Description:

The purpose Modular Wireless Xbox Kinect Data Transfer Device, shorted MWXKDTD, is to counter hardware obstacles faced when attempting to operate multiple Xbox Kinects on a Windows device.

The idea for the project originated from a computer vision project hosted by Dr. Motai two years ago. The core of the research was using three Xbox Kinects with a Windows PC in which the team encountered hardware difficulties when trying to plug in each Kinect into the same PCI-E card. They deduced that each Xbox Kinect required its own PCI-E card for multiple reasons. First, each Kinect draws enough power to only allow for one device to operate on the card. Second, each Kinect required a unique serial ID, therefore, it was unable to share the same serial bus as another Kinect. Lastly, a commercial motherboard for regular PC’s usually allocate anywhere from two to four PCI-E slots, so a limitation presents itself in the number of Kinects that can physically be connected to the PC.

In simplified form, we are presented with the following complications when utilizing a Kinect on a PC:

• Installation of PCI card per Kinect device

• Motherboard may not have multiple PCI card slots

• Each Kinect device requires unique serial ID

• Each Kinect utilizes full power capacity of PCI card

The goals we seek to accomplish from our research given these problems are as follows:

• Develop software to interpret data from Raspberry Pi

• Adapt existing Raspberry Pi Kinect API for wireless data transmission

• Implement software to virtualize Kinect video

• Test functionality by operating our solution with an existing computer vision project that uses multiple Kinects

We created a modular, wireless device that alleviates those issues. By having each Kinect communicate with a Raspberry Pi over a serial connection, then having the Raspberry Pi communicate to a local Wi-Fi network wirelessly, which then sends the information to the host PC connected to the same local network.

The open source software used to make this possible is Robot Operating System and RVIS, a 3D visualizer for the ROS framework. RVIS allows the user to see what the robot is seeing, in our case, a point cloud image was developed. Along with that, we were able to get the functionality of the Kinect itself working, which consisted of the RGB image, depth image, and infrared image. All these images were streamed to the Raspberry Pi to the router Wi-Fi local network to the host PC on the same network, which created a video compiled of nearly 30 frames per second.

**Engineering Design Specifications**

1. Customer Requirements
   1. Human factors considerations
      1. 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.
2. Metrics for customer satisfaction
   1. 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 1.0 with Windows 10 will now be able to be completed using Wi-Fi. As an example, the most obvious use of this product is to use Wi-Fi Kinect 1.0 cameras to setup a point cloud for 3D mapping buildings/rooms.
3. Single use (consumable device) or reusable
   1. Reusable.
4. Sterile or non-sterile product
   1. Non-sterile.
5. Functional Requirements
   1. What should the product do?
      1. 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 virtualized 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 an API. The Pi will then transmit the data to the host computer via Wi-Fi.
   2. Performance metrics
      1. Video quality, connection range, data compression and decompression speed, GUI quality and software responsiveness.
   3. Engineering characteristics (units, ranges, limits with allowable tolerance ranges, etc.)
      1. Size, dimensions, weight (should be constant with target market application norms)
         1. The product will be a rectangular prism. The product will be lightweight with dimensions of approximately of 9x6x3 cm.
      2. Mechanical properties
         1. The Xbox Kinect and Raspberry Pi 3 is of electronic nature. The Pioneer 3 robot will be mobile via command sent through Matlab.
      3. Material properties
         1. The only materials used are going to be 3D printed rigid plastics, plexiglass and aluminum.
      4. Electrical, non-electrical requirements
         1. The Pioneer 3 robot will be manually controlled via keyboard commands
   4. Production Methods
      1. Appropriate manufacturing methods based on scale of components and design requirements
         1. The only component of the product is the Raspberry Pi 3 shell casing. The shell is going to be 3D printed plastics. The other components i.e. plexiglass to mount the Xbox Kinects, the aluminum support beams are purchased.
      2. Quality Requirements with Allowable Tolerance levels
         1. The Raspberry Pi 3 casing shell must be able to effectively and safely secure the Raspberry Pi 3 microcontroller, while still remaining elegant in aesthetics. The Xbox Kinect stand/mount must be at an appropriate height to be able to create a point cloud of the ground and the ceiling, while maintaining stability onto the Pioneer 3 robot.
      3. Packaging and Storage Requirements
         1. 1The product must be portable/mobile and be able to easy detached to meet height restrictions.
      4. Brief cost analysis
         1. Raspberry Pi 3 Microcontroller - $35.00
         2. Casing - $2.50
         3. Xbox Kinect 1.0 - $60.00
   5. Constraints
      1. Does this product have to function with specific other products? i.e. If computer program is part of design, what is spec on required hardware, etc.
         1. The product must deal with the constraints of two other products. The first constraint is that the Xbox Kinect streams massive amounts of data, roughly 300 Mbps. This constraint is solved by using a wireless router that is able to transmit and received two different channel frequency, 2.4 GHz and 5 GHz. The wireless router is also able to delivers up to 3150 Mbps Wi-Fi speeds. The second constraint is the integrated Wi-Fi adapter on the Raspberry Pi 3 board, which can only deliver up to 100 Mbps. This constraint is solved by installing a Ourlink 600mbps AC600 Dual Band USB Wi-Fi dongle
      2. Does solution have to utilize specific materials or manufacturing processes?
         1. No

**Prior Art**

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

**Societal Impact**

Demographics

The goal of any senior design is to find the solution for a given problem, usually one that affects how people engage with different technologies. Our senior design seeks to create a modular apparatus for one such solution; one which allows researchers, creators, and consumers alike to access new technologies easily. The Kinect currently has two revisions, both of which exist in a multitude of spaces. Initially offered as a consumer product, they can be found in many homes, connected to XBOX systems for gaming purposes. They have found a new home in many research labs across the country, such as our own. It provides the ability to do 3D imaging, video capture, and application development in an inexpensive package. The cost of the device encourages the creation of unique applications, as it allows labs and users of all kinds of funding levels entry into the Kinect ecosystem.

Our project seeks to add another option for researchers who want to work with multiple video sensors, and for the users of these such applications. When multiple Kinects are chained together, creators can create a 3D scan of a room instantly, or track users across a larger field of view. Current setups require being tethered to a desktop machine, and that too a desktop machine which can accommodate as many PCI slots as cameras needed. This setup gets costly and inefficient, something which can be stifling for developers and very foreign for a consumer. Our solution targets this inefficiency, as we have identified this as universal sore point. By creating a modular solution, we allow for any number of Kinects to be connected to a host system (which doesn’t necessarily need to be a Desktop anymore, as the information is routed over the internet). In removing this efficiency, we encourage innovation (by allowing more people to develop due to accessibility) , interaction (by providing an easy setup procedure for a consumer), and creation. This project aligns with growing access to the internet, computer, and connected devices by students and people around the world. Thinking globally and with accessibility in mind has allowed us to fulfill our goal of finding a solution for the people that use and interact with these technologies.

Health and safety implications of the design solution

According to the Health Protection Agency and the Wi-Fi Alliance, wireless networks do not cause a health hazard if it is maintained properly. Basic handling guidelines are to be followed to prevent and safety or health hazards, they are as follows: the location of wireless network installations must be registered and published by the institution using them, purchase of related equipment should follow manufacturer’s instructions for installation and operation, and do not touch or move antennas while a unit is transmitting or receiving.

There has been evidence of units such as cell phones that transmit at the same general frequency of radio waves and have been classified as a possible carcinogen, however, Wi-Fi signals transmit at a much lower power in comparison. I could not fully conclude that wireless networks are harming the environment, on the contrary, some may argue a negative effect from ‘always-on’ networks would be the amount of energy we consume. As the world shifts more towards a cloud and network-based functionality, there must be constant power provided as well as backup power in case of an outage, which equates to an increase of energy consumption. Could this argument be used against our proof of concept data transfer device? I would think not at this stage of development.

Economic issues

The Modular Wireless Xbox Kinect Data Transfer Device, shorted MWXKDTD, can be considered in either a global or local manner. Let us compare the design at a global, or higher level. Currently, there exists many hardware solutions for wireless data transfer for the Internet of Things that fall under working group 802.15 of the IEEE: WPAN/Bluetooth, Coexistence, High Rate WPAN, Low Rate WPAN, Mesh Networking, Body Area Networks, Visible Light Communication, Peer Aware Communications, Key Management Protocol, and Layer 2 Routing. Currently, at the higher level, our product, which falls under the 802.11 protocol, utilizes the wireless networking aspect (Wi-Fi). Let us examine why MWXKDTD can be examined as the most cost-effective proof of concept to be released commercially in comparison to the other 802.15 protocols.

Bluetooth, being the driving idea behind our project, deemed itself to be slightly cheaper due to the lack of a Wi-Fi router priced at $150, currently being used. However, the price-performance ratio unanimously eliminated a Bluetooth device as an option due to its deficiency in frame rate, therefore, we cannot justify Bluetooth as a proper comparison to our current approach. Now, let us compare the current model to the remaining 802.15 IEEE protocols. WiMax, being the next viable option for our product, did not fit our criteria in terms of the range of the bandwidth being too wide and the data rate slightly lower than required for our simple proof of concept model.

Furthermore, if our project were to progress to the next stages of development and find an adequate application, WiMax could be feasible option to achieve ranges greater than Wi-Fi capabilities. By deductive reasoning, we can conclude that if WiMax is the nearest alternative option but it not a possible one, thus, eradicating the possibility of the other 802.15 protocols from being an option, as they become exponentially more expensive.

Environmental Implications and Sustainability Issues

In our senior design, the two major devices in use that would have a consider amount of environmental implications and sustainability issues are the Xbox 360 Kinect and the Raspberry Pi 3. Let’s start with the Xbox 360 Kinect packaging. Being a Microsoft product, Microsoft follows the i-E2E program management which has the following principles,

• Eliminate environmentally unfavorable materials

• Minimize packaging weight and materials

• Increase use of recycled content

• Design for end-of-life recycling materials, separable components, and clear material markings

• Increase use of bio-based and other sustainable materials

• Reduce logistics and packaging manufacturing footprint

• Source raw materials responsibly

• Evaluate product platforms’ overall environmental impact through life cycle analysis

The Kinect packaging comes in a cardboard box with interior padded foam to prevent the Kinect from being damage in transport. We know cardboard can be recycle and is widely accepted at numerous recycling plants, but the foam commonly referred as Styrofoam or foam #6, is a thermoplastic which can be recyclable repeatedly, but not many community recycling programs accept it. There are numerous reasons. First, the foam isn’t a contributor to the waste stream. Foam #6 represents less than one percent of all products generated. Secondly, foam #6 takes more effort to collect a pound of than a pound of cardboard.

If foam #6 isn’t desirable by most recycling plants, why should we recycle it? Once the foam becomes compacted, it now has a value associated with it. There is a market demand for foam and the value or price often exceeds the price of cardboard and other recycling supplies. Secondly, recycling foam into a new product reduces the demand to harvest crude oil and all the energy associated with the process to make virgin resin. In the end, there is a reduction in greenhouse gases and the demand for foreign oil is decreased.

For the Xbox 360 Kinect device, Microsoft has taken the initiative to make sure the Kinect complies with the Waste Electrical & Electronic Equipment (WEEE). According to Microsoft,

Microsoft pays fees in each Member State to cover the WEEE management costs of its covered EEE.

• Microsoft provides information to reuse centers, treatment, and recycling facilities regarding Microsoft EEE as required by each Member State and the WEEE Directive.

• Microsoft products are designed to promote recycling, reuse, and proper waste management.

• Microsoft products are labelled or stamped with the WEEE marking as shown below in accordance with European Standard EN 50419.

All the components that make up the Xbox 360 Kinect can be used to make new products or generate energy by partnering up with recycling organizations, joining forces with collection schemes, and working with Microsoft stores and our OEM partners to facilitate the return and end-of-life management process.

As for the Raspberry Pi 3, the Raspberry Pi complies with the Waste Electrical and Electronic Equipment Directive (WEEE) by setting recycling and recovery targets. In addition, the Raspberry Pi 3 PCB assembly is RoHS (Restriction of Hazardous Substances) compliant, which forbids the use of lead or harmful metallic elements in the PCB manufacture. The RoHS helps reduce the amount of lead or harmful metallic elements exposure to people and the environment, especially in third world countries where most of technology trash ends up. Raspberry Pi 3 packaging comes in a cardboard box and according to recycle across America, recycling cardboard only takes 75% of the energy needed to create a new cardboard. In addition, recycling one ton of cardboard can save roughly 46 gallons of oil.

Public Policy Considerations

As technology moves forward, sometimes it is important for us as a society to re-evaluate our laws and customs to accommodate these new products and innovations. It is important to make predictions on how public policy should be shaped to prevent harmful use of the product.

In our senior design, we have developed a device that allows, using a Raspberry Pi 3, the Xbox 360 Kinect to transmit data wirelessly to a host computer. It is important to note the intended way that our group planned for the device to be used by the consumer. We developed this idea and product to help guide research efforts. This seemed a logical way to use the device, as the Xbox Kinect was dubbed a commercial failure, but has found great use in research applications. However, while the primary use for the product that we envision is benign and unlikely to affect public policy, there are always alternative ways to use any product. It is these other uses that might have a potentially negative impact on society and will require a re-evaluation of public policy.

The Xbox Kinect has innated facial recognition, which makes it much more useful than a standard camera for tracking and identification purposes. A store could implement this easily using our product. Simply set up an array of Xbox Kinects to provide a good layout of the store. Then, if trouble arises, the Kinect will be able to automatically find where a shopper is at any given time. This is the benevolent use of our product that will protect both shoppers and loss prevention by lowering the cases of false accusations of shoplifting. The main way we see our product needing public policy change is if a store or building was to keep a secret database of customer faces and their shopping habits. This already is a common thing for stores to do online, and, without policy change, will surely do with in person shopping. A change in public policy could prevent companies from doing data collection or at least require the company to inform the consumer that they collect the data.

**Prototype Development**

In our early stage, our first iteration was to mount each of the three Xbox Kinects to their own tripod and attach each of the three Raspberry Pi’s to one of the three legs of the tripod. The USB portable charger will also follow the same mounting setup as the Raspberry Pi’s. The Xbox Kinects were mounted to the tripod via a clip-on mount that attaches to the underside of the Xbox Kinect. The tripod design allowed our team to manually pick a certain direction and was easy to transport/portability.



Figure 1: First prototype iteration

Our second iteration was to keep all three Xbox Kinects stationary, while velcroing them on top of a trapezoid plexiglass. The plexiglass was then mounted on two aluminum support beams via screws and bolts and the support beams are attached to a bottom metal plate using L brackets via screws and bolts. The metal plate is velcro on to the Pioneer 3 robot. Each of the three Raspberry Pi’s are inside a 3D printed case, which are also velcro to the plexiglass. We had to opt out of using the USB portable charger and used a heavy duty portable power supply, due to the fact the Raspberry Pi’s weren’t getting enough power. The heavy-duty power supply was placed in between the two aluminum support beams. We also attached a Ourlink 600mbps AC600 Dual Band USB Wi-Fi dongle to each of the three Raspberry Pi’s. The monitor was velcro onto the heavy duty portable power supply in the front with the microcomputer placed in the rear.



Figure 2: Second prototype iteration

**Performance Testing**

The first performance test we performed was a network speed/transfer rate benchmark on each of the three Raspberry Pi’s to test the Ourlink 600mbps AC600 Dual Band USB Wi-Fi dongle when streaming the point clouds. We installed a speed test application to monitor the upload and download bandwidth from each of the Raspberry Pi’s to the master computer. This benchmark resulted in discovering that all three Raspberry Pi’s network channel will cause some collision/interference with one another. The average network transfer rate for all three Raspberry Pi’s was 8-15 Mbps for upload and download.

The second performance test we completed was a second network benchmark test with the same setup, but this time it involved in changing the channel frequency for one of the Raspberry Pi’s to 2.4 GHz instead of 5 GHz and have the other two Raspberry Pi’s on the 5 GHz frequency. This resulted in less collision/interference between all three Raspberry Pi’s and an increase in average network transfer rate from 8-15 Mbps to 10-18 Mbps. The third performance test we performed was to check the response time it took from when a command is sent from the onboard minicomputer and received to the Pioneer 3 robot. This performance test resulted in a 4-millisecond delay. The last performance test we performed was a distance benchmark. This distance benchmark was to check the maximum range our wireless router could communicate with the Pioneer 3 robot. This resulted into a maximum range of 30ft from the wireless router.

**Design Solution**

Our final product was the culmination of two semesters of testing, prototyping, adapting, and debugging. At the beginning of the second semester, we decided to make some radical changes to our design due to unforeseen limitations in our prototype. In the prototype, we sought to use a combination of Bluetooth and the Kinect for Windows SDK to realize our product. Unfortunately, Bluetooth had a transfer rate that was below our tolerance level, and the Windows SDK was locked down behind proprietary code. These original specifications restricted us from moving forward, until we were able to come up with not only suitable, but better alternatives. We swapped Bluetooth for TCP/IP based communications, and we swapped the Kinect for Windows SDK with the ROS Kinetic Kinect SDK.

Our design has always revolved the idea of modularity. In our original iteration, we associated this modularity with portability. This portability meant creating a solution which minimized wires, had as few pass through devices as possible, and limited hardware. The idea behind this was as the need for using multiple 3D Sensors (specifically Kinects) arise, the setup process becomes much more tethered to a single location. This is due to the sensors requiring additional hardware (PCI/e cards) to be placed inside the host system so that the full bandwidth of data can be passed through. This is only feasible with a desktop computer, and is a costly solution. We wanted to be able to run projects that required multiple sensors on a laptop, or other desktops, without needing to move the hardware out from host machine to host machine.

Our very first design was a hardware based rate limiter which would take in the input of three or more kinects into a device. We would then code some software for the device that would filter data out from in the input stream, and send the combined feed to the host machine. Some noise would have been added to the stream, and used to initialize three virtual kinects on the host. This design was deemed unsuitable by Dr. Elks. It required an advanced knowledge of hardware design, and in hindsight, would not have worked with the final tools (ROS, Linux) that we ended up using.

Our second design carried on the general idea of the first design, but approached it in a radically different way. We wanted to keep the idea of the host machine doing all the processing, but realized that the data needed to be transferred from the kinects in a different way. In keeping with the idea of modularity, we decided that the information should be sent wirelessly, instead of through a wired hub. In lieu of a direct connection to the host computer, the Kinect would have to be connected to some microcontroller which could process the data. With wireless, there exist two main options: Bluetooth and TCP/IP. We went with Bluetooth, as Bluetooth doesn’t require any intermediary devices (I.E. Wireless Router), and the microcontroller we selected to do the processing (the Raspberry Pi) has Bluetooth built in. Once this specification was decided, we had to decide how we would receive the information. As this project was supposed to be an extension of a previous Kinect project (which was completed and implemented using the Kinect for Windows SDK), we decided to stick with Windows. As with the previous prototype, this meant we needed to reverse the Kinect for Windows SDK to learn how to create a virtual Kinect.

As the semester progressed, we ran into issues with the Bluetooth protocol, specifically the interaction between the Raspberry Pi and a Windows host machine. The Windows host machine viewed the Raspberry Pi as an audio device, and was only able to control the volume. Though we were able to get a single file transfer initiated, we were not able to replicate the behavior continuously. We made the decision to change the protocol from Bluetooth to TCP/IP as we were able to easily initiate Wi-Fi transfer between the Raspberry Pi’s and the host computer. Additionally, Wi-Fi afforded us the opportunity to increase the bandwidth from a maximum of 10Mb/s to 434Mb/s. This negated the need to rate limit any of the streams from the Raspberry Pi’s. Switching to Wi-Fi was the catalyst we needed to change other features of the project.

Upon switching to Wi-Fi, we began to notice the problems with the Kinect for Windows SDK. Our initial design relied on being able to send image data in real time, process it using a custom software, and then pass that to the virtual Kinect instance. This idea had to be scrapped as we were unable to create the virtualized Kinect. Virtualizing the kinect would have required the reverse engineering of the Kinect SDK. As with the original hardware solution, this implementation required an advanced knowledge of programming that was out of our expertise. Additionally, the SDK is not publically available to edit, so we were unable to access the code. This mandated the change of SDK from the Windows default to the ROS (Robot Operating System) Kinetic SDK. This SDK is intended for Linux based OS’s, so we had to change the operating system we used from Windows to Linux (E.G. Ubuntu). Alongside this change came a change with the software used to display the images. We switched to a combination of RTABMAP and RVIZ (both of which can be found in ROS). This combination of software lets us view the depth data, infrared data, and image data in one consistent view.

Finally, after some input from our professor, we were able to settle on our final design. The final design solution consists of 3 Raspberry Pi’s, each running Ubuntu Mate 16.04 and paired with a 802.11 WiFI AC with MU-MIMO adapter. The Raspberry Pi’s connected to a AC3150 Wireless Router, which was connected via Gigabit ethernet to a Host Computer. The host computer was running Ubuntu 16.04. Both the Pi’s and the host computer had ROS Kinetic installed. A Microsoft Kinect V1 was connected to each of the Raspberry Pi’s. ROS was used to process the data on the Raspberry Pi side, and then was used to interpret the data on the host side. The information was sent over a 802.11ac WiFi connection (through the wireless router). The Pi’s and the Kinect’s were mounted onto a Pioneer 3 mobile robot. On the robot, a monitor was mounted and connected to our host machine, so that the 3d image could be viewed as the robot was moving. Additionally, a mini computer was located on the robot as its controller. The robot was a late-stage addition, and came about due to a suggestion from our professor. He thought by mounting the project on the robot, we would be able to show a real application of our solution (i.e. 3D scanning). This flow is described more in detail in the two images below:

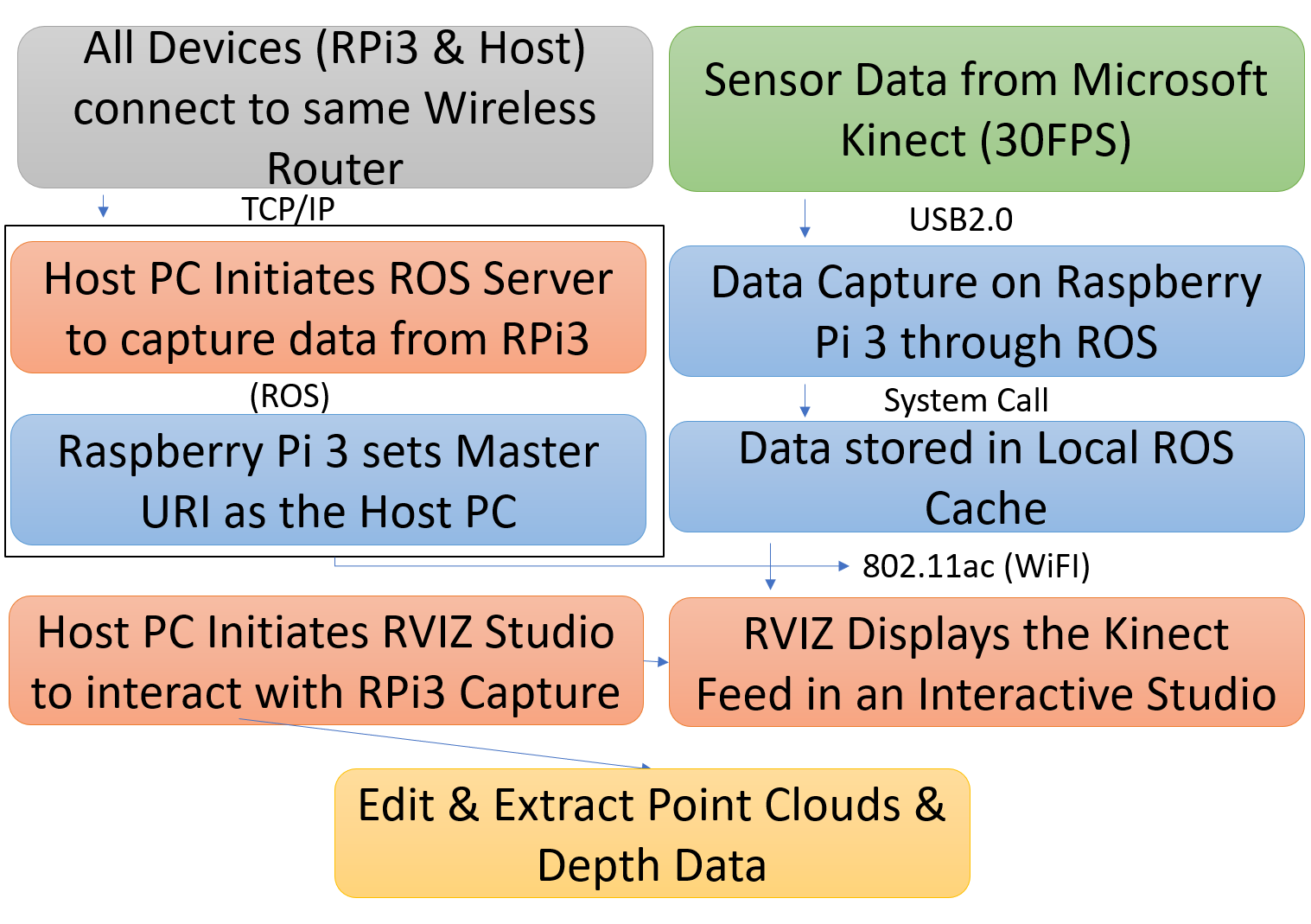
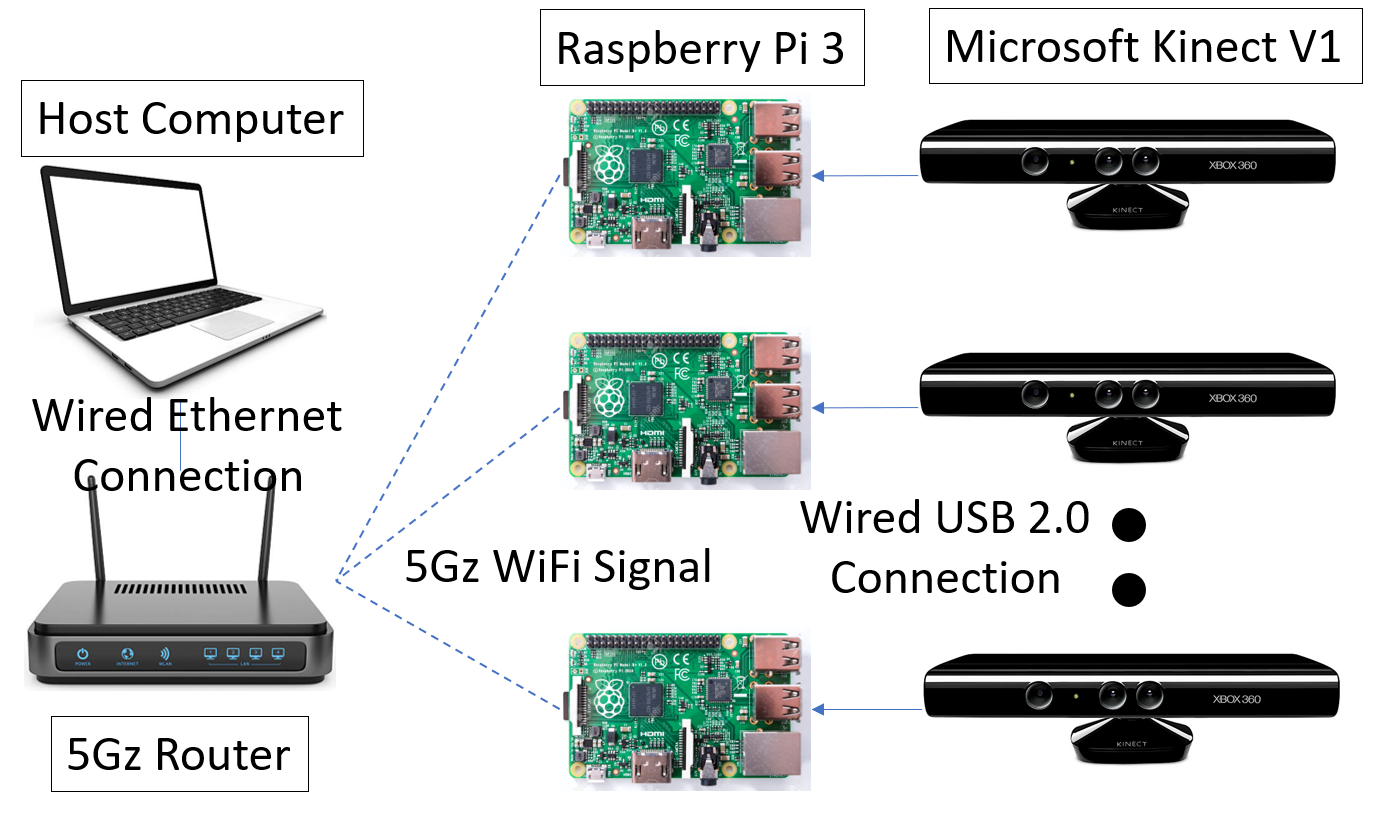


Figure 3: Schematic diagram of system Figure 4: System Design Protocol

Our final solution was truly a result of the mistakes we learned from our earlier designs. Without first trying the far-fetched and improbable ideas that we originally considered, we may not have been able to appropriately search for alternatives. As indicated, we built up from the ideas of each prototype, and changed key resources and tools as it became clear we could not use them. Our final product delivered on all the promises we originally made. Additionally, due to the way it was designed, the pain points upon which the project could be improved are clear. These improvements are straightforward, and could be easily continued on by another group.

**Financial Regulatory Aspects**

Regulatory

Robots have been in use for over 6 decades now, and have helped many industries move

forward and automate inefficiencies and safety hazards out of the manufacturing chain. As robots

evolve, their use in industry does as well. Increasingly, companies are using mobile robots in

their manufacturing operations; to move and deliver objects. While there is regulation that exists

for general machine use in these environments, regulation on Mobile Robots is low. These

devices of course have to pass FCC and FTC regulations to be deemed safe to use, but there is no

government-level regulation on the unique uses of mobile robots. At an engineering level, there

are some regulations that have been published by widely accepted bodies that govern standards

and regulations for new protocols and technologies. One such regulation comes from the

American National Standards Institute (ANSI) and the Industrial Truck Standards Development

Foundation (ITSDF). This regulation (B56.5-2012) is mostly in regards to driverless

vehicles/cars, but extends to any automated or mobile robotic device. Another, comes from

ANSI and the Robotic Industries Association (RIA). This Regulation (R15.06-2012) concerns

industrial robots and their relevant safety requirements. These safety standards cover how to

safely integrate these mobile robots in an industrial environment, but due to their age miss out on

the risks that come with better path correction (the training required to get this to work, and the

mistakes that can happen) and accessory development (adding functionality to the mobile robots

such as robotic arms, and how this affects other workers).

An annual conference, the National Robot Safety Conference (NRSC) lets these bodies

work with each other to propose and draft new regulations. Between May of last year (2017) and

September of this year (2018), the RIA in conjunction with ANSI and the International Standards

Organization Technical Committee 299 (ISO/TC 299) is drafting new regulations which will be

voted on in 2019. Some of these regulations include R15.08 (ANSI &amp; RIA) is specifically

targeted for Mobile Robots in an industrial context. With the ISO, the RIA is developing and

refining the WG1 – WG6, JWG5, JWG9 and SG1 standards and definitions. These standards

seek to regulate and standardize the terminology and usage of medical robots, industrial robots,

industrial mobile robots, service robots, and more.

These standards are needed to ensure that we are able to build environments and mobile

robots that can co-exist, and where the safety and integrity of both human works and the robots

can be ensured. As these technologies gain steam, we will see more problems arise should these

standards not be finalized and implemented. Our robot is currently an academic endeavor, and is

in a prototype phase. The regulations that would be implemented would likely not apply to the

testing scenarios that we have devised, but may be implemented if the robot is further refined

and used for actual 3D scanning projects. In that scenario the robot would be operating in an

environment where it will interact with human, non-human, and inanimate obstacles. To ensure

the robot doesn’t break, and that the safety of the environment is upheld, we would then have to

adhere to these regulations. Some of these regulations would include path safety and obstacle

detection and stopping.

Lastly, another one of the main regulated components of our senior design project is the Raspberry Pi 3. Since the Raspberry Pi 3 has already passed European Conformity (CE) and the Federal Communications Commission (FCC) regulations, then it will have no effect on our project outcome, but if it wasn’t regulated then it would have initially had to pass intentional radiators and unintentional radiators test from the FCC and the electromagnetic compatibility (EMC) from the CE. EMC testing is a necessary prerequisite to obtaining various consumer-protection certifications including the CE marking, which test if the Raspberry Pi 3 generates unacceptable levels of electromagnetic noise. The intentional radiation test means the FCC are purposely putting out RF signals i.e. the onboard Wi-Fi and the Bluetooth on the Raspberry Pi. The Wi-Fi and Bluetooth must be tested and filed with the FCC before the Raspberry Pi 3 can even start selling or be advertised. This regulation is under CFR 47 Part 15 Section 247. For unintentional radiation test, this could be switching noise from a power, accidental antennas from poor ground pours, or long clock traces. Testing reports also would need to be sent and filed to the FCC. This regulation is under CFR 47 Part 15 Section 109.

Financial

The Modular Wireless Xbox Kinect Data Transfer Device seeks alleviating the cost of implementing multiple Kinects on a PC. The previous computer vision project attempted to develop software for 3 Xbox Kinects running on a PC. To do so, the team needed to purchase a PCI card for each Kinect that connected to the PC. Each Kinect required a unique serial device ID and used the entire power output from the PCI card it was connected to. Also, most motherboards do not have multiple PCI card slots, which meant the user had to purchase a motherboard advanced enough to support multiple PCI card slots. The cost of implementing the previous project accumulated mostly from hardware upgrade requirements rather than the software to actually implement their solution. This triggered the idea to develop our modular solution, eliminating the need for various hardware changes. An estimated cost for hardware changes to implement the previous project can be found in the table figure 1, below.

|  |  |
| --- | --- |
| Hardware | Cost |
| Gigabyte/Asus Motherboard with 3+ PCI | ~$65 |
| PCI Card | ~$25 |
| 500W Power Supply | ~$80 |

Figure 5: Estimated Hardware Cost for Alternative Project

In figure 1 we find most 4-PCI motherboards to be about $65, each PCI card to be $25, and a power supply, we assume 500W, sufficient enough to provide enough power to each PCI card along with other components, $80. The base total cost would run the team $220, assuming the cheapest parts are purchased, not including other miscellaneous hardware upgrades to be compatible with the motherboard such as processor, ram, graphics card. The previous project also had to purchase the Kinects, each Kinect costing $100. A rough final cost estimate could easily run the team $800-$1000, taking into account all possible components.

The cost of our project was the cost of the each Raspberry Pi, a router, and small miscellaneous components listed in table 2. We do not include the cost of the Kinect because they were already available from the previous project.

|  |  |
| --- | --- |
| Hardware | Cost |
| Raspberry Pi 3 with Heatsink x3 | $40 |
| Router | $150 |

Figure 6: Hardware Cost of MWXKDTD

We can roughly estimate the total of our project to be $300, including any small, miscellaneous components we had to purchase such as cables. We can confidently say we have cut the cost of implementing the hardware of the computer vision project by at least half by eliminating the need for PCI hardware upgrades and implementing a modular, wireless data transfer solution.

We have several recommendations for the financial and regulatory aspects of the projects as it stands. The plan is for the robot to become more and more automated and to eventually do an automated task, such as automatically navigate a room and map its layout and inhabitants. The robot will need to eventually be able to work in an environment where it will interact with human, non-human, and inanimate obstacles, the addition of standards for the robots functionality become a focus in the future as the product becomes more finished. Notably, the regulations and functionally involving path safety, obstacle detection, and stopping. For improvements, the biggest thing moving forward is that faster speeds be implemented so that data updates more rapidly on the host PC. This will be easily accomplished, as advances in WiFi speeds are happening constantly.

**Discussion, Conclusions, Recommendation**

The project was a huge success, as we met all of our goals set forth in our Project Description. Our original design of having a wireless Kinect transfer device was conceptualized and brought into and completely through the design and prototyping phase. Using our projected concept of using an array of three Raspberry Pi 3.0s and three Xbox Kinect cameras, we were able to make a system of wireless cameras that would send data through 2.4 and 5 GHz band WiFi to a host computer for data processing.

The prototype was thoroughly tested with expected results, which can be found in the Performance Testing Section. Our results cover most of the goals we created when the project was started, and the speeds we were able to achieve were greater than expected.

The goals that we set out to achieve were as follows:

• Develop software to interpret data from Raspberry Pi

• Adapt existing Raspberry Pi Kinect API for wireless data transmission

• Implement software to virtualize Kinect video

• Test functionality by operating our solution with an existing computer vision project that uses multiple Kinects

All of these goals were met except for goal number 3. We developed software that would interpret the Kinect data, we adapted an existing API for the Kinect for wireless data transmission, and we were able to recreate the facial recognition project of the year before, as well as some other functionality that was displayed through our many demos. We did not, however, virtualize the Xbox Kinect. While the task sounded simple when we set the goal for ourselves at the project interim, it was a much more complex task than we initially believed. Instead of directly virtualizing the Xbox Kinect, we decided to further modify an existing Xbox Kinect library called libfreenect. This library is made for Linux OS, and does not interface with windows.

Some recommendations we have for next year are as follows. The most obvious improvement would be to virtualize the Kinect in a much more direct way. This would allow the system to be run with a windows host computer. The next upgrade that could be done would be to increase the data transfer speed with multiple user multiple input multiple output (Multi-user MIMO) upstream. This would help eliminate collision between the data transfer coming from each camera. The technology was not available during our iteration of the project, so we could not implement it. The last recommendation we have is to use a stronger microprocessor in order to support the Kinect V2, which is a superior camera. The raspberry pi 3 cannot support the Kinect V2, as it only has 2.0 USB support.

**References**

Association, R. (2017). Robot Safety Standards for Industrial Mobile Robots. [online] Robotics Online. Available at: https://www.robotics.org/blog-article.cfm/Robot-Safety-Standards-for-Industrial-Mobile-Robots/63 [Accessed 13 Apr. 2018].

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

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

Europa.eu. (2018). EUROPA - European Union website, the official EU website. [online] Available at: https://europa.eu/ [Accessed 13 Apr. 2018].

Federal Communications Commission. (2018). Federal Communications Commission. [online] Available at: https://www.fcc.gov/ [Accessed 13 Apr. 2018].

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.

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/](http://www.mariolukas.de/2015/04/proof-of-concept-3d-scanner-with-kinect-and-raspberry-pi/).

Microsoft. “Kinect API.” Kinect API Overview, Microsoft, 2017, msdn.microsoft.com/en-us/library/dn782033.aspx#ID4E3.

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/.