Project Documentation

QwikBox

High-end Remote System Optimized for Uploading High Definition Video

Department of Electrical Engineering and Computer Science
University of Central Florida
Senior Design 1
Dr. Lei Wei

Group 25

Jonathan Kerbelis	Computer Engineer	jonathankerbelis@gmail.com
Eric Downey	Computer Engineer	edowney29@gmail.com
Harold Frech	Electrical Engineer	roldfrech@gmail.com
Todd DeNoyer	Project Sponsor	todd@qwikcut.com

QwikBox Table of Contents

Total pages: 95

List of Tables

List of Diagrams

List of Pictures

Executive Summary

QwikCut is a company that provides video and analytical solutions for youth sport teams. The QwikCut (1) team hires camera men to go out to these youth sporting events to capture the games and then the company edits the videos, creates copies, and uploads these games to a hosting platform called Hudl (2) for internet viewing. The coaches and players want to have access to the video as soon as possible to enable these people to quickly critique themselves. The sooner the video is delivered the more time there is to prepare for future opponents and self-critique. Thus, it is imperative to get the video back to the customers as fast as possible. This is where the QwikBox excels. Our project is to build a system that can quickly encode, store, and upload video. It will take a full game's worth of video clips and upload it onto a cloud based server in real time. This system will interact with a server to transfer the video onto a database so that the customer can directly access videos, from either a smartphone or computer, as soon as the uploading is complete.

To implement this process the project will include a combination of hardware and software components. The hardware will include a processor, memory, wireless modules on a PCB, batteries, and HDMI to USB interface, and a video camera. The video camera will live stream the video to the QwikBox via HDMI, and the camera operator will split the video into clips through a user-friendly mobile application. The video will then be clipped and encoded into a desired codec. Once encoded, the video files will be stored on a micro SD card and queued for upload to the server provided by the sponsor. Since upload speeds will vary depending on signal strength, a queue is necessary to ensure that each video file is uploaded successfully before the next file begins. The software will be an application that allows the user to access their respective video and be able to do so from a web based server. The desired video codec will be in H.265, a video compression standard, and the wireless systems will be designed for the 4G and 5G LTE networks. In addition to wireless cell networks, the system will have Wi-Fi and Ethernet capabilities. These standards will ensure that the system will be relevant for years to come.

2. Project Description

2.1 Project Goals

Our main goal for this project is to create a QwikBox that satisfies all requirements presented to us by our sponsor QwikCut. The project involves a cubed shape computer system about the size of a video game console. The box itself will be able to withstand mild weather conditions and be completely portable with the utilization of wireless data transfer and portable power source. QwikBox will involve the process of compressing, encoding, storing, and uploading high definition video data in the High Efficiency Video Coding codec; usually referred to as HEVC or H.265 codec (3).

For the end user, our goal is to enable the user to have access to their videos through an online web based server which will allow them to view and edit video files. An android application will be developed to give our user access to these video files. Above all, the system needs to have quick uploading speeds so that users can have access to their video soon after it is recorded. The wireless communication feature will provide QwikBox with this ability.

2.2 Project Objectives

The objectives of this project are to create a functional prototype of the QwikBox. We will do this by combining certain components to make a complete computer system capable of recording, compressing, storing, and uploading high definition video. In order to do this a Raspberry Pi microcontroller will need to be fitted with a microSD storage and be connected to a camera. This will allow QwikBox to capture high definition video from a camera and store it to a microSD where it can be used later for compression and uploading. Along with that, HEVC encoding software will be needed and must run on the Raspberry Pi (4). We will program the system to pull the captured video from the storage device and convert to a compressed format using the software on the microcontroller. From there, the now compressed high definition video will be uploaded to the web based server using the antenna connected to the Raspberry Pi. A printed circuit board will be created in order to allow the connections to be made between the microcontroller and antenna. The process from converting to storing and uploading needs to be as fast as possible so we will optimize every step of the process as the video is uploaded.

QwikBox will also be accompanied by a companion app that can be used to control the setting of QwikBox and view uploaded videos. This application will have a start and stop button on the featuring on the app that will allow the QwikBox to know when video should be recorded. The app will also feature a settings option that will allow user to name and edit videos. This application will need to communicate with the Raspberry Pi via Bluetooth commands meaning programming will need to be done to

the Raspberry Pi in order to understand these commands. The application will be needed to run QwikBox. This means that both the application and QwikBox will need to be fully functional by the end of this project. We plan to accomplish all these objectives on time and fully completed.

2.3 Requirements and Specifications

There are certain requirements and specifications that we must meet in order to complete the task given to us by our sponsor. The multiple specifications issued to us include a low cost production per unit on a mass production scale, a weather proof unit that is able to withstand mild climates outside, the fastest possible upload times using 4G LTE wireless communication, and the user friendly QwikBox that anyone can use. To make the the QwikBox with the lost production value a Raspberry Pi has been chosen as the main component within the system. This has cut the cost of using a small computer by replacing it with a microcontroller. Also, by choosing the Raspberry Pi, the separate storage device can be entirely removed and replaced by a micro SD card which plugs directly into the Raspberry Pi. To create a weather resistant box, we will 3D print a plastic container which will seat the Raspberry Pi, power source, and antenna inside and be sealed used rubber liners to prevent water damage. To ensure the greatest possible upload times, the wireless connection must be established and uninterrupted. This means that a powerful antenna will need to be used. This goes with that fact that video compression and storage must be equally as fast. To make QwikBox user friendly, only a HDMI input, micro B USB input, and power button will be seen by the end user. This design concept will be implemented within the QwikBox container and was chosen to prevent confusion and to increase the ease of use. These specifications must be met to ensure the highest quality product for our sponsor.

Along with that, a main requirement for this project will be to make a companion application that is able to control QwikBox. Also a main requirement for this project is to develop a process for pulling saved high definition video from the storage device, converting that video into a compressed file size using the HEVC codec to maintain the quality of the video, and storing that compressed video where the antenna is able to upload it to a web server. This whole process needs to happen at the push of a button and within a reasonable time frame. It’s also important to note that during the entire file storage and compression process the high definition quality of the video cannot be lost. To ensure this doesn’t happen in the newest H.265 compression standard will be used. From here, the last thing needed for a fully functional QwikBox is the to register the antenna onto Verizon’s 4G LTE wireless communication servers and program a method for storing video on the web server. These requirements must be met in in order for the QwikBox to be a completed prototype.

Specifications and Requirements Breakdown

Specifications	Requirements
Completed task	Around 5 minutes
Low cost production	Under $300
Weather resistance	Water, dust, and heat
Upload speeds	Around 5 Mbps

Table 2 – 1

2.4 Quality Analysis

In order to ensure this product will be able to function throughout all of the design process we will tackle this project in stages. Actively and accurately identifying problems as they arise in the individual stage of the project allows for no ambiguous troubleshooting problems. A project can get backed up very quickly on one aspect of the project when trying to complete the whole thing at once. Preventing a poorly produced project can be easily accomplished at each stage. We can start by making sure can record a clip. Additionally being able to manage the clips storage with a raspberry pi 3 is a following task. Lastly important part is through a service provider we can efficiently place the clip to a server for the users viewability. The table below describes some of the many possible outcomes which could arise will creating QwikBox. These issues can have a positive or negative impact on the design of the product. Quality will be ensured by essential testing every aspect of QwikBox,

House of Quality

		Storage	Processor Speed	Battery Life	Durability	Size
		+	+	+	+	-
Upload Time	+	↑	↑ ↑	↑	--	--
Video Quality	+	↑	↑	↑	--	--
Cost	-	↓	↓	↓	↓	↓ ↓
Power Usage	-	↑	↓	↓	↓	↓

Table 2 – 2

3. Research

3.1 Existing Similar Projects and Products

This project is unique in design components and functionality but there is a very similar existing system designed by Hudl. This project is meant to compete with and even better the Hudl product in many aspects. Hudl Sideline [5] is a product that allows users to use their existing cameras and camcorders to stream live video to a local mobile iOS device so that it can be replayed on the sideline by coaches and players within a few seconds. It uses a portable router that acts as a bridge between the different devices that are recording, sending, and receiving video. There are benefits as well as drawbacks with this design approach, and they are outlined in Table 3 -1 below.

Pros and Cons List

Pros	Cons
Ability to establish a seamless connection between multiple devices that allows for video transfer between them.	Pricing is huge drawback of Hudl Sideline. The service costs each program $900 per YEAR. There is also an premium service offered for $1500 a year. This price can be extreme burden.
Video can be uploaded directly to the Hudl platform once a stable Wi-Fi connection is established.	Hudl Sideline relies on a stable wireless internet connection in order to upload video to the Hudl platform. A cellular network is not an option, so this leads to a lot limitations with the upload capabilities.
Can be viewed by coaches and players on the sideline of a sporting event in as little as eight seconds	When purchasing the Hudl Sideline system, it comes with multiple different devices that customers have to configure and set up themselves. Including these multiple parts into one single device would make it easier on the different consumers that use the product.
Can use either a video camera or an iOS device to record the video.	Hudl sideline requires at least two people in order to properly operate its functionality. One person is using the video camera to focus on the game footage, while another person is using an iOS device to clip the video into individual segments or plays.

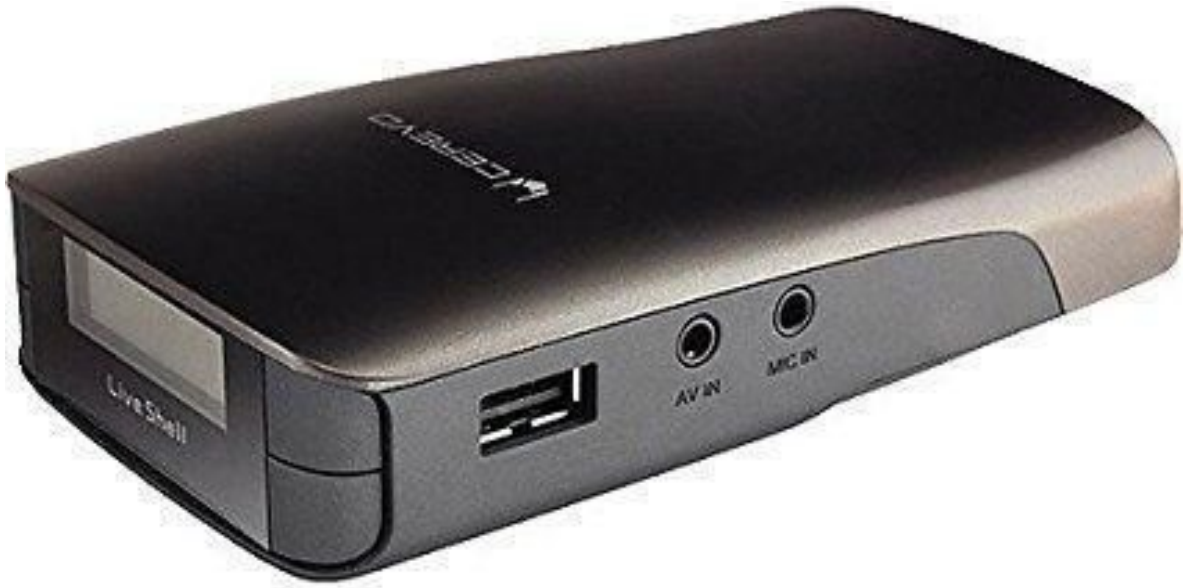
Can be used with the mobile iOS platform.	Mobile support is limited to only iOS device. Android support does not exist at this time.

Table 3 – 1

Our QwikBox project primarily focuses on improving on the drawbacks of Hudl Sideline and making it accessible to many more sports programs around the state of Florida, or even the entire nation. It will cost much less per year, and it will still have most of the same features that Hudl Sideline has. The video will be uploaded to a server to be used by anyone who has access, whether that be a coach on the sideline or a parent 200 miles away. The QwikBox will also have the ability to upload video over a cellular network when a wireless or Ethernet connection is not available, which is often the case at sporting venues. Various components will be combined into one simplified device with very minimal setup, and it will only require one person to operate the system when only one video camera is used. We will also have support for both iOS and Android devices so that practically anyone with a smartphone can operate the end product.

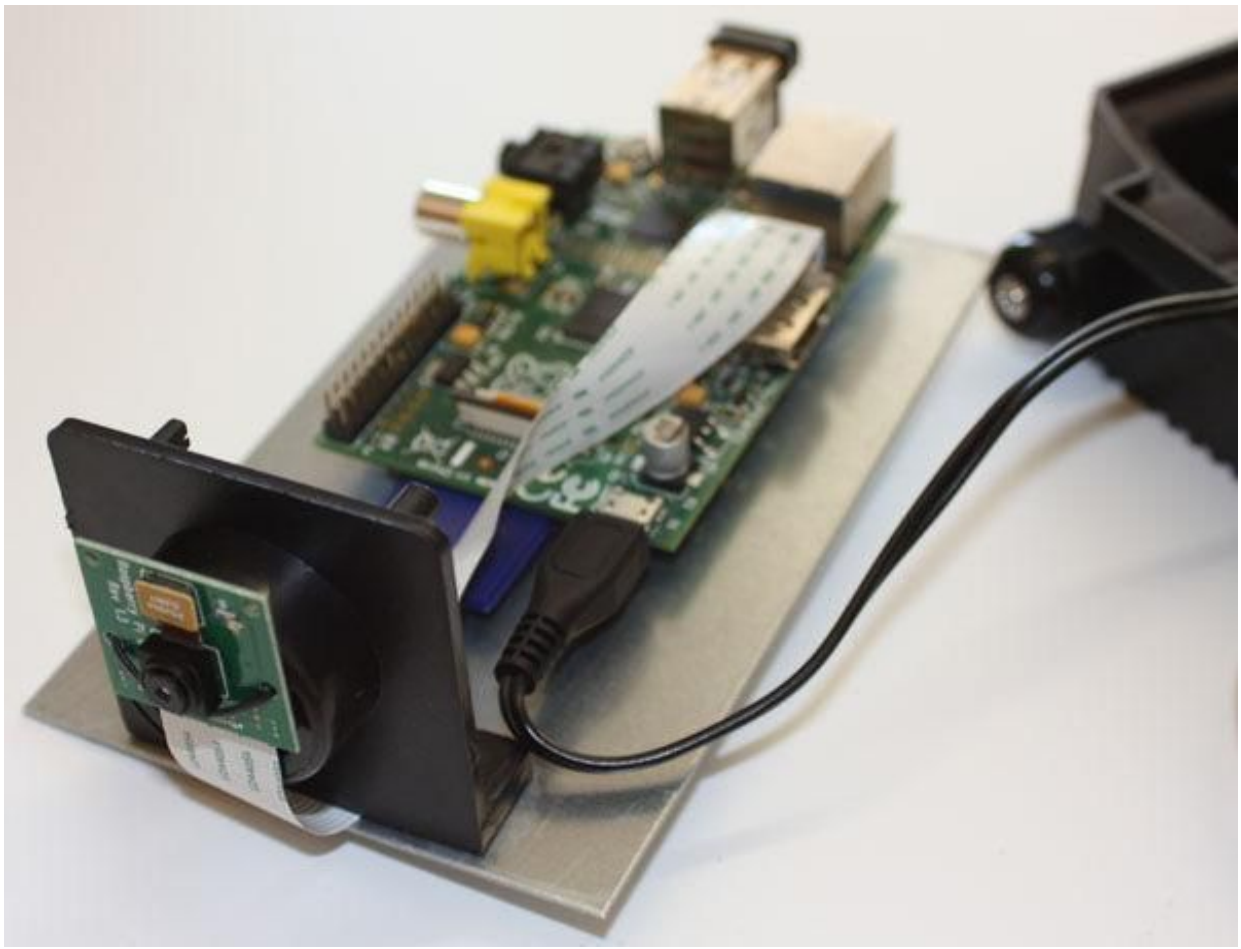
There are also devices like the Cerevo LiveShell that takes in a live video input and uploads it to various online storage options. This device can take in a HDMI input from a camera, process it, and then upload it. Our QwikBox will be able to do all of these things as well, but it will be more tailored to sporting events. It will also be uploaded to a custom server and it convert and compress the video before it does so. The ability to control the streaming device from a mobile application is also a feature not offered by the Cerevo LiveShell (Picture 3 – 1).

Cerevo LiveShell



Picture 3 - 1

There are also a few examples of existing projects that feature the conversion of high definition video to a smaller compressed size via a Raspberry Pi. These projects were designed to convert video into a smaller format using a portable solution and this is where the Raspberry Pi microcontroller becomes extremely appealing (Picture 3 – 2) . In these examples the high definition video was converted to a smaller format using VideoLAN's VLC media player running on a Raspberry Pi. We plan to implement this exact process in order to convert any high definition video to the HEVC codec format. VLC supports this H.265 video compression standard using a third party extension. Considering the Raspberry Pi will be running at a high potential, it requires a small amount of power to operate and is able to run very efficiently. We intend to build upon these examples and include a Raspberry Pi as the main component of QwikBox system. On top of that, our design will also have to upload the compressed video to an online server via 4G LTE wireless communication. This will be done using a PCB and wireless communication antenna connected the Raspberry Pi. Uploading video will prove to be difficult but lucky NimBeLink has designed an interface to connect a Skywire antenna to the Raspberry Pi. By using this interface our project will become easier to build and maintain. This also means that other similar project have need the same solution as we do and that documentation for this interface will be available for our use. These aspects from other projects have helped us determine which products to use in the final design of QwikBox.



Picture 3 – 2

There are also other types of hardware called stream encoders which provide video compression for any high definition video format. The unit transfers the video through an HDMI input connection to an HDMI output source. The unit takes the video and compresses it to the HEVC format. Not much is known about the hardware inside the stream encoder due to intelligence property rights. With that being said, the unit was in our initial design but was taken out after extensive research. This was due to the costly price of each unit and the large size of which the units are made. Along with that, it would have been hard to adapt a small computer with a stream encoder and the overall system would draw too much power to be a portable. The design also provides no means of storage and would have to be fitted with some kind of temporary storage until videos were uploaded. All in all, using this design would a have proven to be a disaster from a marketing standpoint and be an engineering nightmare. Though these processes won't be built upon, it was good for us to research the topic and obtain a better understanding as to what we want QwikBox to be able to do and how we want it

to do those task. This project will expand on stream encoder idea and create a new system that is portable and would hopefully be able to replace a stream encoder unit as a whole.

3.2 Relevant Technologies

In order to produce a functional QwikBox, there are some relevant technologies that must be taken into account that will enable every aspect of QwikBox to work together to produce the end result. These technologies are absolutely necessary to the success of QwikBox and will be explained in greater detail as to why this is so. Below is a list of technology used throughout the QwikBox system. Some of these technologies are new to the market and all the technologies fit the current standards. This will ensure the QwikBox is equipped with the best possible solution for the task at hand. A key component for QwikBox is its ability to compress video and transfer it using wireless communication data transfer to online web servers. For this to be successful, QwikBox must utilize the best technology available.

List of technology:

1. the Raspberry Pi interface
2. a wireless antenna for data transfer
3. a high capacity external battery pack
4. the High Efficiency Video Coding codec (or H.265)
5. an HDMI to USB video and audio interface conversion tool
6. the Linux operating system
7. software for converting high definition video to H.265
8. the Amazon Web Services storage solution

1. Firstly, The Raspberry Pi has been used in a plethora of projects to create fully functional systems from arcade video game consoles to robotic autonomous drones. There is no technological project that the Raspberry Pi cannot be adapted too and this is why it is hugely popular among the homebrew hobbyist who creates these different things. This versatility in form and function are what make the Raspberry Pi the perfect candidate for the QwikBox system. The Raspberry Pi comes equipped with great features such as expandable storage, multiple USB 3.0 hubs, and an Ethernet and HDMI port. The Raspberry Pi provides us with many options for choosing how we want QwikBox to operate.

2. Another important technology to consider is the wireless communication device. This antenna will feature 4G and 5G LTE data transfer speeds and will have to be usable on Verizon's LTE data service. The antenna itself will be a part purchased from a retailer who makes plug-n-play antennas for Raspberry Pi interface. A couple notable sources of these antennas are from NimBeLink and Adafruit Technologies. These companies have produced an economical means

of obtaining and adding antennas with wireless communication abilities to small project such as the Raspberry Pi. These antennas will prove to be the best option for QwikBox.

3. A high capacity battery pack is not really an extremely new technology but only recently have they been pushed as a cheap market consumer product. These battery packs are very efficient and can carry a charge for the span of a few days. Using one these high capacity battery packs with our lower power consumption Raspberry Pi would allow us to use QwikBox for multiple days without the need for recharging the power source. The battery packs in consideration are made by a bunch of different brands but hover around the 20000mA range which is more than enough to power the system. These battery packs will enforce the idea of having a portable QwikBox unit.

4. The High Efficiency Video Coding codec is currently the smallest compressed encoding standard that a video can be while still maintaining high definition quality. What this means for us is that we can use this standard to compress any video captured by QwikBox and compress it down to a very small file size. This will enable us to then upload the that same high definition video onto a web server a lot faster using wireless communication data transfer. Currently H.265 is the leading standard in data transfer because it's zero quality loss and 20% original file size compression algorithm. This standard is currently being used for service like Netflix and local news channels because the money saving aspect of less bandwidth usage based on file size. This technology is crucial in the development of QwikBox and will allows to better utilize every resource available in the Raspberry Pi.

5. An important part of the QwikBox system is having the ability to adapt any high definition video stream to a usable interface for the Raspberry Pi. This can done in many different ways using different types of tools from software to hardware. A typical way to make HDMI input some accessible is by changing the format to a USB input interface while maintaining the proper quality and format of the original video. This is where an HDMI to USB input conversion hardware will come in handy for universal use on the Raspberry Pi.

6. The Linux operating system is the main OS for all Raspberry Pi systems because of its general accessibility, ease of usage, reliability, and lightweight performance. When comparing this to other system such as Windows or Mac, it becomes clear that Linux is clearly superior for projects such as this one. Linux also offers cross platform accessibility to other microcontroller because which will make the QwikBox system portable to other platforms. Using the Linux OS will also mean that any distribution of Linux can be used on the Raspberry Pi allowing for greater control over speed and performance. Also Linux offers a lot

of documentation from online communities giving us a greater understanding on to best use the Linux operating system.

7. Another important aspect to the QwikBox is the video conversion method that will be used. There are many ways to go about doing this but each method includes many different advantage and disadvantages. Software such as VLC media player will allow us to easy convert video from any format to the H.265 codec but at a slower speed than physical hardware conversion would offer. Although advantage to this method means is the fact that the compression is done completely by the Raspberry Pi. This is good because it gives us greater control over the entire process through the Raspberry Pi operating system. From here we can see the entire process happen and determine when is the best time for each step from recording to compressing to uploading to happen.

8. By using Amazon's Web service as a storage solution for QwikBox, it given the user more control over their uploaded videos. As of right now the biggest competitor for sports film hosting is Hudl. We plan to use our application and Amazon's storage to support our on hosting service for videos. Amazon is also very reliable and offers automatic backups of all uploaded files. This allows us to guarantee that will video will be saved for future use to our users.

3.3 Strategic Components and Part Selections

It was determined that the best option for the brains of the QwikBox system would be Raspberry Pi 3. This is the newest version of the Raspberry Pi that offers the greatest amount of performance in its price range. The Raspberry Pi 3 comes packed with many features that gives it an over its competitors. Below is a list of the selected parts that will be included in the QwikBox build. Every part has been extensively researched and is completely compatible with the Raspberry Pi interface. These have been determined to be the best parts for accomplishing our project goals.

List of selected parts:

- Raspberry Pi 3 Model B
- Skywire Embedded 3G Modem
- Intocircuit 26000mAh Portable Battery Pack
- Rongyuxuan HDMI to USB 3.0
- Sandisk Ultra 32 GB SD Card
- Custom Printed Circuit Board (PCB)

Single-board Computers

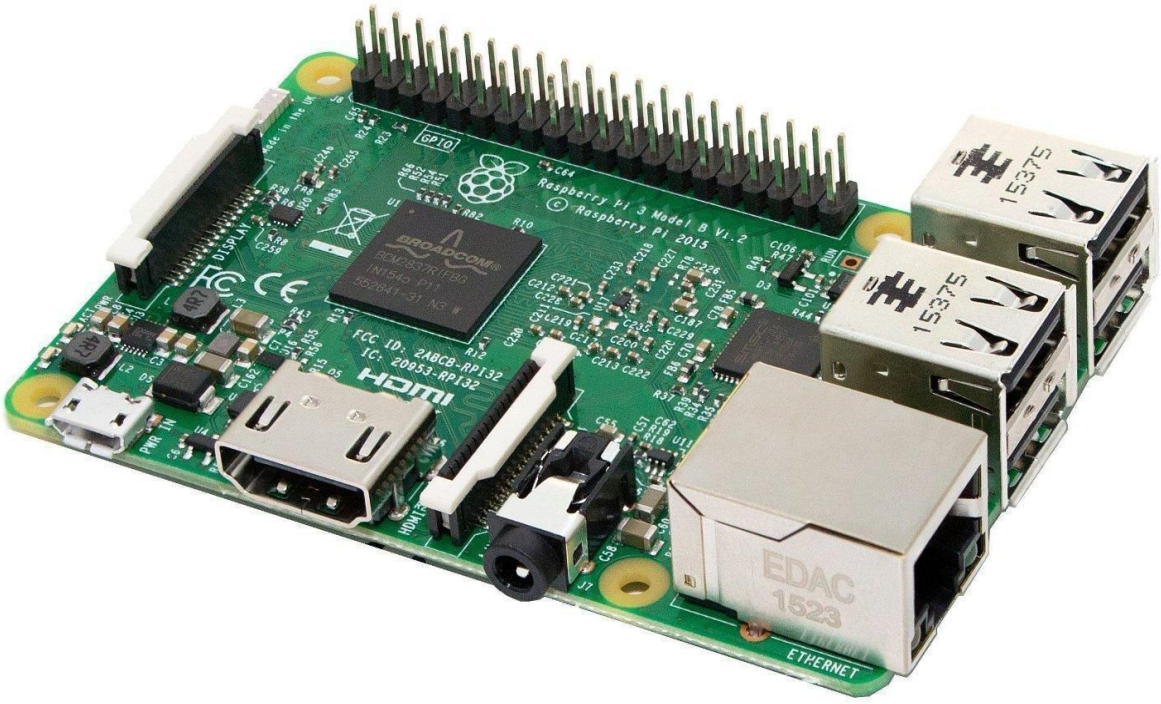
Among everything else, choosing the system controller for the QwikBox system was the toughest decision. There are many requirements that a part must fit in order to be considered as a potential part. Most important among these requirements was the

cost vs performance ratio that we needed the system to fit. With the bigger more powerful system available there were many drawbacks to consider like size of the dimensions and the power consumption of the unit. Ultimately, the Raspberry Pi 3 Model B (Picture 3 – 3) was chosen for various reason along with the ones just mentioned above. This latest model includes 802.11n Wi-Fi, Bluetooth 4.0, 1GB of SDRAM, a VideoCore IV graphics processing unit running at 400MHz, and quad-core 64-bit ARM Cortex A53 running at 1.2GHz. It also can be a usable desktop computer running a Linux distro and is widely available at retailers for $35.

Advantages

- Huge community support
- Affordable price
- Lost power consumption of boards
- Industry standard

Raspberry Pi 3 Model B



Picture 3 – 3

Although there are other fantastic alternatives available that could enhance the performance of QwikBox, we determined them to be inferior to the Raspberry Pi system due to consideration of our project goals. The other interesting considerations that could have been used for QwikBox include the Pandaboard-ES Rev B3 and Banana Pi M64.

These boards both feature many great tools that would make QwikBox an overall faster product. Unfortunately, there are also a few downfalls to consider with these units such as efficiency and cost. The Pandaboard (Pcicture 3 – 4) offers 802.11n Wi-Fi, Bluetooth 4.0, 1GB of DDR2 RAM, a POWERVR SGX540 gpu running at 384MHz, and a dual-core ARM Cortex A9 running 1.2 GHz. The Pandaboard was designed for high definition video output and runs natively at 1080p with 30 frame per second. This allows the Pandaboard to perform very well under large loads of graphics encoding. This would benefit the QwikBox system immensely. The only drawbacks are that this single-board computer cost around $180 and is about four times the size of the Raspberry Pi 3 Model B. This would set us back one of our goals to be cost efficient because it fills up almost all of the budget by just purchasing the unit. Overall, the Pandaboard SE [25] would be a great addition to QwikBox and we would prefer to use it over the Raspberry Pi.

<u>Advantages</u>

- Native 1080p video
- Most powerful board

Pandaboard-ES Rev B3

Picture 3 – 4

The other single-board computer that we considered using was the Banana Pi M64 (Picture 3 – 5). The Banana Pi is among the most powerful units we considered for QwikBox. For one, it runs on 64-bit architecture which allows it to run many more instruction sets at any given time then the 32-bit architectures of the Raspberry Pi and Pandaboard. The Banana Pi [26] also comes packed with 802.11n Wi-Fi, Bluetooth 4.0, 2GB LPDDR3 RAM, a dual-core MALI-400 MP2 running at 500MHz, and a quad-core ARM A53 running at 2GHz. These specifications make it the must power consumption unit and the one with the fastest processing speeds. This unit is able to process video outputs at 4K which would be a huge benefit for the QwikBox system to have. With greater specifications then the Raspberry Pi the only drawback is the $80 price and the increased consumption in power.

Advantages

- Runs on 64-bit architecture
- Reasonable price per performance range
- Great for video processing

Banana Pi M64

Picture 3 – 5

Wireless 4G/5G Antennas

The second biggest dilemma in our project was choosing the proper wireless communication devices that would allow us to achieve our goals for this project. The wireless antenna had to be able to perform a couple number of task and needed to be able to be adapted to a printed circuit board. Achieving this will satisfy our requirements for this project. Furthermore, it will present us with an opportunity to replace parts in the future for better efficiency and performance. Among all the antenna we researched there was one in particular that would enable us to easy install it to a printed circuit board. This antenna is called Skywire Embedded 3G Modem. It was determined that this communication device is the best fit for this project.

The Skywire 4G LTE CAT 3 Embedded Modem (Picture 3 – 6) [27] features a 20-pin architecture which can be used in numerous ways. This modem has a voltage input of around 5.5 V and runs on around 29 mA. It also is able to withstand a vast range of temperature and has speeds of around 100 Mbps download and 50 Mbps upload. Since the chip is so small it can be added to the project Raspberry Pi with little interface towards space. The Skywire 4G LTE has industry standard wireless communication bands which will remain relevant in future.

<u>Advantages</u>

- FCC & Verizon OD Certified
- Longest network life
- Improved reception with RX diversity
- GPS+GLONASS option
- Bundled data plans available

Skywire 4G LTE CAT 3 Embedded Modem

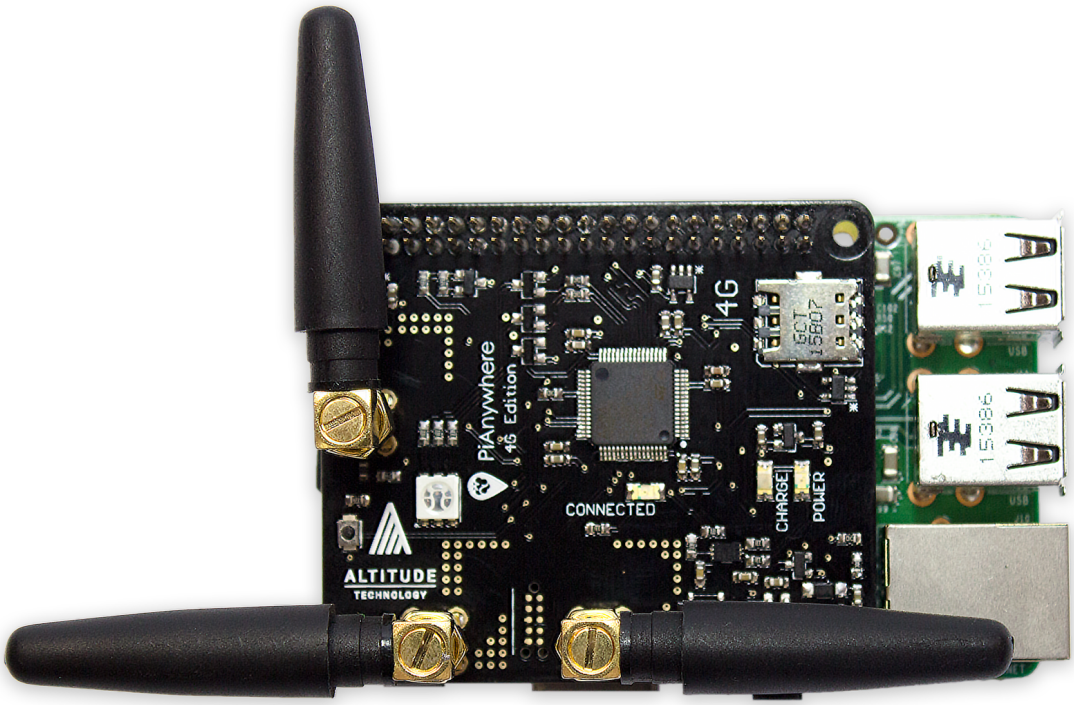
Picture 3 – 6

The other options for the wireless 4G LTE antenna is the PiAnywhere 4G Shield (Picture 3 – 7) [28] and the SX1272 LoRa Shield for Raspberry Pi. The PiAnywhere is unique in its design specifications but features a plethora of great benefits to our project. The PiAnywhere 4G is shield board that is meant to be installed on top of Raspberry Pi 3. This is the only design we researched that has a wireless communication antenna adaptable directly to a Raspberry Pi. This shield comes with 100 Mbps download and 50Mbps upload and can be used with an micro sim. It's comes with four separate communication bands that allow for better connectivity. This shield must be powered with a micro USB power cable and is a reasonable size for our project. This unit cost around $300 and would be very expensive to use with QwikBox. Although, the PiAnywhere 4G shield connects directly to the Raspberry Pi it would be the easiest to install and setup for QwikBox. Unfortunately, using this would not require a printed circuit board and would not satisfy that part of our project requirements.

Advantages

- Supports any micro sim
- Triggered events through shield
- High-efficiency power regulation up to 3 amps
- GPS optional
- Easy install and setup

PiAnywhere 4G Shield



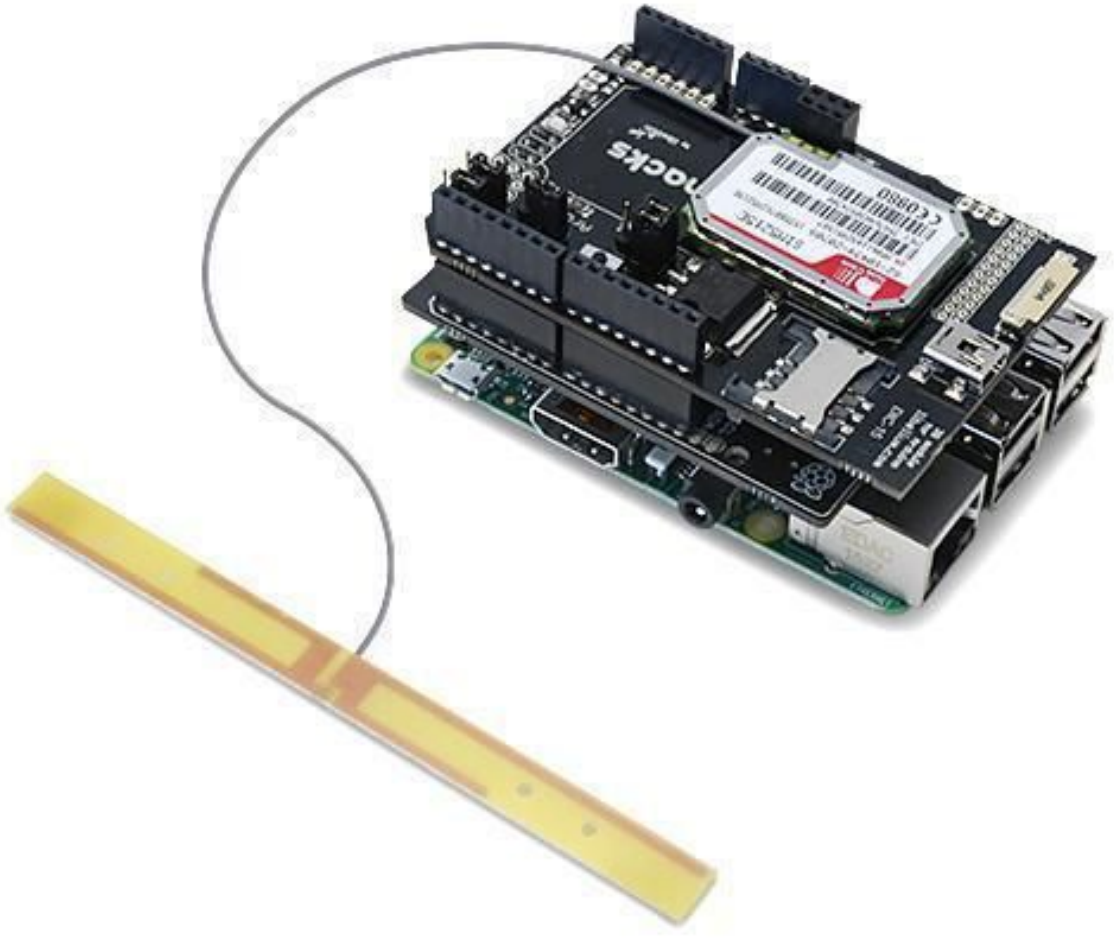
Picture 3 – 7

The other type of antenna that could have been used is an Arduino board with a wireless communication module (Picture 3 – 8). It would be set up the same way as the one pictured above but would feature a Raspberry connected to an Arduino which is connected to a wireless 3G modem. This modem is simply called SIM5215A and is able to connect to any Arduino. By utilizing this setup we could isolate the various functions of each component. This would allow the Arduino to control all the uploading of videos functions and the Raspberry Pi would take care of the entire compression process. Before the IoT caught up to technology available now, this was the best way to enable wireless data communication through 3G. While this method of providing wireless data transfer is obsolete it would absolutely work in our design of QwikBox.

Advantages

- External compression

Arduino and 3G Shield with Raspberry Pi



Picture 3 – 8

Portable Battery Banks

For the QwikBox system to perform well, it needs to be powered by a decent power supply. This power supply will be responsible for powering the Raspberry Pi and the Skywire 4G LTE antenna. This means that the power supply will need to be large enough to power the system all day and produce a steady enough current to keep everything. This will keep the QwikBox system portable and reliable. Another thing that needs to be included in the portable battery bank is a USB port. This is necessary in order to fulfill our requirements for the project.

The first battery pack we considered from our research was the Mophie Powerstation [22]. This battery supply has a capacity of 4,000 mAh. It features a USB port that is able to power almost any device. The battery output is able to produce three levels of voltage 500 mA, 1 A, or 2.1 A and cost around $20. The battery is a Lithium Ion battery which provides a great deal of longevity and performance. The dimensions of itself are a decently workable size into the QwikBox system. Overall, the Mophie Powerstation would be able to power the entire system. The only issue with this power supply is that is not able to output power while charging and wouldn't be able to last the entirety of a day. It was estimated that the unit would only last about 2 hours. That would make it unsuitable to use within the QwikBox system.

<u>Advantages</u>

- Cheap price point
- Multiple power output modes
- Lithium Ion battery

Mophie Powerstation

Picture 3 – 9

The next power supply researched was the Intocircuit 26,000 mAh portable battery pack (Picture 3 – 10) [21]. This power supply unit comes with packed with one of the largest charge capacities that is available and can charge a normal laptop battery. This power supply features a six hour charging time and can be used with many different devices. Since it's able to match the power output of most DC charging cables, it will be a great option to power the QwikBox system. It boast various output level up to 4 A and 12 V with the extremely powerful Li-Polymer battery. These battery are new industry standard and are very reliable and can be used over 1000+ cycles. Another great feature of this product is that it's made with an aluminum body so it can withstand drops and won't expose the battery to the elements. This would make Intocircuit's battery pack one most reliable power supplies we've researched. This unit cost $60 and has the best price for performance ratio we could find. It was estimated that this battery pack could power QwikBox for the entirety of the day.

Advantages

- Li-Polymer battery
- Numerous power level outputs
- Fast charging ready
- Best cost vs performance ratio
- Can power QwikBox for a the whole day

Intocircuit 26,000 mAh Portable Battery Pack

Picture 3 – 10

The last battery pack we researched was the BESTEK 300 W Power Inverter DC 12 V to AC 110 V (Picture 3 – 11) [20]. This battery pack is able to power a personal computer for about 6 hours and charges directly from a car port cigarette lighter. This means that this battery pack would be very portable and that only a car would need to power it. Since it features an extremely high output, it would have no problem powering the QwikBox system. It was determined that the battery could last on better end of a week just powering QwikBox. This device also features multiple outlet ports for which could be used in a tight situation to charge a camera as well. This would make QwikBox a very diverse product in what it can do. The only major drawback is the fact that this unit is massive and would essentially double the weight and size of QwikBox. This is not good for what we want QwikBox to be and thus makes this battery pack a less likely candidate to be used.

Advantages

- Multiple ports for other uses
- Largest battery unit
- Can be charged with a car

BESTEK 300W Power Inverter

Picture 3 – 11

Video Converter

The next aspect of the project that was researched was the video input device. We had to figure out a way to get high definition video from a camera via HDMI output. This output had to be in a format that would allow use to take it as input to the Raspberry Pi without loss of quality. The input that the Raspberry Pi 3 Model B can take is anything from a USB. While it does have an HDMI port, it is only usable as output. This means that in order for the video to usable we have to convert raw HDMI video data into a USB interface that can be used on a Linux operating system. This can be done in many ways and there are lot product that have this type of conversion ability.

The Rongyuxuan HDMI to USB 3.0 1080p Video and Audio Capture Device (Picture 3 – 12) is a nifty little device that enables the quick conversion of an HDMI stream to USB 3.0 [30]. This is super beneficial to the Raspberry Pi because USB 3.0 is one that fast data transfer protocols available. This interface structure is import because it allows this unit to transfer 1080p video at 60 Hz which is the standard high definition output. The device is also able to maintain an audio quality of 48 KHz which allows that standard that the DVD format operates on. This stream encoder is perfect for QwikBox and only consumes around 2.5 W of power. With the good enough battery supply this piece of the QwikBox will have no trouble performing. One of the only drawbacks of this device is the $170 price of the unit. Since this technology is relatively new, the price is understandable. The team has determined it would in our best interest to use a device such as this one to convert HDMI video into a USB format usable on the Raspberry Pi.

<u>Advantages</u>

- No loss of video or audio quality
- Operates with low power consumption
- Industry standard formatting

Rongyuxuan HDMI to USB 3.0

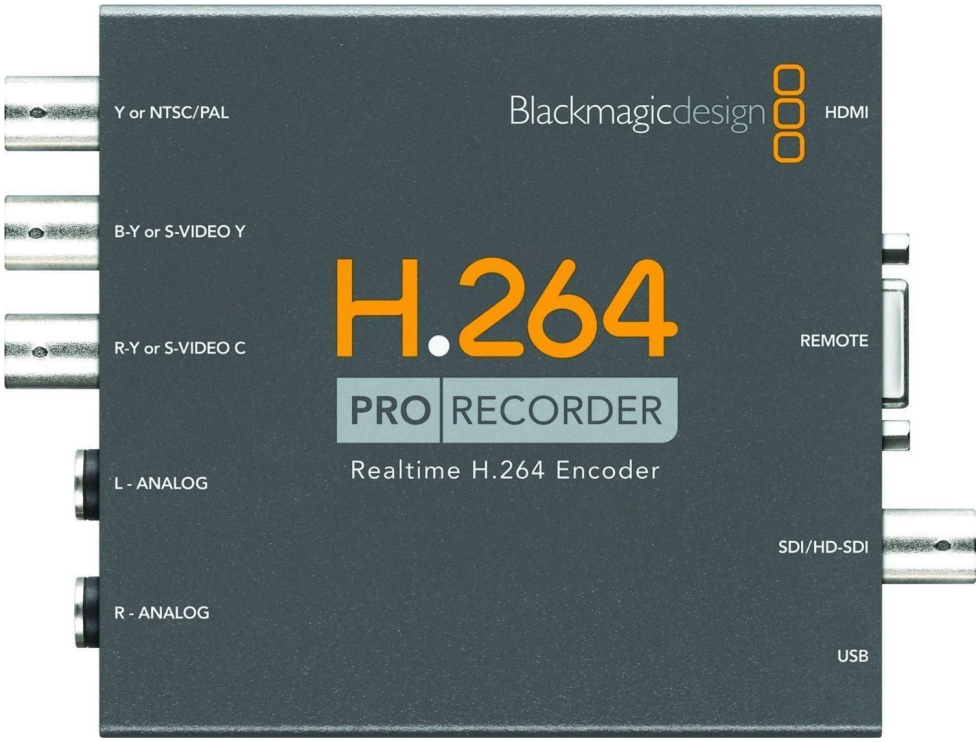
Picture 3 – 12

The other stream encoder device is called the Blackmagic Design H.264 Pro Recorder (Picture 3 – 13) [24]. This device is able to convert and compress video to the h.264 codec externally from the Raspberry Pi. This would enable us to separate each part of the QwikBox system to individual units. This device in particular would allow the HDMI video output from the camera to be compressed and converted instantly without the need of the Raspberry Pi. Instead it would pass the video to it already encoded. This would take a huge load off the Raspberry Pi and allow it to focus on moving the files the antenna and uploading them. While it may be a huge benefit to operate the system this way, the Blackmagic Recorder only supports h.264 video encoding. Since h.264 is an industry standard format it will produce high definition video with no quality loss. Unfortunately, this isn't optimal for the QwikBox system but it can still be considered for this project. Another factor is that the price of this device is around $470. This would be a huge impact on our budget and would not satisfy our project requirements.

<u>Advantages</u>

- External video compression
- No quality loss
- Power efficiency

Blackmagic Recorder



Picture 3 – 13

The other possible interface that could have been adapted to the QwikBox system is the Matrox Maevex [29] h.264 HDMI over IP converter (Picture 3 – 14). This device takes in HDMI input and is able to compress it down to the h.264 codec format. It then transfers the video through IP Ethernet cable to an online web server. Since the Raspberry Pi is able to take in input via an Ethernet cable to which it could then save the files to for upload to the server. This unit like the device mentioned above takes care of the encoding externally from the Raspberry Pi allowing it to focus on storing and uploading video files. Also since it's using the h.264 format it will be in high definition and without video quality loss. The downfalls for this come with its $300 price and high power consumption. On top of that, it is unable to use the newest standard of H.265 codec which would not fit our project requirements.

Advantages

- Low bandwidth usage
- Fastest compression device
- External video compression
- No video quality loss

Matrox Maevex HDMI over IP

Picture 3 – 14

3.4 Possible Architectures and Related Diagrams

In the design of QwikBox there are multiple functioning parts that come together to make a complete system. This means that each individual part has its own functions and architectural design. Furthermore, each company has its own trade secrets that might protect certain design concepts for being released to the public. The following images below are of the design concepts and architecture of each device. Some parts contain a vast amount of information while other contain little to none. The first part we'll look at is the Raspberry Pi. The main CPU [31] is pictured below and details the many interfaces supported by the Raspberry Pi (Diagram 3 – 1).

Raspberry Pi ARM CPU Architecture

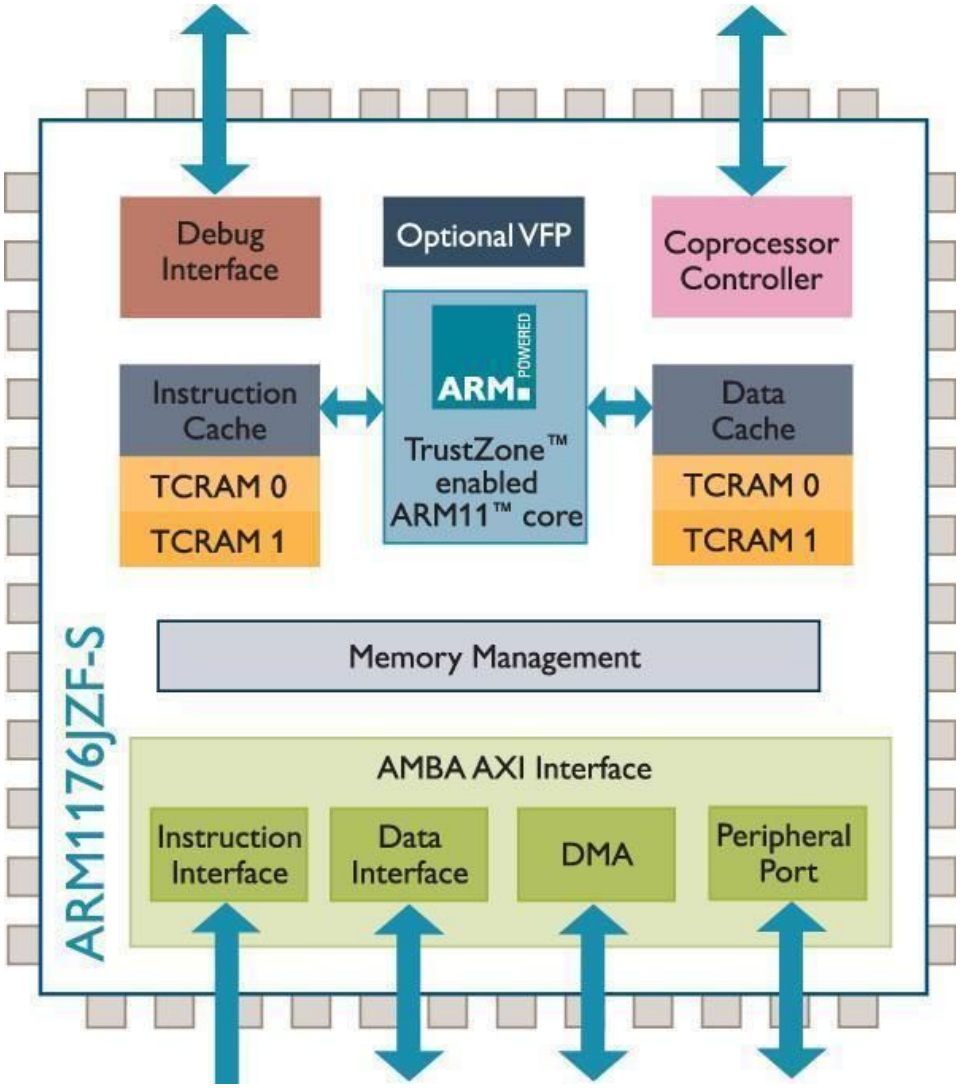


Diagram 3 – 1

The other major component of the QwikBox system is the Skywire 4G LTE antenna which NimBeLink provides vast amount information on. Since this device can be used with many different types of projects, it's important to understand the architecture in which how the device operates. Below is a block diagram that features the basics of how the Skywire 4G LTE antenna operates (Diagram 3 – 2).

Skywire 4G LTE Block Diagram

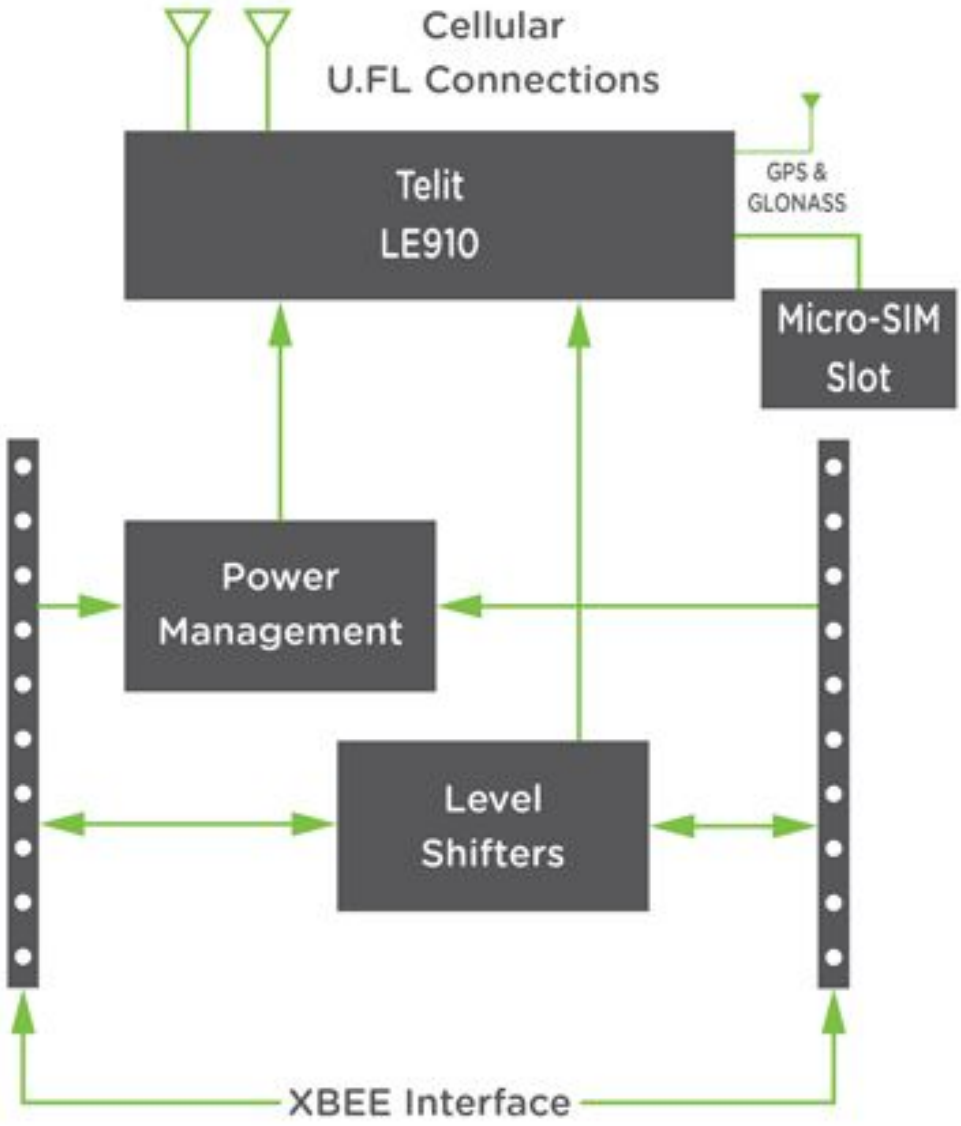


Diagram 3 – 2

In order to obtain a better understanding on HDMI works, we did research on how the USB interface converter works. We discovered that the converter device is able to simply take the Data EDID pin from the HDMI cable output and transfer that into the

format of a usable USB interface. The converter also captures the Clock EDID, Ground, and Power pins from the HDMI cable. In the diagram below, it shown how exactly that converter would work. It features A USB cable and module that manipulates the pin connections to create the USB interface (Diagram 3 – 3 and Diagram 3 – 4) [23].

HDMI to USB 3.0 Conversion

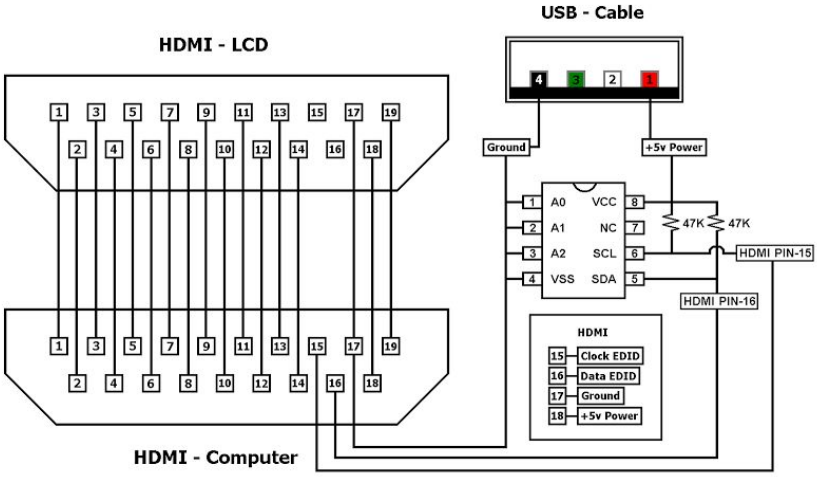


Diagram 3 – 3

HDMI Interface

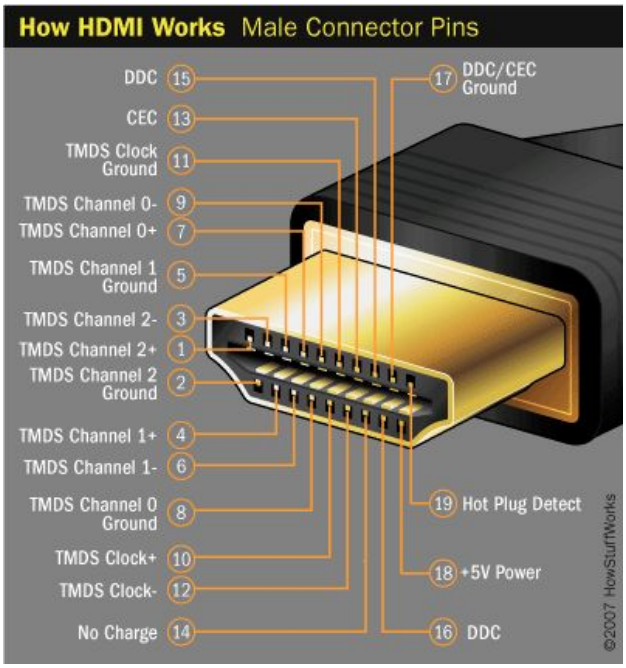


Diagram 3 – 4

3.5 Parts Selection Summary

A Raspberry Pi 3 Model B was the best option we could find to serve the purpose of the central processing device/motherboard for this project. This is because it has the ability to run a Linux-based operating system that has an actual graphical interface to make programming the device easier. It also comes with an Ethernet port, multiple USB ports, an HDMI port, Bluetooth technology and Wi-Fi all included. This has saved us time and resources since the Raspberry Pi is widely available and cost as little as $35.

The next design task involved the process of retrieving video from the video camera as quickly as possible. At first, we thought the best way to do this was to pull the video file from the camera's SD card through a USB mass storage interface. After experiments with the video cameras supplied to us from our sponsor, it became clear that this was not going to be our best option. This is because the cameras don't allow other devices to access the SD card storage while in use, and many require a power supply as well. Since waiting until the end of a sporting event does not satisfy our requirements, we had to explore other options.

There is a company called Hudl which has attempted to tackle a similar project to this one. They use an HDMI/Bluetooth interface to projects real time video to a nearby tablet or phone. The user of this tablet or phone then presses a button to start each video clip, and presses it again to end it. The video is then converted and sent elsewhere. While our objective is slightly different once the video file is produced, this method definitely has proven to inspire our design. For our system, the video camera will be connected to the QwikBox through an HDMI cord. Inside the box, the HDMI interface will be converted to a USB interface since the Raspberry Pi 3 does not have a HDMI input. After interface conversion, the video is streamed to the Raspberry Pi with a very slight delay. Now comes the mobile application's main use, as it will have a button that the camera operator presses at the beginning of every clip and again to end the clip. This video will be recorded by the Raspberry Pi between presses, and encoded to the correct codec after it has been recorded. The encoding and conversion will be done by VideoLAN's VLC media program, or a program with similar capabilities. Once the video file is in the correct format, it will be added to a queue to be uploaded. The process will then repeat for every clip until the end of the sporting event.

The system will be powered by a Intocircuit High Capacity Power Castle Portable Charger External Battery Pack that will last approximately 9 hours under full stress. This battery was chosen because of the large capacity, USB interface, and relatively inexpensive cost. The QwikBox will also have a custom PCB that implements an antenna that connects to Verizon's 4G wireless network. It will have two sets of 10 pins with 2mm pitch female receptacles and will connect to the Raspberry Pi 3 through a USB interface.

4. Related Standards and Realistic Design Constraints

4.1 Standards

4.1.1 ANSI Standards

Throughout the team's research, a couple of standards have been determined as beneficial to the design of QwikBox. The newest standard of high definition video compression is called High Efficiency Video Encoding (HEVC). It is otherwise known as H265 encoding. H265 is needed for this project because of the number of benefits it provides compared to its predecessor, H264 (AVC). H264 was first developed in 2003 and it succeeded MPEG-2 encoding. H265, developed in 2013, provides a 40%-50% bit rate reduction over H264 without any loss in video quality. This allows the file size of videos to become 40% - 50% smaller to save space on a server. H265 is without a doubt the codec of the future as it supports HD, 2K, and 4K video at a relatively smaller file size. This is opportune for broadcasting, streaming, online file storage, and many more possibilities. Table 4 – 1 below highlights the key features of the H265 HEVC format.

Video Compress Codec Standards

	H.265/HEVC	H.264/AVC
Names	MPEG-H, HEVC, Part 2	MPEG 4 Part 10, AVC
Approved Date	2013	2003
Progression	Successor to H.264/AVC	Successor to MPEG-2
Key Improvement	● 40%-50% bit rate reduction compared with H.264 at the same visual quality ● It is likely to implement Ultra HD, 2K, 4K for Broadcast and Online	● 40%-50% bit rate reduction compared with MPEG-2 ● Available to deliver HD sources for Broadcast and Online
Support up to 8K	Yes	No, only up to 4K
Support up to 300 fps	Yes	No, only up to 59.94 fps

Table 4 – 1

In this diagram below, it compares the file compression sizes. The Q in Diagram 4 – 1 stands for quality where the closer to 1 the Q is the greatest raw video quality present. Thus, a Q of 24 is 1080p is standard assessments. (30 is 720p and 40 is 460p)

H.265 vs. H.264 Size Comparison

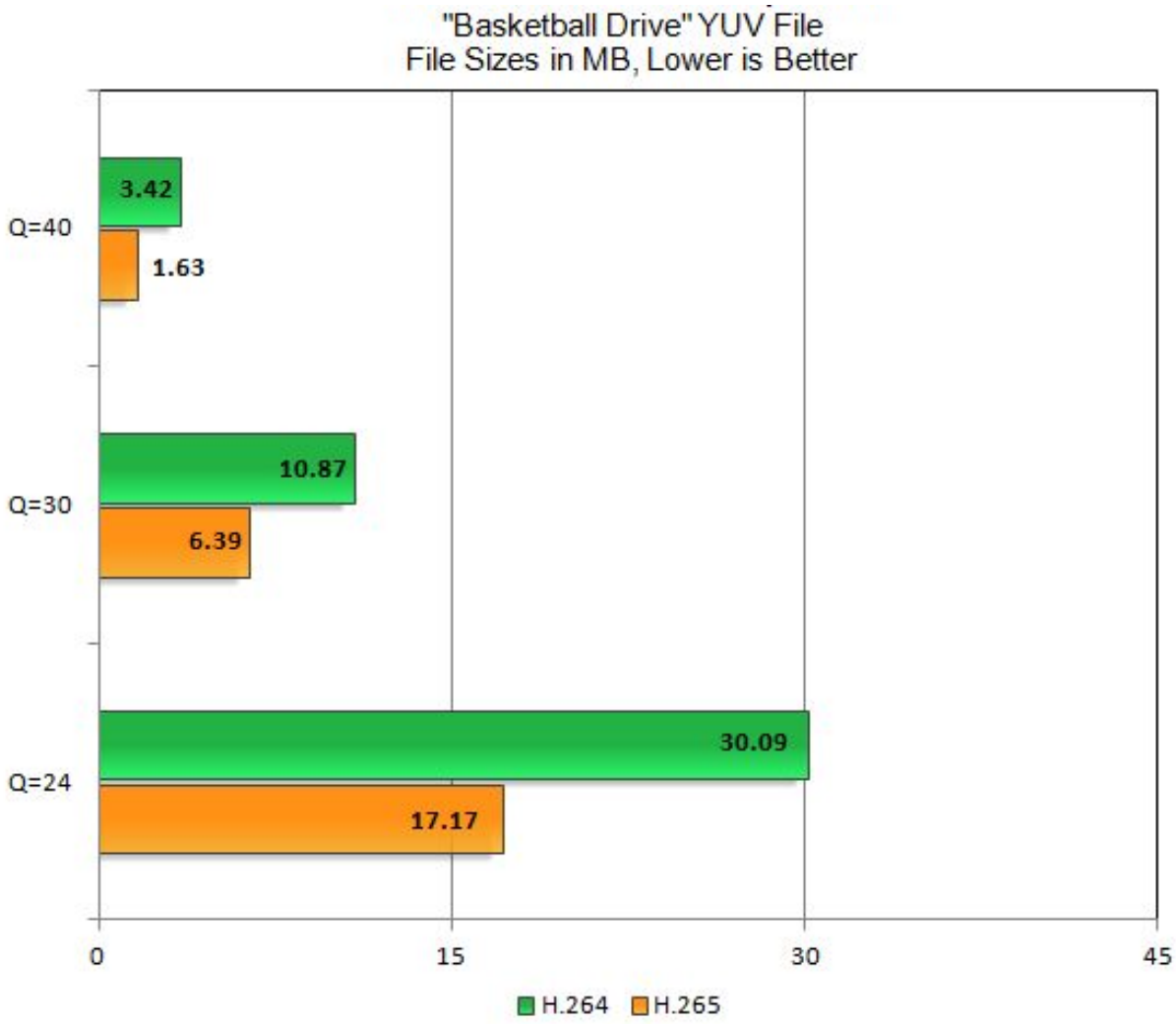


Diagram 4 – 1

4.1.2 Design Impact of Relevant Standards

The standards that are being used and considered for this project will have a minimum impact on the design of QwikBox. Since the software we are using takes care of any codec issues we would have during the encoding process, there isn't any reason for the design to be change. The newest HEVC will maintain the high definition video

quality and will be able to convert video files to this format using VLC media player. VLC features multiple tools which will allow a user to compress video sizes and edit files. This is extremely useful in our design because it keeps the entire process within one program. Since the Raspberry Pi is able run VLC media player through Linux, this whole process will become more simplified. These standards of high definition video standards can are a major aspect to this project.

4.2 Realistic Design Constraints

4.2.1 Economic and Time Constraints

With consideration to economic restraints, the team's biggest issue will be creating a working prototype by required date. This is because a lot of designing is needed to be done before testing can begin. To get all the parts working together in the correct environment will prove to be the most difficult. Some parts will require programming and others will require physical installation in the QwikBox system. Once this can be achieved, testing will begin which also take a great amount of time to perfect. We want QwikBox to be completely working by the end of this point. While testing if anything doesn't work then that component will need to be implemented again and retested. Since there is no telling how long this process could take, there is a great chance that the team will not have functional prototype by the end of the due date.

We were given a $1000 budget for this project, but it has been a priority to make this project as cost-efficient as possible. The end product will need to be able to be marketed as a cheaper, and in many aspects a better alternative to the Hudl Sideline product that already exists. The ability to produce a product that can be purchased for much less than $900 a year will be highly beneficial to the youth and high school sports around the state and possibly even the entire country. The only time constraint is to have the end device finished and thoroughly tested before the beginning of the 2017 football season. This will be easily accomplished since the time frame of this project ends in April or May of 2017, months before the start of the football season. However, there are spring leagues and scrimmage games played in the springtime every year. These games will serve as vital testing environments for our project.

4.2.2 Social and Safety Constraints

Based on the products capabilities a considerable safety constraint is the ability to provide the customer with the ability to review the actions of the players on the field. Certain implications of reviewing allow the customer to give constructive criticism to the individual players. As one athlete gets a more focused approach to apprehending the game we can see an increase in safety for the overall sport. Knowing the skills to handle a blow during a contact sport or where to position your player during a play is very important. Not only can we improve upon the skills of our players but we can also

replay any falsely judged plays or misconduct during a game. Currently, most non-professional leagues do not every game and have no way of reviewing a play for protecting players from uncalled fouls. As a major safety constraint if a certain player is actively disrespecting the sport by breaking rules, there is now a way to properly discuss a review of the wrongdoing. As for both safety issues concerning the player's health and skills we require the ability to quickly review these plays.

4.2.3 Manufacturability and Sustainability Constraints

Since QwikBox can be created using the sum of purchased parts, it would be relatively easy to manufacture any number of QwikBoxes, given that those parts are readily available. These parts include a Raspberry Pi 3, micro SD storage, wireless antenna, PCB, and power supply. The design scheme allows these parts to be purchased at a lost cost and grants a quick build time due to its plug and play nature. Once all the back end coding and implementation complete the QwikBox will be a completely standalone system. This will allow for the greatest manufacturing output and will hopefully decrease producing cost of QwikBox in the future.

The QwikBox will be designed for sustainability and will grant many hours of usage throughout its life cycle. Considering that any part can also be replaced at any time gives the QwikBox a great sense of longevity. The parts that QwikBox uses them are designed with high production value. This will produce the greatest quality of QwikBox.

5. Project Hardware and Software Design Details

5.1 Initial Design Architectures and Related Diagrams

The system will consist of the following components:

- MicroSD card for storage
- Raspberry Pi 3 computing device which includes an HDMI port, USB ports, an Ethernet port, and Wi-Fi
- Power supply and battery
- Verizon 4G LTE antenna embedded in a custom PCB design
- HDMI to USB conversion interface

Initially, we had quite a different set of components in mind. As we continued to research and experiment, we came across numerous issues that caused us to change up the design. Arguably the most monumental change that was made had to do with the means of retrieving the video from the camera. In our initial design, we planned to pull the already-processed video files straight from the video camera. To do this, we gathered some data after testing transfer times with various types of storage. Data was gathered using a hard drive, and a solid state drive.

Test information:
160 clips
Total size: 4.96 gb
Average clip size: 31 Mb
Max clip size: 104.85 Mb (53 secs) – 9.21 sec transfer time
Min clip size: 6.02 Mb (2 secs) – less than one second transfer time
The second largest clip was 64 Mb.
Model used: Sony HDR-CX210.
Resolution: 1920x1080
Frame Rate: 29 frames per second
Audio bitrate: 256 kbps

Hard Drive:
Longest Transfer Time: 104.85 Mb in 9.21 seconds
Shortest Transfer Time: 60.02 Mb in 0.8 seconds

Solid State Drive:
Longest Transfer Time: 104.85 Mb in 0.5 seconds
Shortest Transfer Time: 60.02 Mb in negligible time

As can be seen, the solid state drive is considerably faster when using a SATA connection, though the speed was still limited by the nature of the camera's USB connection speeds. However, as we experimented more with the cameras we were provided, it became clear that our original plan needed altering. The cameras were simply not able to be used at the same time their storage was being accessed. This would prevent our device from being able to do a true real-time upload, and would cause serious problems for its users. We began to pursue other directions where we weren't required to access a camera's storage during a sporting event.

An online interview and explanation of a system that used a Raspberry Pi and a HDMI to USB capture device to record streaming video and save it to the device was found, and began our inspiration for the current design. If the Raspberry Pi was able to receive a streaming signal through one of its USB ports and save/compress the video from this signal, it would solve the issues we were having. However, we would sacrifice storage speeds if this route was followed since a Raspberry Pi uses a MicroSD card as its main storage. The low price and extensive features of the Raspberry Pi is what made the decision easy.

The implementation of the Raspberry Pi 3 into the system eliminated the need for other parts by combining them all into a small reliable microcontroller. Our initial list of components and hardware Diagram 5 - 1 were as follows:

- Storage device
- Processor and memory
- HDMI port
- Ethernet port
- Power supply
- Verizon 4G and 5G LTE antenna
- Wi-Fi antenna
- PCB

Initial System Design

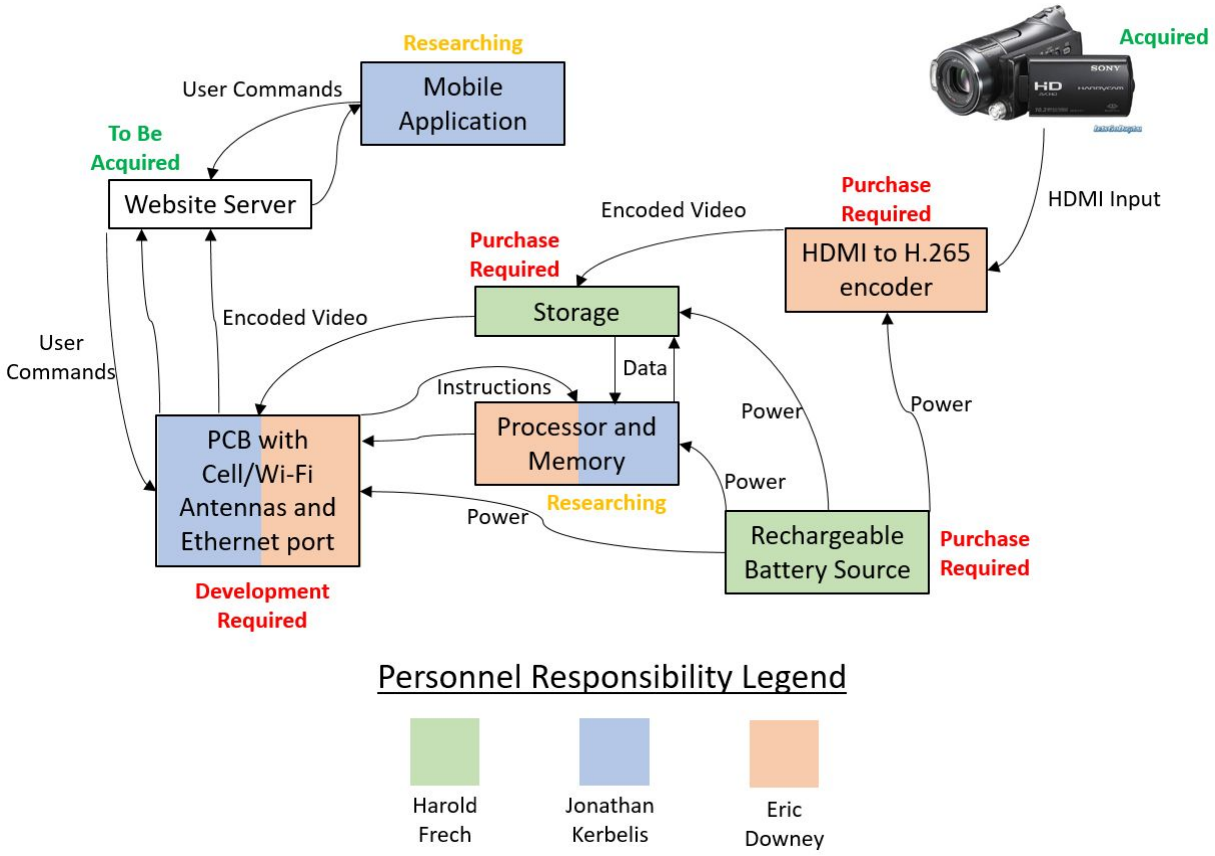


Diagram 5 – 1

Through tireless research, the newer list of components has become much more specific as we have gotten to the intricacies of the project. Through experimentation we discovered that much of our design would need to be changed and the new design for the system is discussed in the following sections.

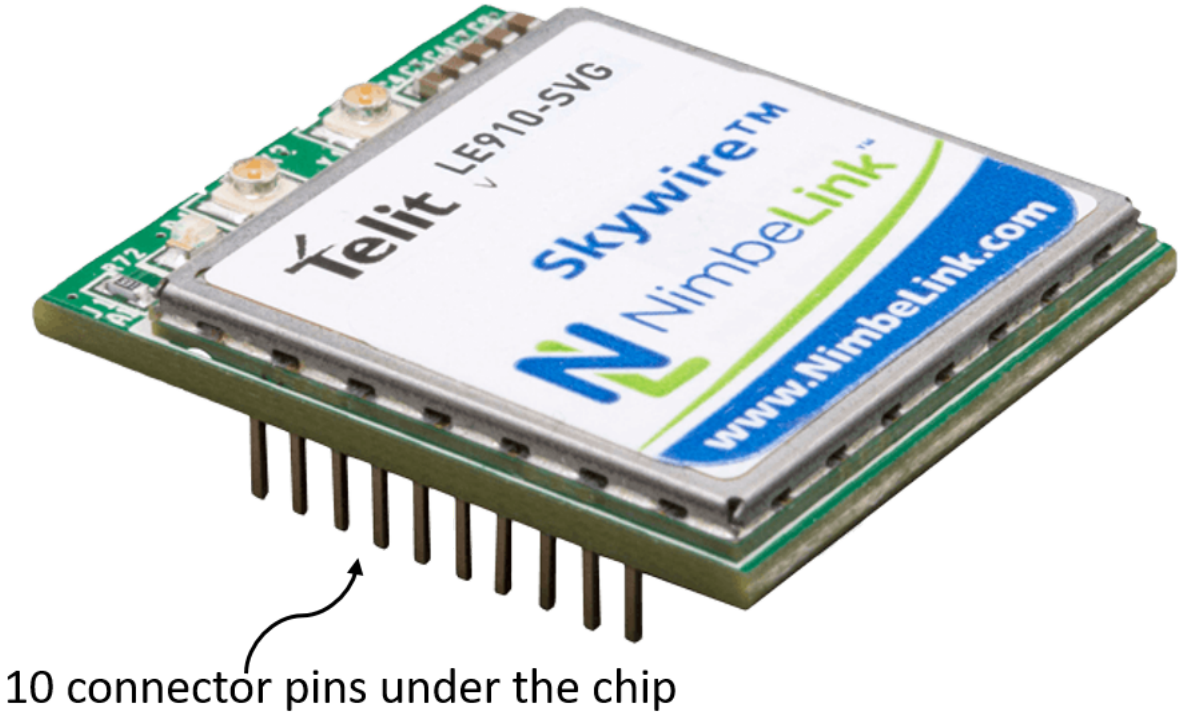
5.2 Cell Comms / PCB, Breadboard Test, and Schematics

The communication aspect of this program will require a connection to a data service provider. In order to ensure this data service is able to connect properly we decided to consult a Verizon representative. During this consultation we discussed the use of a certified LTE cellular modem. The most appealing piece of hardware was the Skywire 4G LTE CAT 3 Embedded Modem. This piece of hardware is capable of handling up to 4G speeds for our requirements. This modem will only require a two sets of 10 pins with a 2mm pitch female receptacles. Currently there is development board created by NimBeLink that contains housing for the chip and a method connecting the development board with the raspberry pi 3. Considering the size of the Skywire Embedded Modem its dimensions are 29.0 mm x 33.60 mm x 10.73 mm.

5.2.1 Setup

Assembly begins by taking both the development board and the cellular modem chip out of the shipping materials and place them down. Using proper methods of handling the device we will want to make sure no excess static charge builds up on the individual handling the board and chip alike. The first thing done was the placement of the Skywire NL-SW-LTE-TSVG. There are 10 pins on each side, with visual confirmation, place the chips such that none of the pins are misaligned. This is a very crucial step when placing the chip considering if any of the pins are damaged this will result in a complete failure of the chip. Ensuring care and no extreme applications of pressure upon the chip will reduce risk of a critically damaging incident.

Skywire 4G LTE Connections



Picture 5 – 1

We can follow this step by connecting the U.FL connector to the X1 located on the top of the Skywire NL-SW-LTE-TSVG chip. When considering this connection there are two other locations for placement of this U.FL connector. The second location this U.FL could be placed is on the X3 connector. Connecting to this location will allow for use of the GPS feature. Some chips don't use GPS or if the user decides not to use it you will not need to attach this to the connector. The third location the user can attach to is the X2 location. This connection is made for the Skywire with the cellular diversity

antenna option. Currently for our project we only have one U.FL connections to the X1 location.

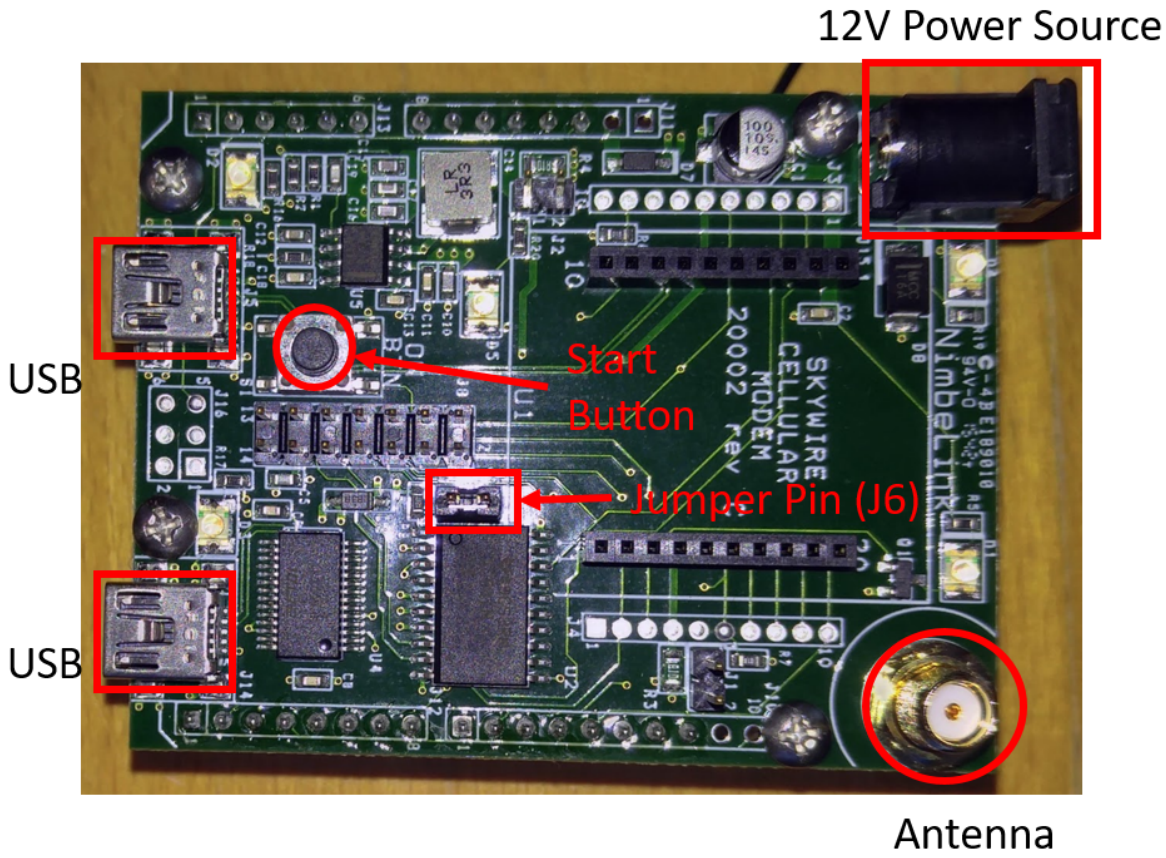
Skywire 4G LTE Antennas

Picture 5 – 2

Once that connection is made, move on to removing the antenna for placement on the board. The antenna is screwed clockwise onto the designated located at the upper right hand side of the board. Focusing on the power aspects we must check the J6 location is shorted with a 2 pin jumper. This ensures the board will work properly

when a power supply is added in the next step. Remove the 12 volt power supply and connect it to an outlet with power supply. This ensures power to the whole board and desired chip. The last component is the USB cable connection to your PC or Raspberry pi 3 in our project. We will test the board on PC before the final integration with the Raspberry pi 3.

Skywire Development Board



Picture 5 – 3

When setup is complete the next step is powering on the board. The user, for the specific Skywire NL-SW-LTE-TSVG chip, must press and hold the S1 button for between one and two seconds. There is a waiting time for software communication to occur which varies depending on your chip. There is no LED to let you know that the board is now active. In order to proceed we must use the PC to communicate with the development board. Using a properly working terminal such as PuTTY is important at this step. Using a improper terminal will cause failure in communicating with the development board. Upon using the terminal we have a few initial things to set up.

The first thing to consider is the Baud rate at 115,200bps. This is how many bits will be sent during communication in one second.

Baud Rate: 115,200 bps

The data must be set to 8 bits.

Data: 8bit

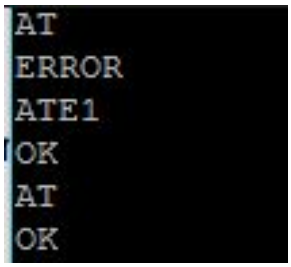
Without a parity bit, one stop bit and no overflow control. Parity bit allows the designer the ability to allow to error detection in the transmitted packets of data. This is useful to use when we want to ensure the data is stored properly. Under our circumstances we will be streaming video which makes using a parity bit unfeasible. The amount of data being sent through wireless communication is such a large amount that if every packet sent had an extra bit we would begin to see latency problems with the video. Also no overflow control is needed since our data is being sent out we will not run into overflow problems considering that the data is being fed in from the camera. These particular settings once properly set up we can proceed to communicate with the development board for the next part.

When beginning communication through a terminal we must ensure the development board is responding properly. Sending a simple command is the first step.

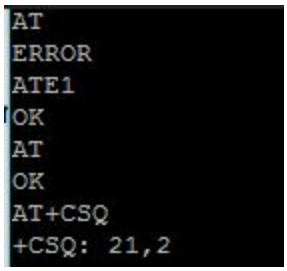
Start by typing the command “AT” and press enter. The proper response should be “OK”.

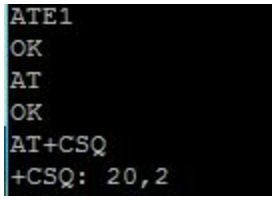
This response ensures that the device is working properly at this step.
On the other hand if your response is “error” then we must take an additional step.

Type the command “ATE1” and press enter. The proper response is “OK” and then proceed to typing the previous command and type the command “AT” and press enter with a response of “OK”.



Next type the command “AT+CSQ” and press enter then type the command “+CSQ” and press enter. This response is “+CSQ xx, yy” where xx is for the signal strength on a scale from 0 - 99, and the yy is for the bit error rate in percent. Testing the output twice to ensure stable reading we have posted the possible results of the development board.





In order to properly understand the information presented by the development board they have a breakdown for the responses that are given to the user (Table 5-1).

Development Board Signal Strength

Values of xx	Relative Signal Strength
0 – 9	Marginal: -113 dBm to -95 dBm
10 – 14	OK: -93 dBm to -85 dBm
15 – 19	Good: -83 dBm to -75 dBm
20 – 30	Excellent: -73 dBm to -53 dBm
31	Excellent: -51 dBm or greater
99	Not known or not detectable

Table 5 – 1

Based on the values given by the development board we receive values of xx and can use this table above to determine a range of decibel. These ranged determines how much of the signal is lost through the environment during communication. Most communication aspects deal with a problem called attenuation. This is effect of the outside conditions causing noise. When a signal is sent we understand that is a wave propagating through a medium until it reaches a receiver. Based on the individual devices we can see a drastic effect on an exponential scale of the attenuation of any signal. Considering the fact that our communications are designed for use anywhere we must be able to determine the strength of our signal. This will enable us to accurately scale how much the environment at the time is affecting the signal.

Development Board Bit Error Rate

Values of yy	Bit Error Rate (in percent)
0	Less than 0.2%
1	0.2% to 0.4%
2	0.4% to 0.8%
3	0.8% to 1.6%
4	1.6% to 3.2%
5	3.2% to 6.4%
6	6.4% to 12.8%
7	More than 12.8%
99	Not known or not detectable

Table 5 – 2

Based on the development board's response we receive values of yy which determine the bit error rate in percentage (Table 5-2). This is a way for the user to scale how many bits are misinterpreted or affected by attenuation during communication. Upon use we understand that this is an important factor in communications. When we see an error rate increase we can understand that the data is being altered and undesirable for the client. By keeping this value of yy low we understand some of the bits may not all be correct but when within an acceptable range the video quality will only be effect in a minor way.

5.2.2 Complications

During testing there was a problem with the instructions given to the users. Upon arrival of the development board and chip, there were no sources of material except a

piece of paper directing the purchaser to view all documentation online at their own website. When referring to the documentation for the initial setup everything dealing with the physical connections was explained properly and concisely. The problem arose when confronting the terminal for establishing communications with the development board. As stated in the documentation “open tera term or similar terminal emulator which we proceeded to do. After visiting the hyperlink provided the terminal after proper setup was unresponsive. This inevitably lead to some troubleshooting. The first attempt was to ensure that each connection to the board was properly connected starting with the power. After reconnecting the power supply the next thing to check was the J6 jumper. Following the jumper we established a connection to the PC using USB which was verified to be connected to the PC by a bell indicating a device was properly connected. Once every connection was double checked to be properly connected we pressed the S1 power button for one to two seconds. With the development board still unresponsive we proceeded to try another terminal emulator. The alternative terminal we used instead of terminal was “PuTTY”. After switching terminals and properly checking all settings as before we found the development board was fully responsive with the chip.

5.2.3 Alternate Boards

When considering boards for this project, there were many appealing boards that would have been chosen for our project but had minor issues which inevitably allow us to eliminate them from the selection process. Some boards could handle the speeds we needed but had to many antenna ports which increased the cost of the board and also caused a size issue. In another case we had a board that was perfect for our Skywire NL-SW-LTE-TSVG chip but required an additional board to interface with the chip. This additional board can cost upwards of $165. With a strict budget that board was quickly eliminated from the selection process.

Skywire Cellular Modem Schematic

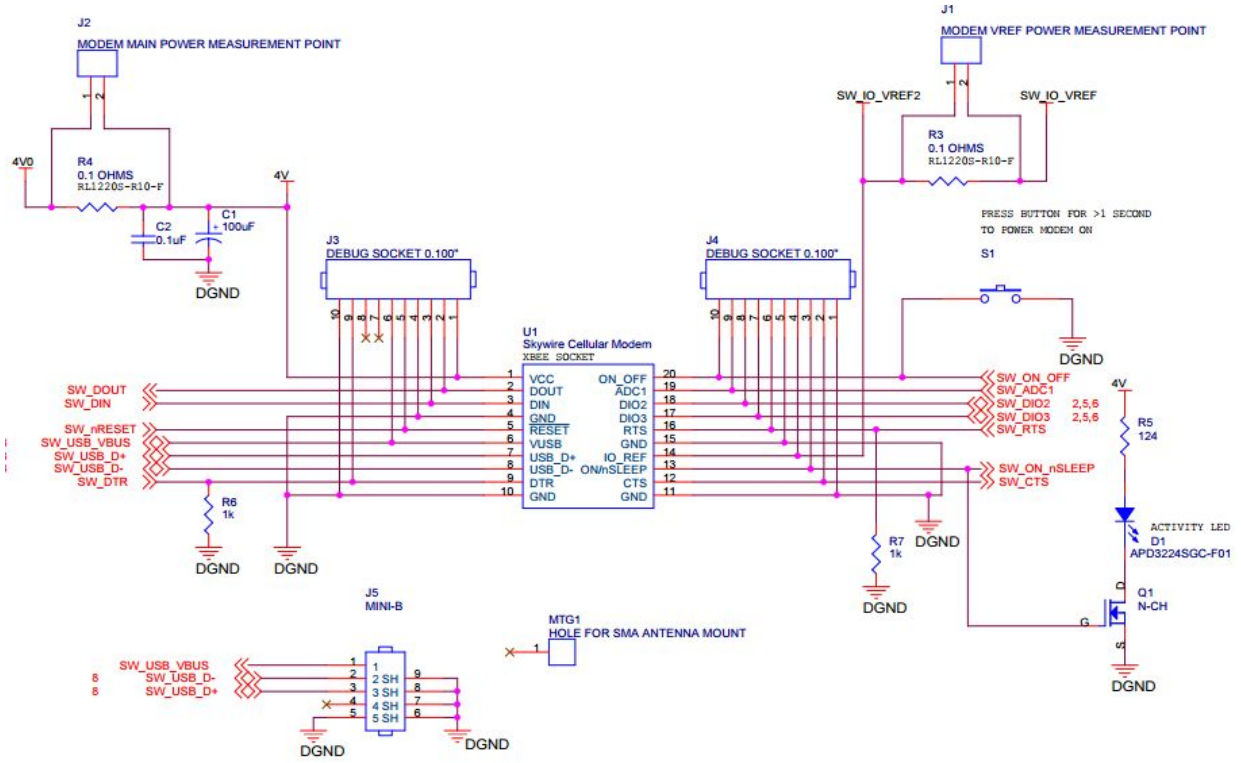


Diagram 5 – 2

After purchasing the board we will ensure that the device can perform all the necessary functions. All documentation was located on the Nimbelink website [27]. When searching for the Skywire NL-SW-LTE-TSVG chip below the product information is a pdf file to the schematics [32]. Here is where we will take a look at each individual part and its function and its potential to make it towards the project final design (Diagram 5-2).

Initially we can see capacitors in parallel with a resistor in series connected to ground. This circuits purpose is to allow a steady current from the voltage source to the capacitors in parallel (Diagram 5-3). These two capacitors are both connected to ground and in turn charge when the voltage supply is on. An important factor of placing these two capacitors in parallel creates and advantageous amount of charge. Based on these two capacitors we can see a reduction in noise on the input line.

Power Regulator

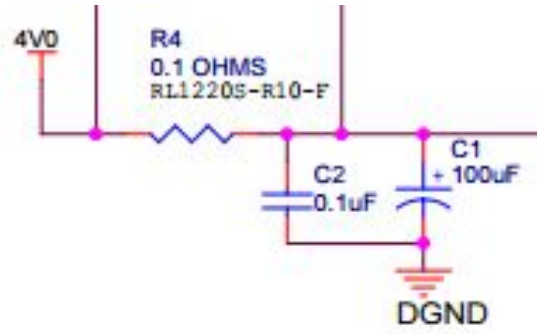


Diagram 5 – 3

Next we take a look at the actual chip itself (Diagram 5-4). Here we can see each individual pin. At this level we can now understand the connections necessary for the function of our project. At this stage we can begin to remove sections of the board that will not be crucial to the end product.

Chip Pins

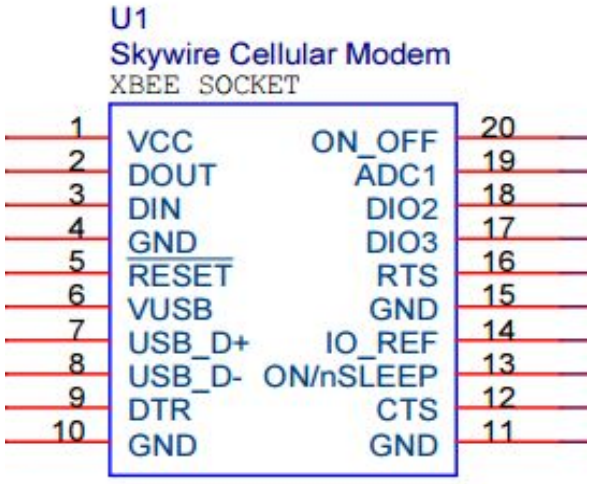


Diagram 5 – 4

Next we can take a look at an important connection for the project. Here we see the port for the USB connection (Diagram 5-5). This is where the device will communicate to a PC for setup and testing. When competition of testing is completed we will use this connection to allow the PCB board to connect to the Raspberry Pi 3.

USB Connection

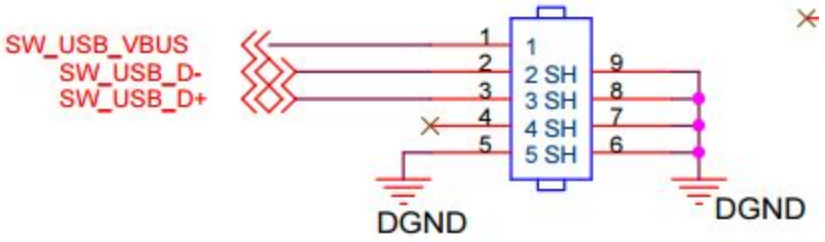


Diagram 5 – 5

Another very critical part of this project is the antenna (Diagram 5-6). The placement of this piece is desirable on the outer parts of the PCB board as not cause interference with the internal circuit or chip.

Antenna Port

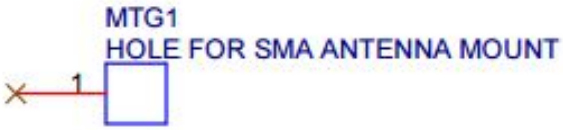


Diagram 5 – 6

Inevitably we need to include a way to power on the device. A simple push button switch is necessary and will be required to be pressed for one second before activation (Diagram 5-7). When powering on the device a period of software communication is necessary. Please allow for a period of possibly greater than 15 seconds of wait time before proper interaction between your device and the terminal on your PC.

Power Button (Switch 1)

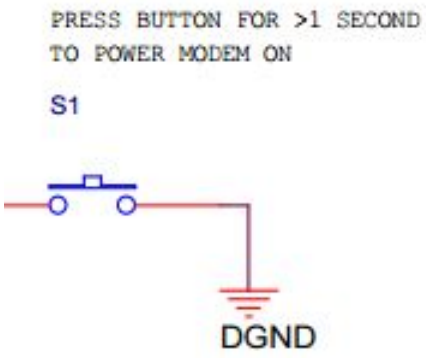


Diagram 5 – 7

Next another part that we will take a look at is the Non-Inverting Octal Buffer/Line Drivers with 3-State Outputs (Diagram 5-8). This device is perfect for power consumption solely on is ability to have two alternate active low output enables. This is advanced CMOS technology to provide an octal buffer / line driver. This chip is also rated for in the extended industrial and military grade level of operation at a range between -55C to +125C

Octal Buffer

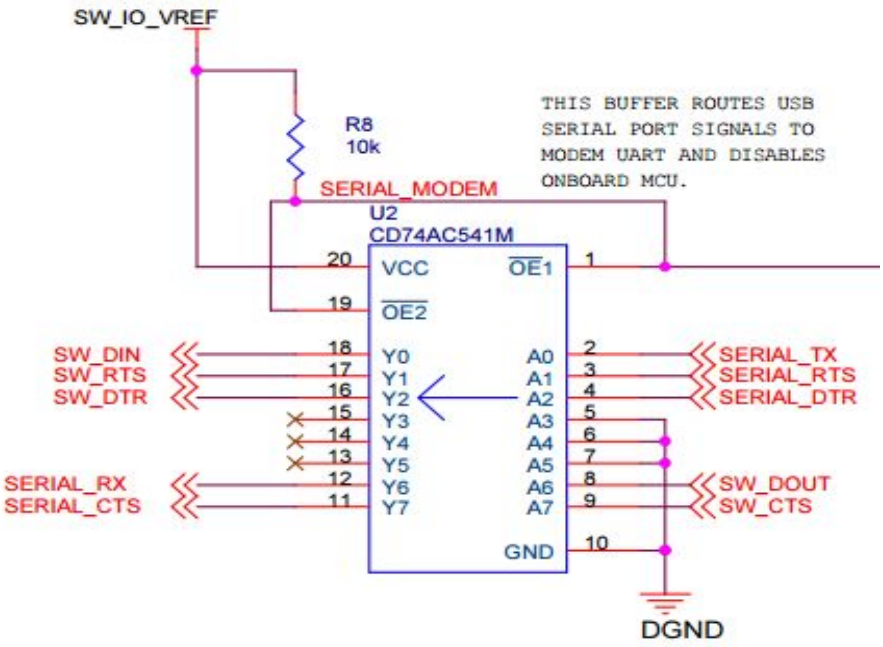


Diagram 5 – 8

We must next consider an extremely important part of our project. This part of the project is dealing with the USB to UART interface (Diagram 5-9). Considering most of our connections for the final device revolve around using USB we will be including this in our final project.

USB to UART Integrated Circuit

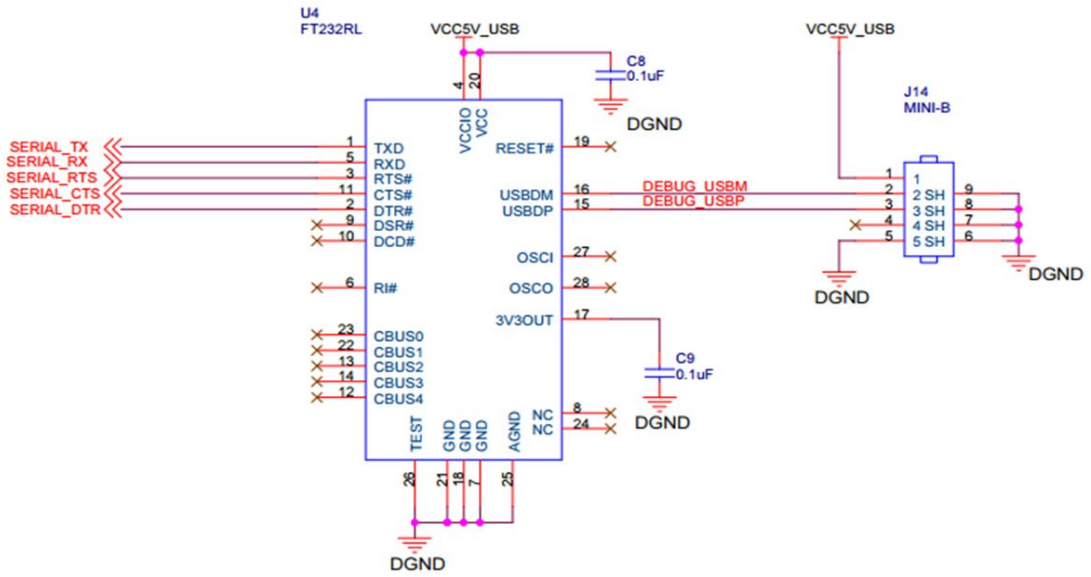


Diagram 5 – 9

5.2.4 Possible Unnecessary Parts

When first observing the board schematics we see Arduino shield connectors (Diagram 5-10). Based on this project we won't necessarily need this part because we will be using the Raspberry Pi 3. Arduino boards were considered an alternate way to take this project. The only issue with designing a project with an Arduino would have provided difficult considering we would have to remove the Arduino each time it needed maintenance or to be re-flashed with the updated programs.

Arduino Shield

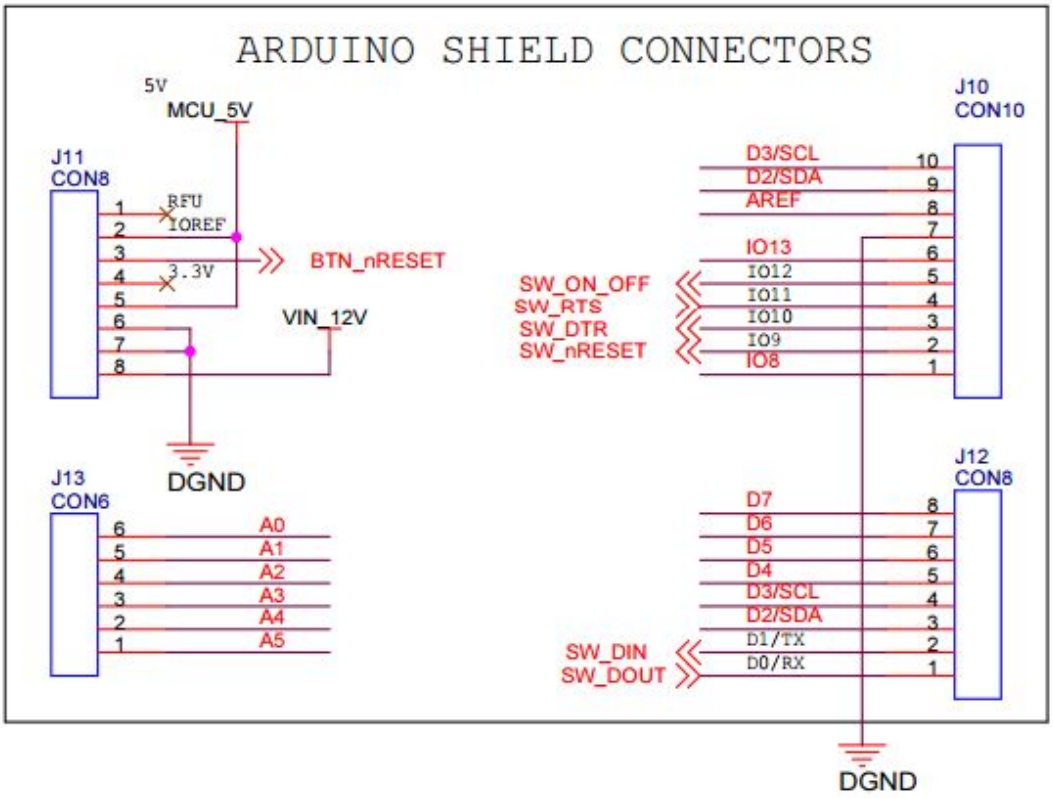


Diagram 5 – 10

The next part is a socket placement (Diagram 5-11). Still uncertain of this devices function we will later determine if this piece will be required in the final schematic.

Rectangular Cable Connection

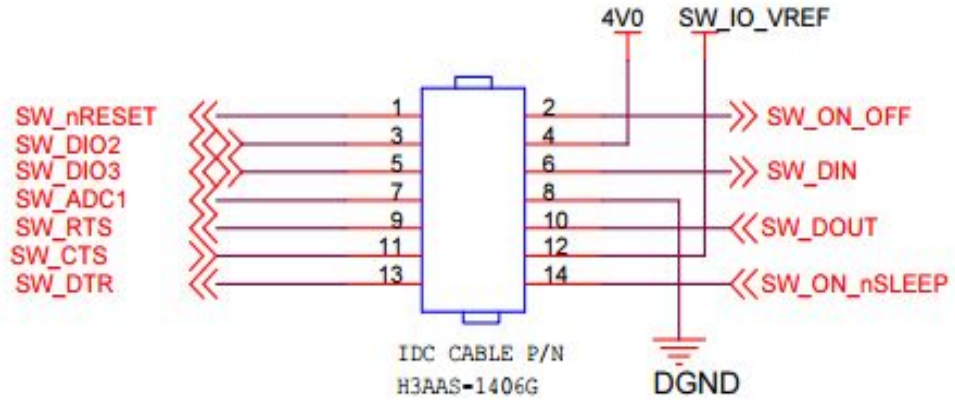


Diagram 5 – 11

5.3 Camera Output Converter Subsystem

Since the Raspberry Pi 3 does not have a HDMI input to receive the video stream from the camera, a separate device is needed in order to allow the Raspberry Pi to receive the stream. This is where the Rongyuxuan HDMI to USB 3.0 1080P Video and Audio Capture Device comes in. This device will convert the raw HDMI stream into a USB interface so that the Raspberry Pi can receive it.

We have found various other projects online that consist of using a Raspberry Pi 3 to receive a streaming input using one of the capture devices and processing the video afterwards. There are numerous different to accomplish this, but it seems the most feasible way to capture the streaming video input is to use VideoLAN's VLC player [6]. The program has a feature to capture a input stream and save it in any one of the various supported formats. Our system will use scripts and another Java or Python program in order to control this functionality remotely.

5.4 Raspberry Pi Subsystem

The Raspberry Pi 3 Model B [4] is the centerpiece of our entire project. It was chosen due to its user-friendly graphic user interface, and its load of features. The Raspberry Pi comes with Wi-Fi and Bluetooth capabilities, along with a HDMI port, 4 USB ports, and an Ethernet port. It also comes preloaded with Python, C, C++, Java, Scratch, and Ruby programming languages, giving us plenty of options with how we want to develop the system. Since it runs on a Linux-based Debian operating system, bash scripting is also extremely useful. These scripts are used for various purposes, including moving files around and organizing them, as well as sending video clips to the server along with other purposes. The specifications of as Raspberry Pi 3 Model B are as follows:

SoC: Broadcom BCM2837
CPU: 4× ARM Cortex-A53, 1.2GHz
GPU: Broadcom VideoCore IV
RAM: 1GB LPDDR2 (900 MHz)
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: micro SD
GPIO: 40-pin header, populated
Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

The Raspberry Pi's storage is a micro SD card and its operating system runs entirely off of that. SD cards aren't quite as fast as flash storage, but they are extremely

cost and space efficient. They are faster than a typical hard drive and have plenty of speed for purposes of this project. The system will be set up with a main program that communicates with a mobile application that is running on a the camera operator's phone. The mobile application will tell the program when to start recording the input video stream. Our program will then control the MJPG-streamer software to stream the video into the Raspberry Pi.

MJPG-Streamer [7] is a command-line tool that takes a stream and converts it to JPEG files, which then also have the ability of being streamed over an IP-based network. It relies on an input plugin that copies the images to an accessible memory location and an output plugin that processes the images that can save a single JPEG file. This video input can be streamed in at various resolutions and frame rates. When the user chooses to end the recording, this stream will cut off and the JPEG file will be saved. For our project this file will need to be at least 1280x720 and have a minimum frame rate of 20 frames per second.

Now that the recording is over, a script will be run that moves the new file to a specific folder. After that, the video will need to be converted to the H265 format so that it takes up the smallest amount of room as possible on the server without any loss in quality. This conversion process will be done completely on the Raspberry Pi. VideoLAN provides the source code and scripts for the H265 compression format to assist users in converting other formats to H265. This software is quite processor-intensive however, so this has the potential to cause some momentary slowdown on the Raspberry Pi. Installing the codecs and conversion software on the Raspberry Pi was a tedious task. FFMPEG software [8] along with various libraries were needed for the process. These downloads alone took hours, and the problems didn't end there. In order to compile the conversion software after the download, a C++ GCC compiler was used. However, the one that was included on the Raspberry Pi continued to run into errors and it became apparent that a reinstall/upgrade was needed. After downloading and installing version 6.1 of the GCC compiler, the conversion software was finally able to be installed.

Moving on, immediately after the video is moved into a specific folder, a script will be run to convert that video into the H265 format. The next part of each video clip's lifecycle is a queue, which is set up to upload all the video files in this specific folder that contains all of the current video clips. Once the upload is confirmed the video files will be deleted to make room for the rest of the clips. Diagram 5 – 12 outlines this process.

Raspberry Pi Logic Flow Diagram

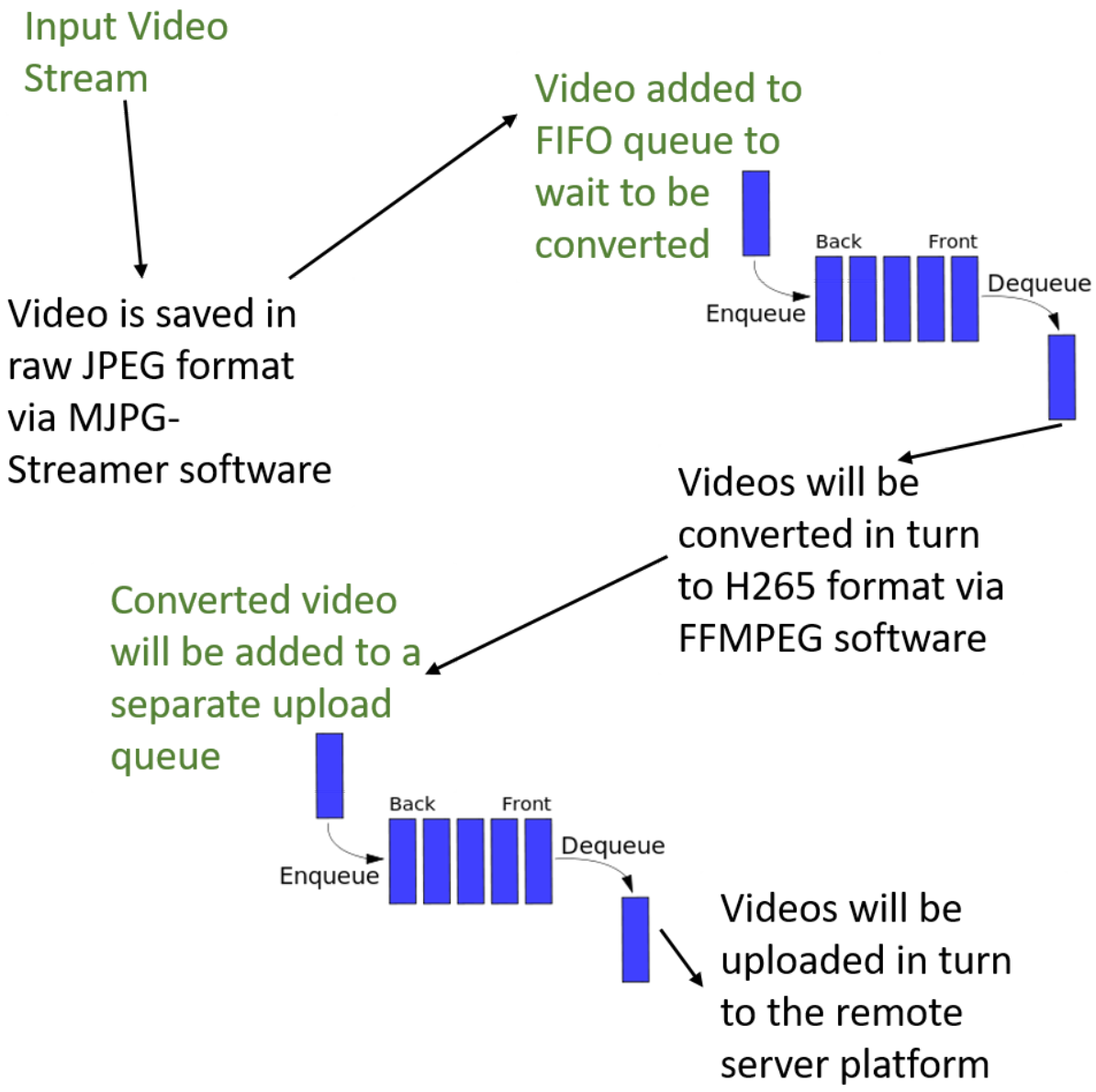


Diagram 5 – 12

5.5 Server Integration/Mobile Application Subsystem

Most of the server's operation is outside the scope of this project, however the video will need to be organized once it ends up on the server. This will done by the sport being played, the date the game occurred, the division of the teams, and the team names. There will be games that have many clips just for a single game, especially for

football. This clips will need to be numbered in order so that the game sequence can be easily followed once all of the clips are located on the server. The server will need the ability to communicate back and forth with the mobile application so that the user can specify how they would like the organization to take place.

There are many possible ways for the mobile application to communicate with the Raspberry Pi including Bluetooth, Wi-Fi Direct, SSH and VNC. SSH provides the ability to access the Raspberry Pi's terminal and file system through an IP address, username and password. This can be extremely useful for a multitude of purposes. The downside of using SSH is that you have to be connected to the same Wi-Fi network as the Raspberry Pi. This would prove to be extremely problematic since most sports venues do not have accessible Wi-Fi networks, not to mention the IP address of the Raspberry Pi can change across networks. VNC (Virtual Network Computing) is another option of note. VNC allows a user to control the graphical user interface of the Raspberry Pi from a remote device. This can be extremely useful for debugging during field tests as it allows for complete utilization of the Raspberry Pi. However, this would not be useful for the end user of our project, since they would have no need to have access to the entire GUI of the system. This feature is mostly used for servers, but it will be enabled on our end devices. Wi-Fi Direct is another option to have two Wi-Fi-enabled devices communicates with each other. This communication is done through a P2P shared network that is closed off to other devices. Both devices and can/receive data as well as commands through this interface. However, this connection is very hard to achieve using a Raspberry Pi. Extra hardware is needed and even after this hardware is obtained; it is an extremely tedious process to get it operating correctly. And this doesn't even include the process of implementing it into a mobile application, which would likely be almost as difficult. This leaves us to the last communication platform that we have considered: Bluetooth.

Luckily, the Raspberry Pi 3 device that we are using for this project comes with functioning Bluetooth. Bluetooth provides a somewhat fast platform for sending and receiving data and commands from one device to another. The obstacle for this project involves getting it to communicate with a mobile phone correctly, since this does not natively occur for mobile devices. There are "Bluetooth Terminal" applications that exist for both Android and iPhone. These applications allow for access to the file system of the Raspberry Pi. This is the exact functionality that is needed for our project, but it needs to be automated to make the interface more user-friendly. Bluetooth connections will need to be seamless using our mobile application, and scripts/programs will need to be started from the mobile application through the Bluetooth interface. Below, Table 5-3 lists the benefits and drawbacks of each communication interface that is plausible for use with a Raspberry Pi in our situation:

Communication Types Compared

	Benefits	Drawbacks
Bluetooth [9]	Seamless and reliable data transfer	Setup can be a pain between mobile device and Raspberry Pi
SSH [10]	Provides access to Raspberry Pi's complete file system, easy to set up	Raspberry Pi and Mobile device must be on the same Wi-Fi network
VNC [11]	Provides a complete remote GUI to control all aspects of the Raspberry Pi	Entire GUI is not needed for this project and it would unnecessarily use a large amount of battery on the mobile device
Wi-Fi Direct	Can provide a fast data connection between two devices	Not feasible in our situation as the setup requires extensive trial-and-error programming for a result that isn't guaranteed

Table 5 – 3

A mobile application is also needed for the project in order for the user to be able to control the QwikBox. The application will give the user the ability to start and stop the recording of clips, as well as apply team names, a sport, and various other attributes to a set of clips that make up an entire sporting event. These attributes will help with the organization of video clips once they are uploaded to the server.

The mobile application will initially be developed for the Android operating system using Android Studio and the Android SDK. Once the development has finished for the Android application and its functionality is tested to assure it is functioning properly, development will begin for the iOS application. A team member has already created a stats-keeping application on both iOS and Android for the QwikCut business. The sponsor has requested that the additional functionality be added to the existing application so that users will not have to manage multiple mobile applications at one time. However, the stat-keeping functionality of the application will have no effect on this project. Below in Diagram 5-13 the flow of the application is outlined.

User Logic Flow Diagram

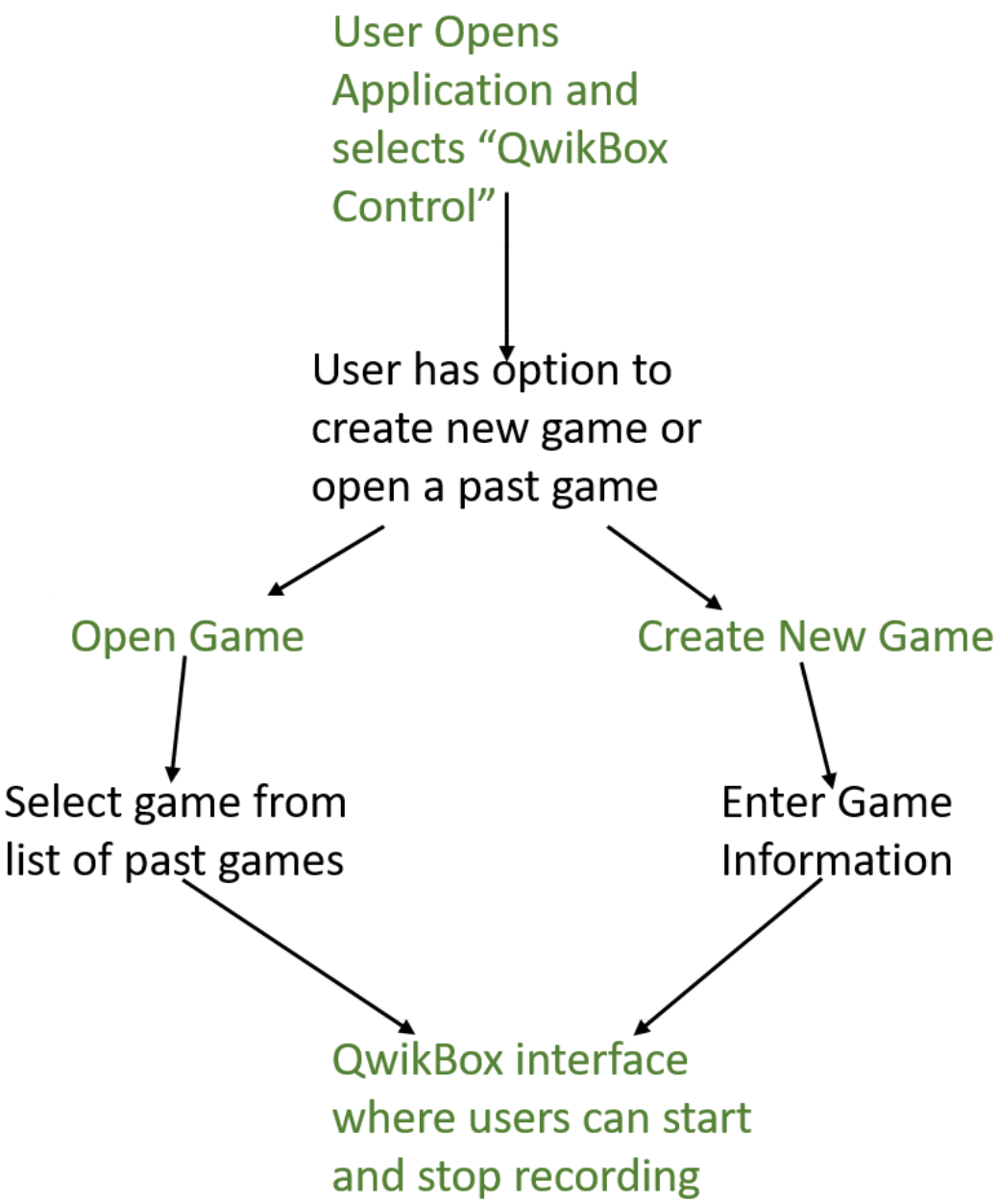


Diagram 5 – 13

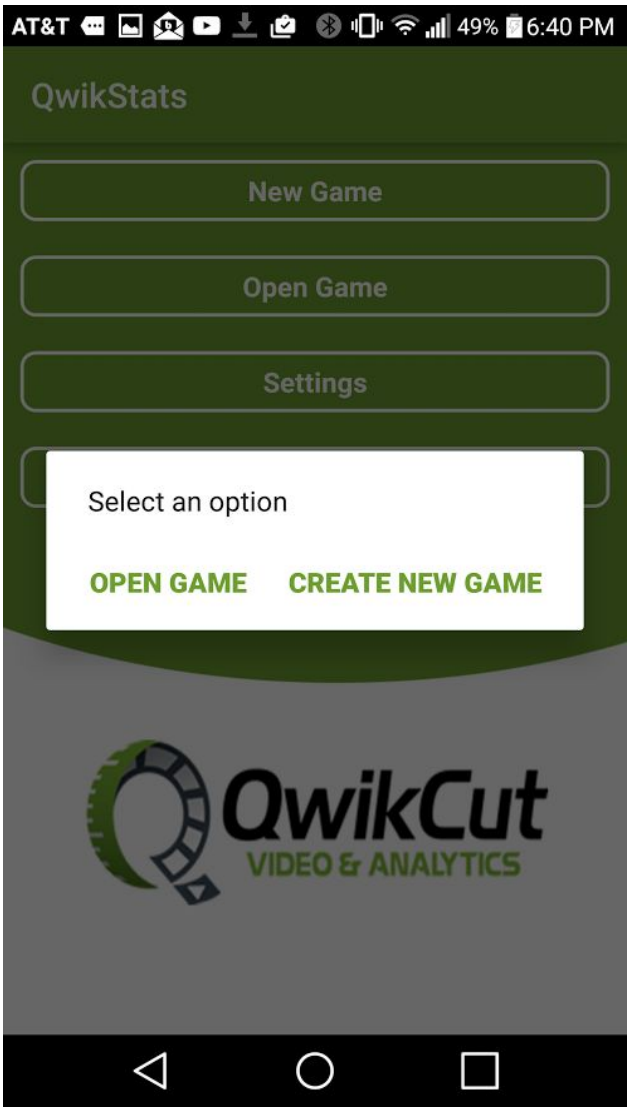
The application opens up to the main screen, which has options for the application's other features that have to do with stat-keeping and exporting. It also has the option for "QwikBox Control", which is the portion of the app we'll be dealing with in this project. It provides all functionality needed with the QwikBox. It appears in Picture 5–4 below.

QwikStats Menu Screen

Picture 5 – 4

The user will then be prompted with a dialog as seen in Picture 5-5 asking if they would like to create a new game, or if they would like to open an existing game that has already had video uploaded for it. If the user selects to create a new game, a new dialog will appear with that functionality. This is the same case if the user selects to open a game, unless there are no previous games on the device. If there are no previous games, a message will be displayed that indicates so.

QwikStats Game File Options Screen

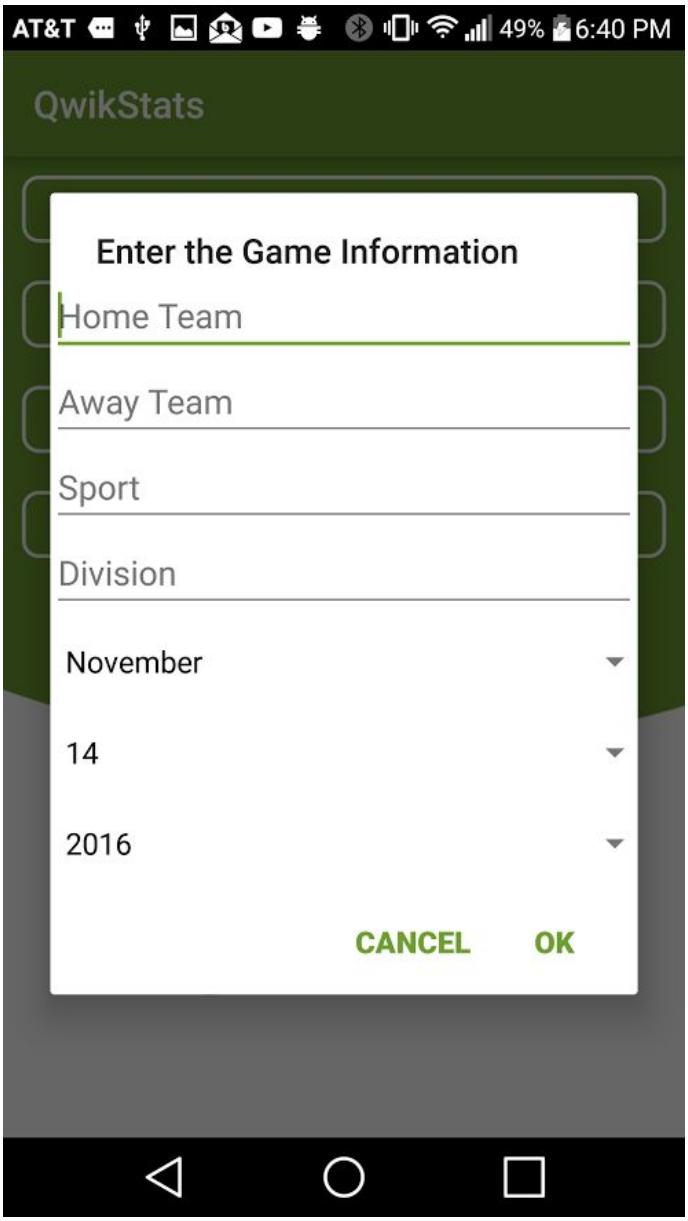


Picture 5 - 5

If the user selects to create a new game, the user will input the sport being played, the team names (specifying the home team), the division, and the date the

game was played (Picture 5-6). The three date fields will automatically be filled out based on the current date. All three date fields are dropdown menus. The other four fields are text input fields. All of this information will be saved in a simple info file on the device that will be included with the video clips when they are uploaded to the server.

Upload Game Screen



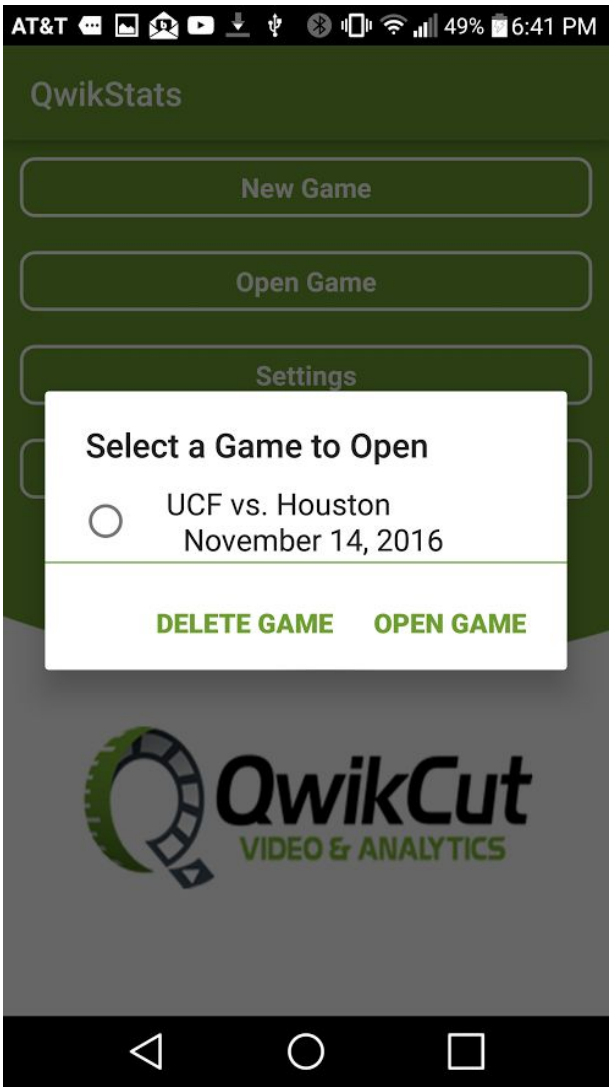
Picture 5 – 6

If the user selects to open an existing game, they will have the opportunity to select from one of the games that have previously been created on their device, or

delete an existing game (Picture 5-7). This is to save users from having to input the same data multiple times in the case that they exit out of the app. A new game is created and saved immediately after the original info is entered.

This popup layout was designed using a completely different layout file. The drop down menus are developed using the Android feature called a spinner. The possible options are populated using Java ArrayLists and built-in methods are used to grab the selected item. For the month drop-down menu, the selected string is fed through a simple method to produce a number for the month. This is then written to the game information file once the game is created. If a user selects to delete a game, they are prompted with a confirmation message.

Open Game Screen



Picture 5 – 7

Next is the game screen. This screen will include a button to start the recording of a clip, and to stop it (Picture 5-8). It will also include the game information and an indicator to confirm that the device is indeed connected to the QwikBox via Bluetooth. This Bluetooth connection is imperative so that the user can control the QwikBox with as little lag as possible.

Record Game Screen

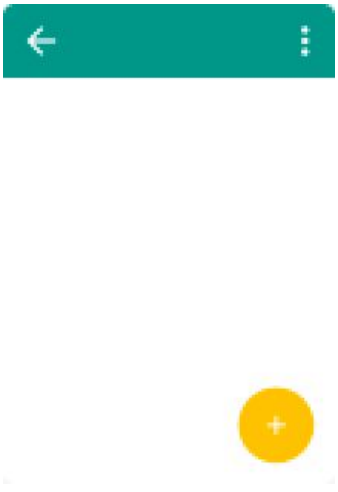
Picture 5 – 8

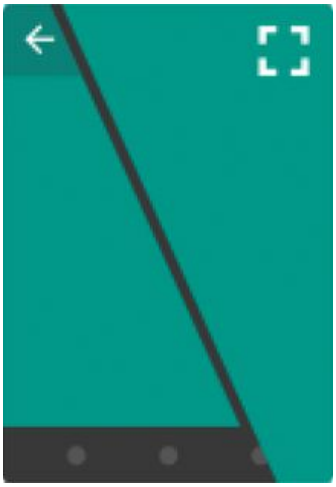
On this screen the center button's image will also change depending on whether the QwikBox is currently recording video or not. This will give the user a good indication of the current state of the recording process.

The design for the mobile application will be practically identical on the iOS platform. This application will be developed using Apple's XCode IDE using the Swift programming language. This is a somewhat foreign language to the developers on this project, so there will definitely be a learning curve. The development process for the iOS app will begin once the Android version has been confirmed to be working properly and communicating with the Raspberry Pi. Like the Android version, an existing iOS app that was developed by a team member already exists. This will assist in development as a new application will not need to be created from scratch.

Android development is done in the Java programming language, but it has many extra components and features that aren't used in standard Java development. The first, and arguably the most important, component of Android development is the Activity [12]. Activities are application components that provide a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which the developer can specifically design its user interface. When creating an Activity in Android Studio, you are given numerous template options for the type of Activity that you want to create. Table 5-4 discusses the different Activity types of Android and the way they are viewed to the user.

Android Studio Activity Table

Activity Type	Image	Description
Basic Activity		This offers a fairly simple app layout with a app/navigation bar at the top along with a customizable floating action button in the bottom right. This is a good starting point for many applications since these are commonly used UI elements

Empty Activity		This template allows the developer to start from scratch when designing the activity's user interface. It provides a single layout file in which the user can add any UI components they desire. This is what we went with for this project for the main screen of the QwikBox functionality because we do not need an elaborate layout and we wanted to make the screen fully customizable.
Fullscreen Activity		This view allows the user of the application to alternate between a fullscreen view and a view with normal UI controls, with the fullscreen view as the default. This can be used for viewing pictures or media, which may be applicable to this project down the road if we were to add features to the application.
Login Activity		This template is for applications that implement a login screen. With email and password fields along with a login button, this comes with all the UI elements that are necessary for a login screen. If the sponsor wants the cameramen to be able to login to the app down the road, we will need to implement this login activity template into the application.

Master/Detail Flow		This activity template is used more for tablets or phones in a landscape orientation. Since our application will likely need tablet support, this layout might come in handy. It creates an item list display alongside a display that shows details or options of the item selected.
Navigation Drawer Activity		This template is widely used across the Android ecosystem. It has all the components of the Basic Activity template from above, with the addition of a navigation drawer. This drawer can be opened at any time by the user and provides navigation options for the entire application. This allows the user to quickly and easily bounce between different application activities, as well as access the settings.
Scrolling Activity		This template is for creating an activity that has a lot of content to display, too much for the size of the screen. It allows the user to scroll down and up to view all of the content, as well as providing a collapsible toolbar that is visually appealing. This toolbar also contains a floating button to be used for a common action, as well as room for a header and other UI components that the developer desires.

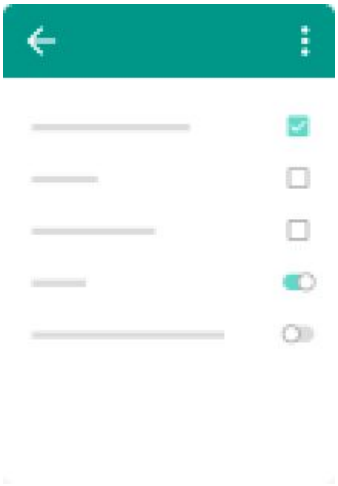
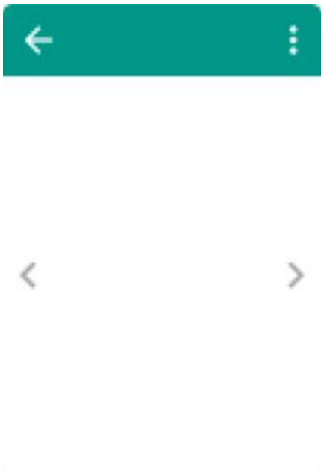
Settings Activity		This template provides a place for the user to edit all user preferences and settings. Depending on the application, these settings could be retaining to the entire app, or just a specific portion of it. There can also be multiple settings activities linked together to provide a more simplistic navigation for the user. A setting activity was implemented into the application prior to the start of this project, but additional preferences that refer the QwikBox functionality will need to be added.
Tabbed Activity		This template creates a layout for an activity that has multiple sections that can be navigated by swiping left or right. For this application, this layout could be useful when choosing between games to load, or choosing between plays to view if that functionality is included in future builds of the application.

Table 5 – 4

The templates provide the developer with a great starting point for each activity's layout, but the rest of the layout still needs to be developed. This is done by using one, or a combination, of the 7 different layout types [13] that Android provides. These layouts are written in an XML file, with a format specific to Android programming. The different layout types are outlined below in Table 5-5.

Android Studio Layout Descriptions

Layout Type	Description
Linear Layout	View group that aligns all UI elements within it in a single direction, whether that be horizontally or vertically.
Relative Layout	View group that aligns all UI elements relative to each other.

	Children can be placed on all sides of other elements, with margins or no margins. They can also be placed relative to the frame containing them.
Table Layout	This view group groups all children UI elements into rows and columns for an easily-managed layout.
Absolute Layout	This layout type allows for the developer to specify the exact location of each UI element. This is not usually considered a good layout to use due to the various screen sizes and resolutions of the devices that run the Android operating system.
Frame Layout	This layout type is used a display a single view or UI element to the user.
List View	This is used to display a list of items that the user can scroll through to see or interact with. This is used in the application for this project in many places, including when a user is selecting a past game to open.
Grid View	This allows the developer to specify a two-dimensional scrollable grid of specified dimensions. UI elements are able to be placed in any pocket of the grid.

Table 5 – 5

Inside each layout, child UI elements are declared in the XML layout file. These elements are declared by specifying a UI type, along with an ID. In addition to that, there are numerous other attributes that can be applied to an UI element. Each of these attributes specifies the location of the element within the layout. This can be relative to other elements or it could be given a specific X and Y coordinate. It's all dependent on the layout, and more than just location can be specified. Color, size and style are all common XML attributes as well. To give an example, below is the XML code for the 'QwikBox Control' button on the main screen of the application.

```
<Button
    android:id="@+id/QwikBoxBtn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:layout_marginRight="10dp"
    android:text="QwikBox Control"
    android:layout_below="@id/settingsBtn"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"/>
```

The top line specifies the button type while the second line gives the button an ID so that it can be referred to in the code. The next two lines give the UI element a width and height. The “fill_parent” designation assigns the width of the layout to the UI element, while the height “wrap_content” gives it the minimum height that still ecompasses all of the element. The margins are relative locations to the edge of the layout or to the elements around it. The text specifies the text within the button, and the layout_below attribute aligns the button below the UI element with the ID specified on that line. The gravity centers the button text within the button.

The layout of the screen may be what the user sees and interacts with, but the java code behind it provides the brains. All of this code is executed in an order based on the activity’s lifecycle. Each activity of an application goes through its own lifecycle as the user navigates through the functionality of the application [14]. Diagram 5-14 below illustrates the stages of the lifecycle for each activity. The entire lifetime happens between the onCreate() and onDestroy() methods, but the visual lifetime (the part the user sees and interacts with) happens between the onStart() and onStop() methods. When onCreate() is called, it declares all variables and UI elements and loads the activity. Afterwards, onStart() brings the UI elements to the foreground and allows for interaction. OnStop() is called when a user navigates away from the activity. The rest of the methods have to do with the specific purpose of the activity and navigation through other apps and the other parts of the current application.

Android Activity Logic Flow

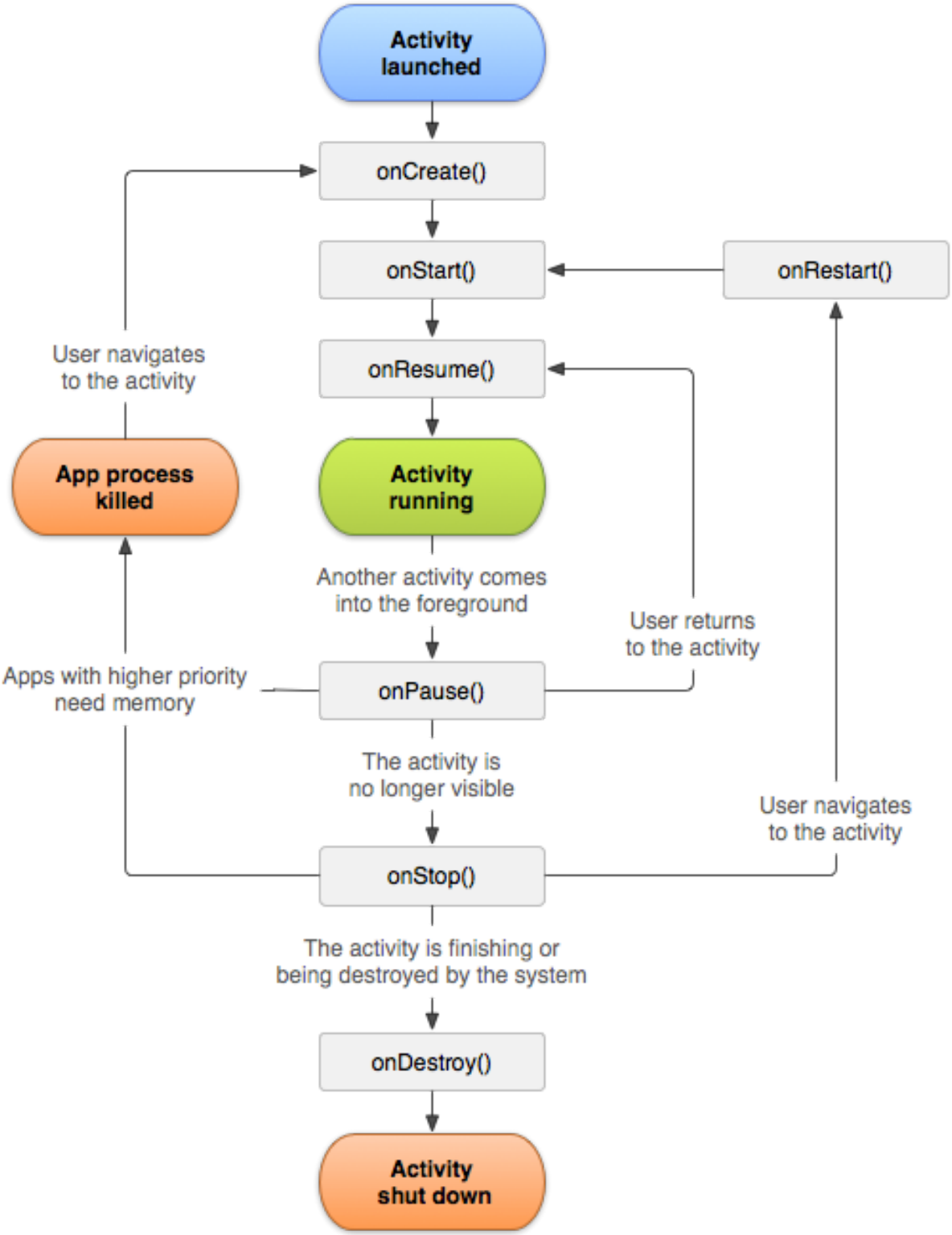


Diagram 5 – 14

The next step in the mobile application development process is designing the application for iOS users, both on iPhone and iPad. Developing for the iOS operating system requires a Macintosh computer with Xcode installed. Luckily a project group member already owned a Macintosh, so to get started only Xcode, Apple's IDE, needed to be installed.

The development process for iOS has its similarities and differences to Android. There are no activities, but instead there are View Controllers [15]. These View Controllers consist of a layout along with all of the code that allows the different UI components within the layout to interact with the application. Like an activity in Android, a View Controller has a typical flow. This is shown in Diagram 5-15 below. The old standard language for iOS programming was Objective C, but recently Apple developed their own programming language. This language is called Swift, and it is what we are using to develop in iOS for this project.

Application Screen Switching Methods

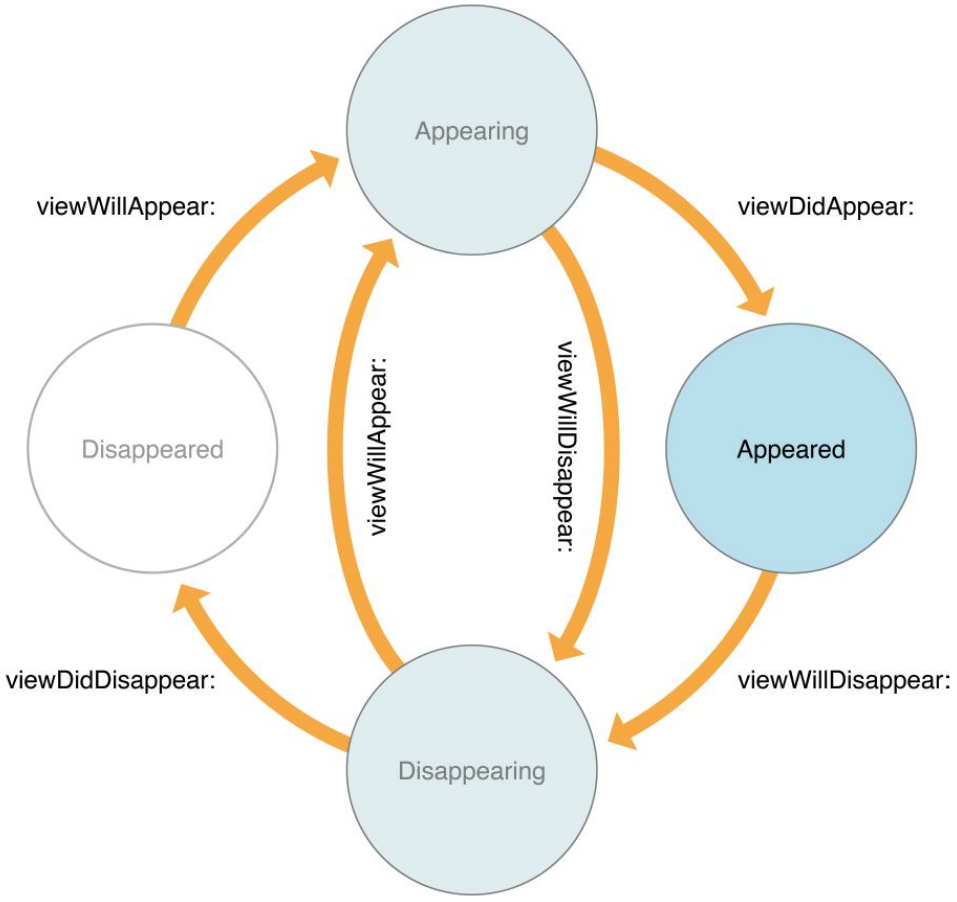


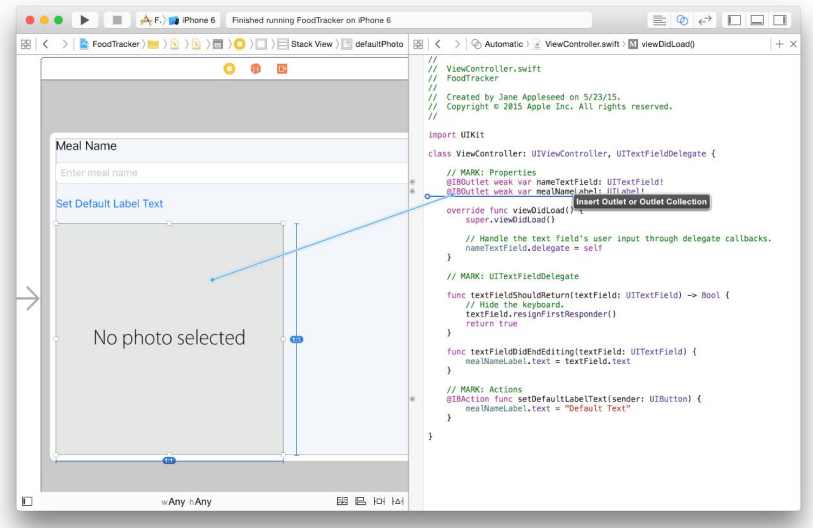
Diagram 5 – 15

This figure shows the various methods that can be implemented when developing a View Controller. The code written in these different methods will be called at different times based on what is shown in the figure. For example, viewWillAppear is for code that gets all of the UI elements and various background processes ready when viewDidAppear is for the code to be run after the View Controller is show on screen to the user.

Producing layouts in Xcode is more user friendly than it is in Android Studio. All View Controllers are shown in the project's "Storyboard" along with the different segues that show how the various View Controllers interact with each other. For each View Controller, different UI elements can be dragged and dropped wherever the developer desires. These UI elements range from labels, to buttons, to switches, wheels and whatever else is needed when developing an application. Different View Controllers can also be included in a Navigation Controller, which allows the developer to set up a sequence of View Controllers that gives the application a certain flow that the user can navigate through.

To link different UI elements to the Swift code, the layout and code can be brought up alongside each other. When alongside each other, a drag and drop method is used to link the two together (Picture 5-9). From there the developer can decide what types of actions are needed to handle by this specific UI element. Also, in order to cater to different screen sizes of iOS devices, constraints for these UI elements are needed. These constraints link different UI elements together and attach them to relative locations of the View Controller. This way of using relative locations instead of set locations allows for the view to be adjusted based on the screen size.

Xcode Linking UI Element to Code



Picture 5 – 9

After the layout is designed and all of the UI elements are mapped to the Swift [16] code, the rest of the process is rather straightforward. When mapped, each UI element is given a Swift version of an Action Listener that tells the program what to do when the user interacts with the user interface. With Swift being an object-oriented programming language, the rest of the application design can be transferred from the Java code used in the Android platform. Classes can be set up almost identically and useful methods and be translated from one language to the other. However, this translation can be rather tricky. Table 5-6 below shows the major differences between Java and Swift that could cause some issues and definitely contributes to a learning curve.

Java and Swift Comparison

	Java	Swift
Features	Endless APIs and libraries to assist the developer to accomplish just about any practical software goal. This does result in a more over-engineered Java language though.	More modern and streamlined with more compact and readable code. Apple continues to roll out useful APIs to help with all types of applications.
Familiarity	More familiar to most programmers as it based directly off and C++ and C. It is often one of the first languages programmers use when indulging in the programming world.	Swift is a new language that is based off of the less-familiar Objective C and the language is still a work in progress. Despite this, Swift is rather easy to pick up if the programmer has experience with Java or C++.
Applicability	Java is widely used around the globe for a variety of applications. It dominates the server world while also begin opportune for many different types of mobile and desktop programs. The Java programming language is huge and it's features continue to grow.	Swift was created by Apple and continues to receive additions and refinements to make it a more widely-used programming language. However, right now the vast majority of it applications are developed using XCode and meant for iOS and Mac OS devices.

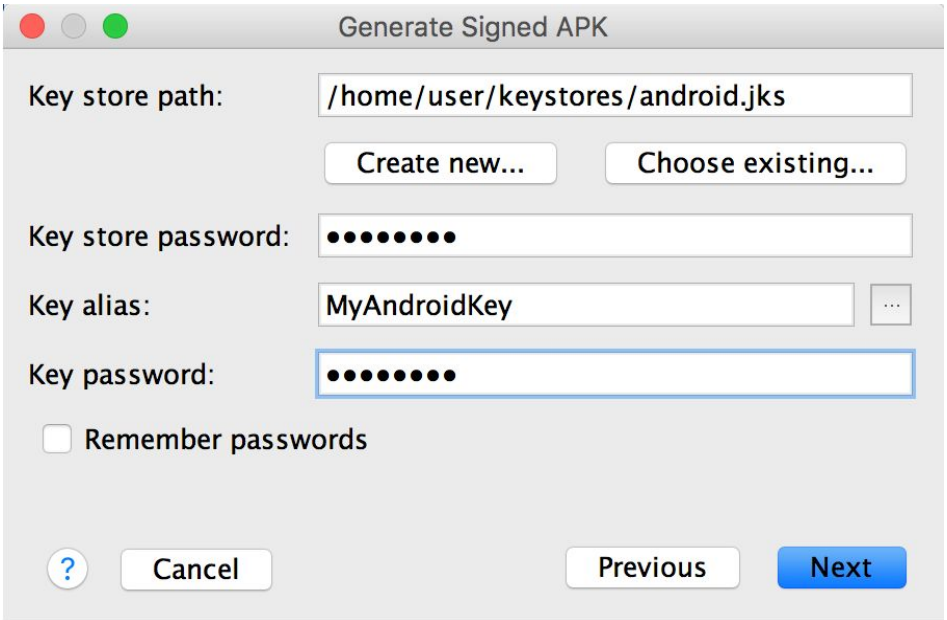
Table 5 – 6

5.6 Mobile Application Publication

Once the mobile applications for both Android iOS are developed and ready for release, they will need to be uploaded to the operating system's respective software store so that it can be easily installed by anybody who wants/needs to use it. Android's software store was created and is ran by Google. It's called the Google Play Store and currently holds over 2,400,000 apps with around 70 billion total downloads to date. Apple's App Store is also a very impressive software hub that currently holds around 2,200,000 apps with over 140 billion cumulative downloads at the time of writing. Both stores have their unique processes for submitting applications.

Starting with Android's Google Play Store, the process for submitting your finished application starts with cryptographic keys [17]. Each installed Android application must be digitally signed with a certificate that is owned by the application's developer. All of these keys are unique and only the developer holds this private key. This allows the Android system to identify the author of an application and establish a trust. If a developer has never submitted an Android application for release before, a new cryptographic key can be generated by Android Studio. The next step of the submission process also occurs in Android Studio: generating a signed APK. There is a wizard within the development environment that walks the user through the process of producing a build (Picture 5-10). Once submitted, this APK file is the file that is downloaded from the Google Play Store and finally installed on the Android device.

APK Signature Generation



Picture 5 – 10

The Google Play Developer Console is used for the remainder of the submission process. It is accessed via a web browser and is the central hub for managing all application submissions. The next step in the process is to create an app icon to be displayed in the Play Store as well as the Android device once it has been installed. This icon along with all screenshots can be uploaded to the Google Play Developer Console and assigned to your specific application submission. After that, a description and a short synopsis of your app is needed. Once all of the required information is inputted, the app can be submitted to Google for review. Applications usually get approved by Google in a very timely manner and can be on the Google Play Store by as early as the next day. Picture 5 – 11 shows the Google Play listing for our application.

QwikStats on Google Play Store

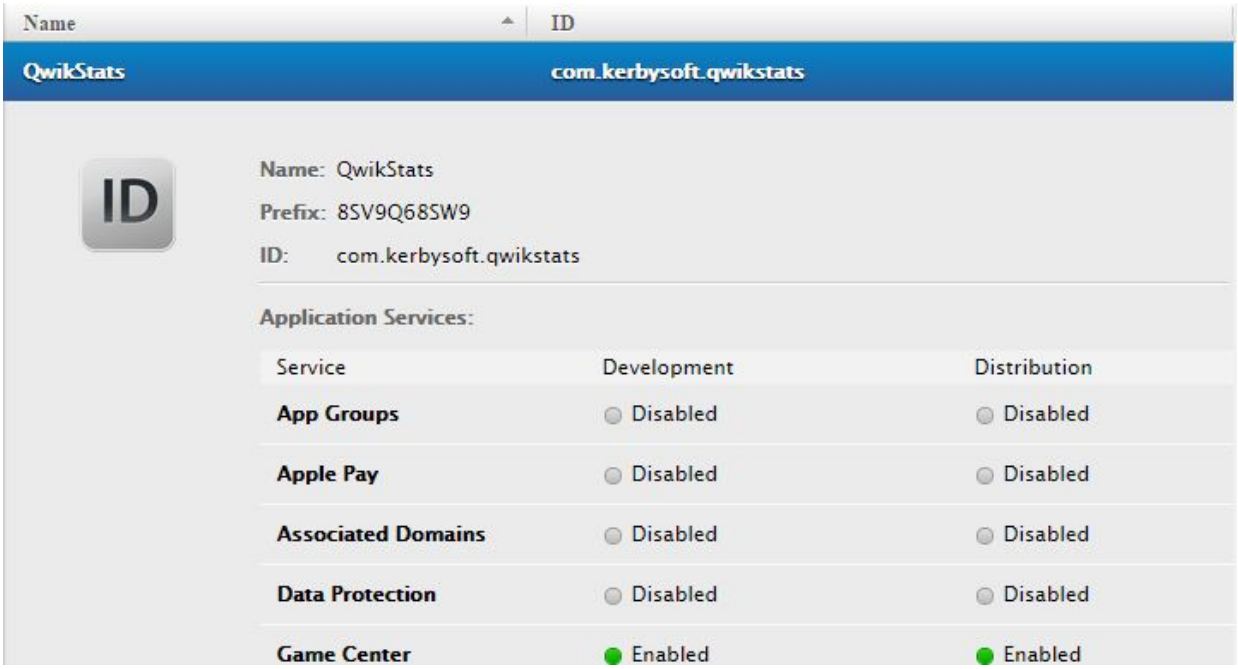
Picture 5 – 11

The process for submitting an application to Apple's App Store is a bit more complex [18]. Our first step is to purchase an Apple Developer Program account so that we have the ability to submit an application. All application information needs to be

gathered also so that it can later be used throughout the submission process. Apple's Developer Portal is also used throughout the process. This portal is very similar to the Android Developer Console as it is a place where developers manage their app submissions and information. The remainder of the process is outlined in a series of steps below.

1. Creating a Bundle Identifier. This is done in the Apple Developer Portal. It involves creating an identifier for your app bundle, which is similar to a package name in Java and is unique for each application (Picture 5-12).

Bundle Identifier Creation



Picture 5 – 12

2. Creating a certificate signing request. The purpose of these is to link your personal computer to your Apple Developer Account. This is a level of security that Apple has implemented to make sure all application submissions are genuine. This is done using a program running on Mac OS called Keychain Access. This program manages all of the keys for an Apple Developer.

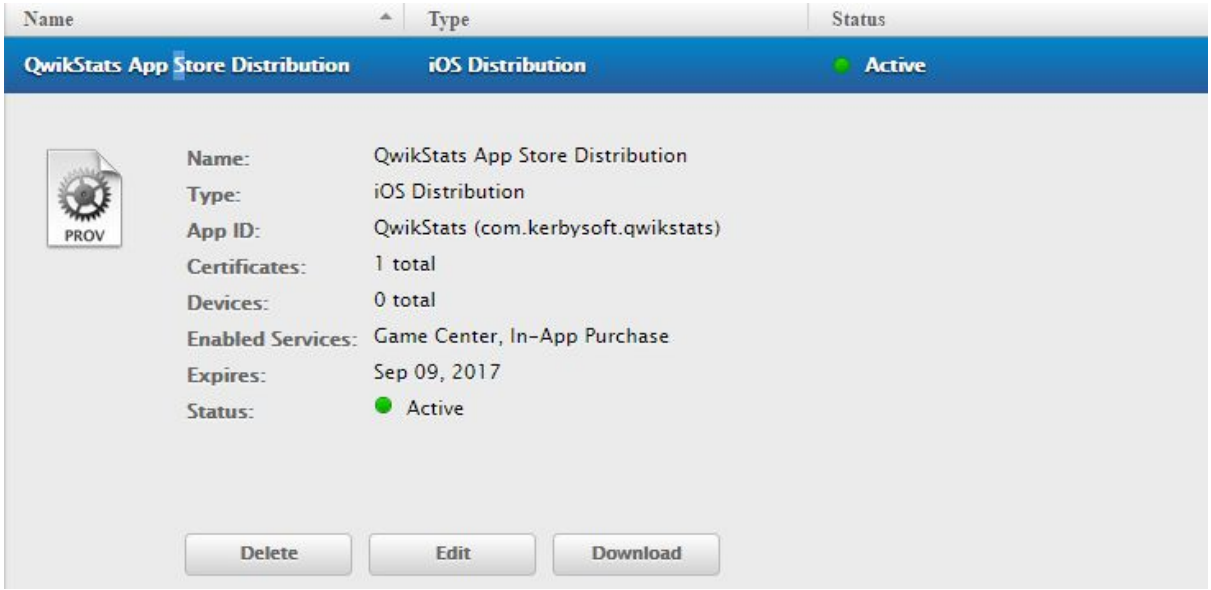
3. Creating an App Store Production Certificate. This is another step that is completed in the Apple Developer Portal. These certificates are used to link specific applications to Apple Developer accounts. It's another level of security to assure that no developer has imposters submitting applications for them. After creation, the certificate has to be downloaded to the developer's Mac OS computer and installed (Picture 5 – 13

Production Certificate Creation

Picture 5 – 13

4. Creating a production provisioning profile. Once again, this step is completed using the Apple Developer Portal. These are packaged with iOS apps so that the user's device can install them. They are bundled with a certificate created in the previous step, downloaded, and then installed on the developer's Mac OS computer. (Picture 5-14)

Provisioning Profile Creation



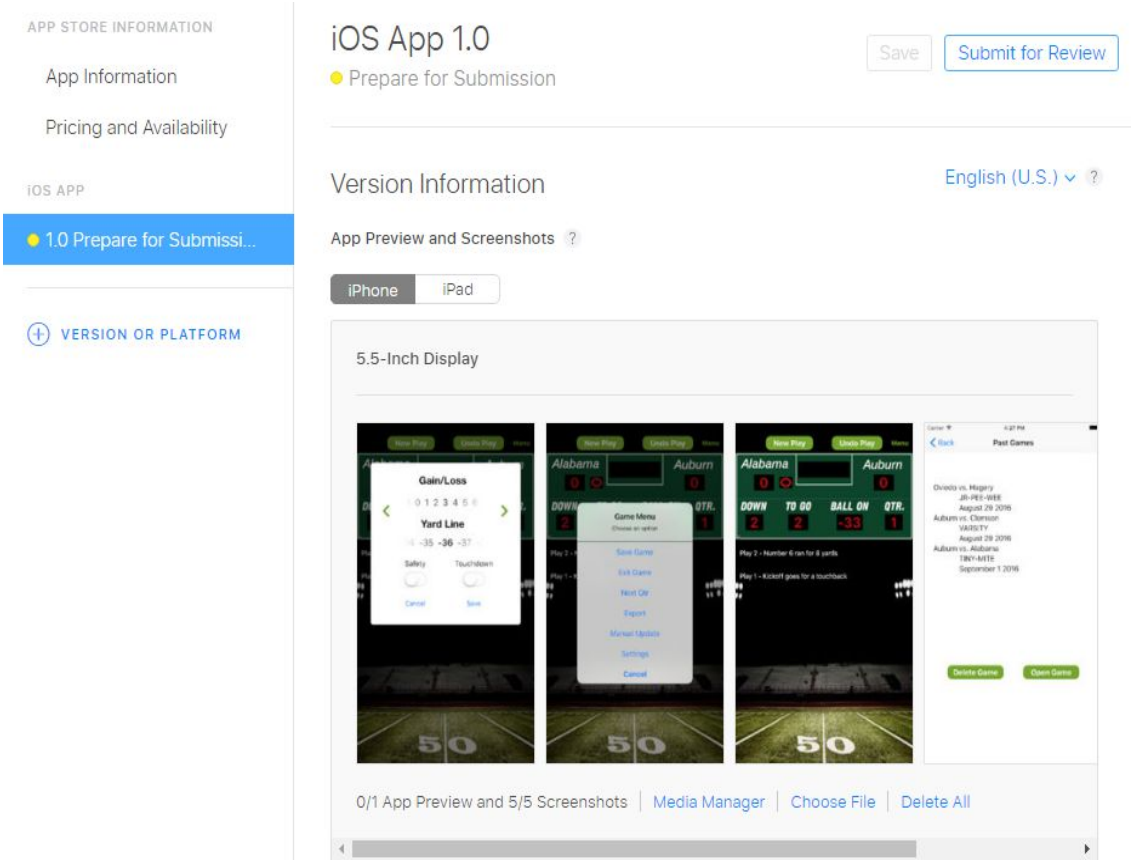
Picture 5 – 14

5. Creating an App Store listing. This is done using another Apple platform called iTunes Connect. By creating a listing, it reserves a spot in the App Store for the developer's application. This spot is reserved by using the Bundle Identifier created at the beginning of this process.

6. Making the release build. This is done on the developer's Mac OS computer in the Xcode IDE. The developer inputs the correct version and build numbers and assures that all of the correct settings are selected. By using the provisioning profile and code signing identity created earlier, the developer can generate a release build for the project. After that, the developer can sign in to their Apple Developer Account within the Xcode interface and choose to distribute the application.

7. Submit the application version for review. This is the final step of the submission process for iOS applications. Within the iTunes Connect platform, the developer selects the specific application. From there, all information is entered, and the release build is uploaded and attached. Screenshots and an application icon are also included. After all necessary information is entered, the developer can submit the app for review (Picture 5 – 15). Apple will then review the application and publish it once it's approved.

iOS Application Submission



Picture 5 – 15

5.7 Power and Battery Subsystem

We have decided to go with a design that only requires power to one components from a battery. This device is the Raspberry Pi 3, which will in turn power the rest of the components via USB interfaces. Portable uses for the Raspberry Pi needs to be powered by a good quality 5V micro USB adaptor and has a max current consumption of 1.2A (1200mA). Under normal use, the Raspberry Pi should only draw about 450 mA of current. The four USB ports benefit from a maximum supply of 2.5A (2500mA) when they're in use. However we will not need nearly the full power that the USB ports can supply. The PCB with Verizon antenna should normally only consume about 30mA of current, with an absolute max of around 700 mA under extreme stress. The HDMI to USB capture dongle should only have a max of about 0.4 mA, according to technical specifications and our calculations although it should normally operate at less than that. The HDMI port and the GPIO pins require 250 mA and 50mA respectively, but they will not be needed when the system is in use at sporting events. Therefore under normal operations our system should require no more than 900 mA of current. Certain components may receive stress at different times, but there shouldn't be a time when the system is operating that all components are maximum stress and drawing the most amount of current they can.

This is why we have decided to go with a rechargeable battery that provides 2.1 A (2100 mA) of current and 5V of voltage. The Intocircuit High Capacity Power Castle Portable Charger External Battery Pack [19] is commonly used for recharging laptops and smartphones, but it will be perfect for use in our system. The battery has a 26000mAh capacity, which is enough to fully charge about 12 smartphones. This capacity allows the system, even when drawing a near-max current of 2.0 mA, to last 9.1 hours. The battery even allows the system to be charged while in use and only costs around $65.

The custom PCB that will be designed to link the modem to the Raspberry Pi 3 has a possibility of needing a significant amount of power. If this is the case, it will need to be powered by a DC input, not just the power drawn from a USB hub. The Intocircuit High Capacity Power Castle Portable External Battery Pack has an additional DC output port that would prove to be highly beneficial in this case. This would allow for the Raspberry Pi 3 to use more of its allocated 2.1 A of current for processing activities instead of designating it to its USB ports in order to power another device. This would improve the overall reliability of the QwikBox system.

5.8 Software Design Specifications

- Develop a smartphone application to interact with a web server where the recorded video is stored. This will allow the user to edit uploaded video properties to the correct file name, date, time, and location.
- The smartphone application will also be able to tell the Raspberry Pi when to

start recording the stream, and when to stop recording the stream.

- Has to be able to convert raw video data on the fly to full HD video in the H265 format.
- Videos must be compressed and uploaded within 60 seconds.

5.9 Summary of Design

Design Designations

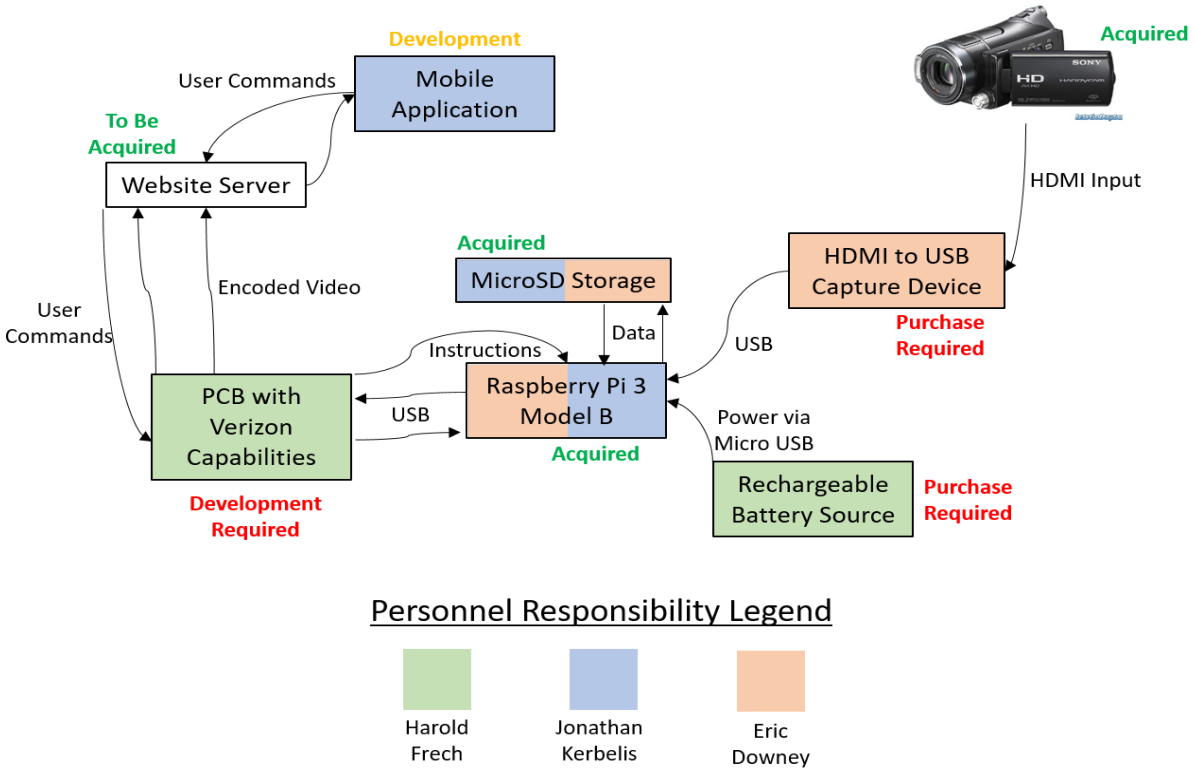


Diagram 5 – 16

Initially we were unsure what type of processor we may use, as well as the type of storage device and memory. However, deciding to go with a Raspberry Pi 3 ended up being the best option we could find. This is because it has the ability to run a Linux-based operating system that has an actual graphical interface to make programming the device easier. It also comes with an Ethernet port, multiple USB ports, an HDMI port, and WiFi all included. This has saved us time and resources since Raspberry Pis are widely available and cost as little as $40.

The next design task involved the process of retrieving video from the video camera as quickly as possible. At first, we thought the best way to do this was to pull the video file from the camera's SD card through a USB mass storage interface. After experiments with the video cameras supplied to us from our sponsor, it became clear that this was not going to be our best option. This is because the cameras don't allow

other devices to access the SD card storage while in use, and many require a power supply as well. Since waiting until the end of a sporting event does not satisfy our requirements, we had to explore other options.

There is a company called Hudl which has attempted to tackle a similar project to this one. They use an HDMI/Wi-Fi interface to project real time video to a nearby tablet or phone. The user of the this tablet or phone then presses a button to start each video clip, and presses it again to end it. The video is then converted and sent elsewhere. While our objective is slightly different once the video file is produced, this method definitely has proven to inspire our design. For our system, the video camera will be connected to the QwikBox through an HDMI cord. Inside the box, the HDMI interface will be converted to a USB interface since the Raspberry Pi 3 does not have a HDMI input. After interface conversion, the video is streamed to the Raspberry Pi with a very slight delay. Now comes the mobile application's main use, as it will have a button that the camera operator presses at the beginning of every clip and again to end the clip. This video will be recorded by the Raspberry Pi between presses, and encoded to the correct codec after it has been recorded. The encoding and conversion will be done by VideoLAN's VLC media program, or a program with similar capabilities. Once the video file is in the correct format, it will be added to a queue to be uploaded. The process will then repeat for every clip until the end of the sporting event.

The system will be powered by a Intocircuit High Capacity Power Castle Portable Charger External Battery Pack that will last approximately 9 hours under full stress. This battery was chosen because of the large capacity, USB interface, and relatively inexpensive cost. The QwikBox will also have a custom PCB that implements an antenna that connects to Verizon's 4G wireless network. It will have two sets of 10 pins with 2mm pitch female receptacles and will connect to the Raspberry Pi 3 through a USB interface. Our updated system diagram is below in Diagram 5-17, and each subsystem's designated developer is represented above in Diagram 5-16.

Final System Design

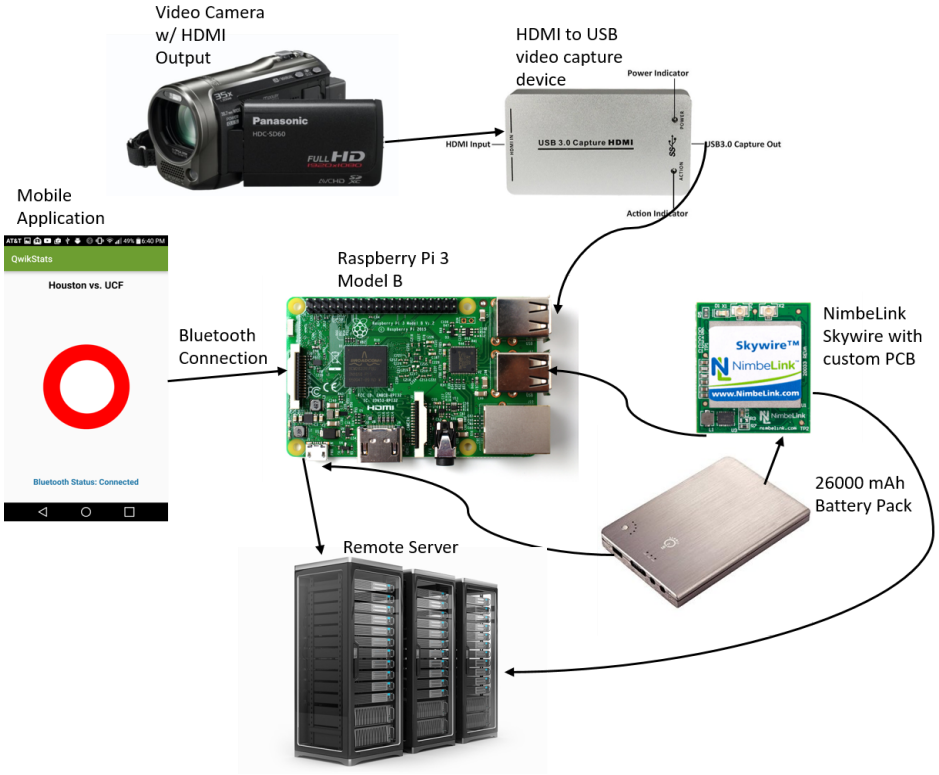


Diagram 5 – 17

6. Project Prototype Construction and Coding

6.1 Integrated Schematics

Octal Buffer / Line Drivers, 3 State

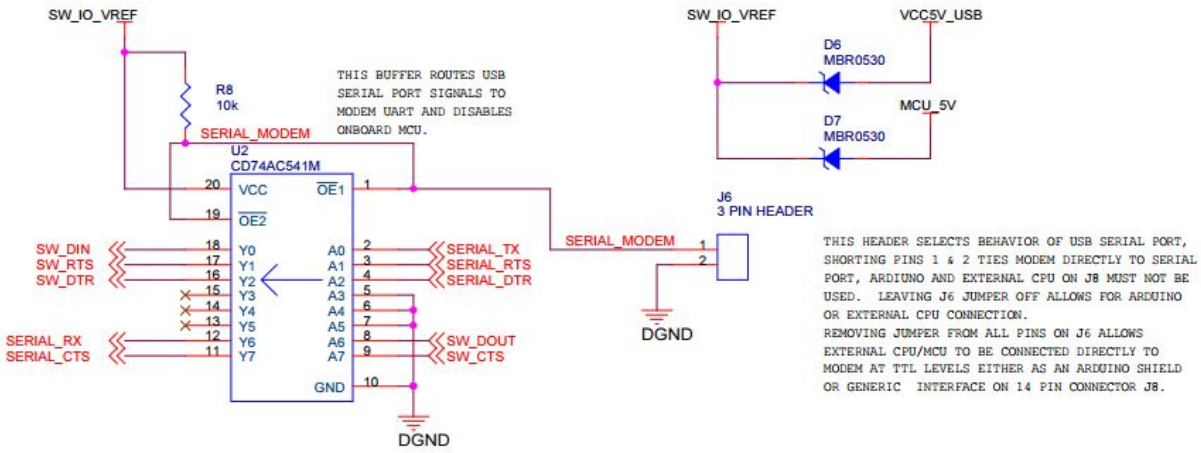


Diagram 6 – 1

When considering the Non-Inverting Octal Buffer/Line Drivers with 3-State Outputs there are two alternate integrated circuits. The first being the CD74AC541 which is included in the schematic diagrams for the development board kit. A second being the CD74AC541 which is a good secondary option if considering higher end products in the future. This IC is QML certified for military and defense applications. The Diagram 6 – 2 also shows a military grade USB to UART interface.

CD74AC541 Device

Picture 6 – 1

USB Interface

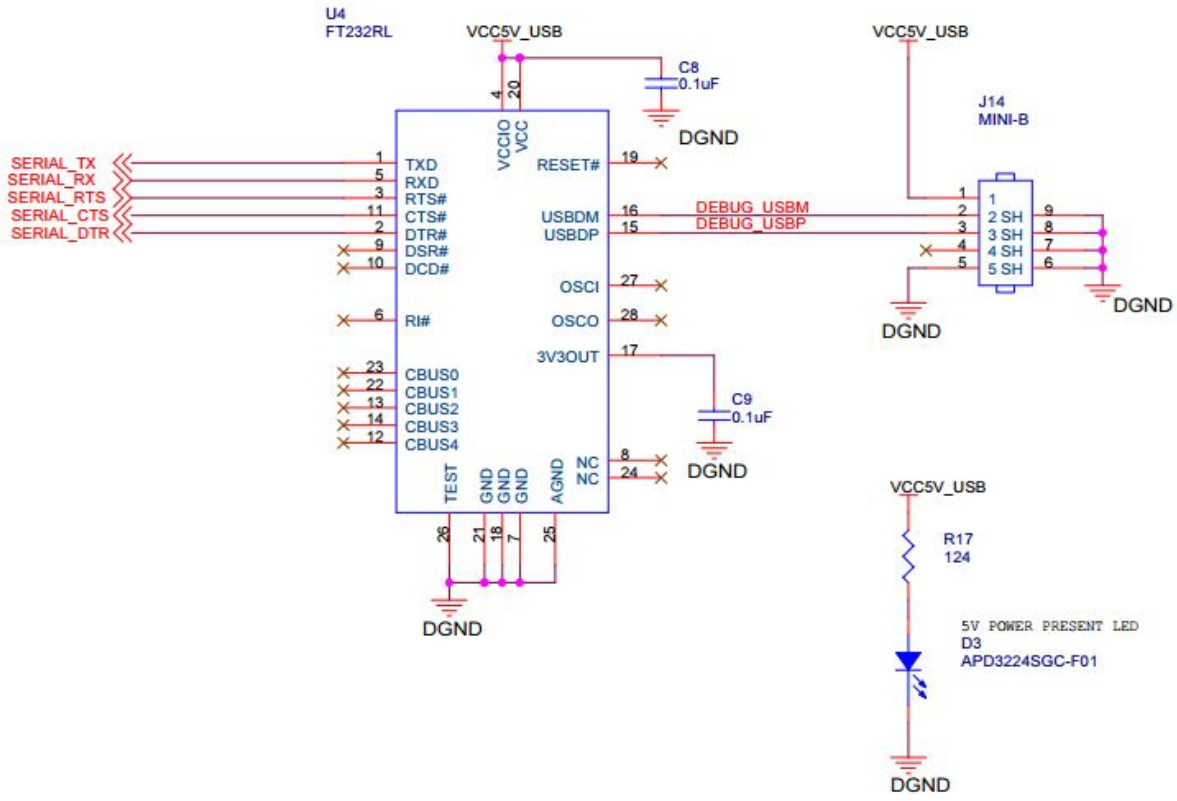
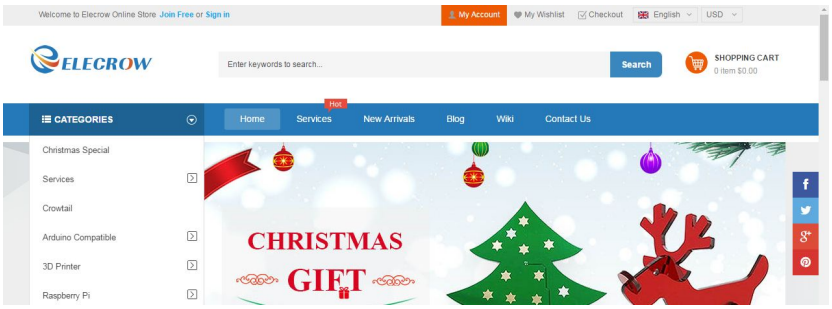


Diagram 6 – 2

6.2 PCB Vendor and Assembly

When considering a proper vendor for PCB production we must consider the project constraints with the due date and a vendor with a quick return time is desirable. The main website up for consideration is http://www.elecrow.com/.

Elecrow PCB Vendor



Picture 6 – 2

As a starting point this website allows for boards for prototyping and also completed boards after submission of proper files to a specific project manager at elecrow. A secondary website with much appeal is www.customcircuitboards.com. This site allows for as quick as 24 hour turnaround time. Based on a time crunch situation this may be the only site that can produce a custom board with good quality at a reasonable rate.

Custom Circuit Boards PCB Vendor

Picture 6 – 3

When considering a vendor the turnaround time is extremely important for due dates. Both sites provided are a clear example of reputable sites. During this step of the process we will be taking the files provided for the schematics and have to take unnecessary circuits out of the final schematic. There is no way of knowing a board design will work until you actually have the physical board in your hands and testing it for its functionality. When a problem occurs the first resort is to troubleshoot the issue at hand. Locating any shorts or pins that have been improperly soldered are easy to spot. Holding the vendor accountable for faulty production is inevitable but when depending on a deadline the failure of the board will put the entire project at a stand still. The designs for the schematics were initially intended to be looked at on the full version of Capture CIS Lite. When using the free version of the software, there is an issue considering our development board. When opening the given factory schematics on the Capture CIS Lite presents an issue considering there are too many parts for the free version to handle. This is unfortunate but will require the designer of the PCB board to use the Capture CIS Lite program and manually copy the individual parts into the Eagle.

6.3 Final Coding/App Development Plan

The mobile application will be initially developed for Android, and then it will be expanded to the iOS platform as well. Android development will be done in Android Studio using the Android SDK and other Android development tools. The app exists on the Google Play Store under the name QwikStats but the development phase for the

functionality needed in this project is just getting started. The app will need to make server calls to interact with the video files located on it in order to organize and manage the various different files for each sporting event. The application will also need to have the ability to communicate with the QwikBox so that the beginning and ending of clips can be designated by the camera operator. This will be done through Bluetooth communications since the QwikBox will be located near the camera operator due to the fact that it has to be connected to the camera via HDMI cable. The iOS development phase will be conducted on a Mac OS computer using Xcode. Xcode is the recommended integrated development environment for iOS development.

Xcode Development Environment

Picture 6 – 4

7. Project Prototype Testing Plan

7.1 Hardware Test Environment

In order to properly test the hardware functionality of the system, all components of the QwikBox will have to be set up and connected. The HDMI output will also be in use so that internal monitoring of the Raspberry Pi can be observed on a screen.

7.2 Hardware Specific Testing

The hardware testing procedure is presented in Table 7 – 1. If completed successfully, it confirms that all hardware components of the system are operating correctly.

Hardware Testing Procedure

Step #	Test Procedure	Expected Result
1	Assure the battery is charged and connect it to the Raspberry Pi. Also connect the Raspberry Pi to a display via HDMI cable	The Raspberry Pi is securely connected.
2	Connect HDMI to USB capture device is connected to the Raspberry Pi via USB	The connection is secured
3	Connect the Verizon network communications PCB to the Raspberry Pi via USB	The PCB is connected securely
4	Turn on the battery	Light are illuminated on the Raspberry Pi and the capture device. The Raspberry Pi is seen booting up on the display.
5	With the Raspberry Pi connected to a Wi-Fi network, attempt to load a web page	The web page loads completely and in a reasonable amount of time.
6	Disconnect the Raspberry Pi from any Wi-Fi network or Ethernet cable	The Raspberry Pi is disconnected from all Wi-Fi networks
7	Attempt to load a web page using only	The web page loads completely

	the Verizon network components.	and in a reasonable amount of time.
8	Once the system is in its plastic case, leave it outside for one hour in the sun.	The system is left in the sun to mimic the conditions at a sporting event.
9	After the hour has passed, repeat the previous steps.	All components receive power and operate just as they did before being left in the sun.

Table 7 – 1

7.3 Software Test Environment

In order to properly test the software functionality of the system, all components of the QwikBox will have to be set up and connected. The HDMI output will also be in use so that internal monitoring of the Raspberry Pi can be observed on a screen. Additionally, a tester with a smartphone that has mobile application installed on it will need to be connected via Bluetooth to the Raspberry Pi.

7.4 Software Specific Testing

The software testing procedure is presented in Table 7 – 2. If completed successfully, it confirms that all software components of the system are operating correctly. The team has determined that the following 14 steps will confirm that QwikBox is being developed properly.

Software Testing Procedure

Step #	Test Procedure	Expected Result
1	Connect all the components and turn on the battery. Connect the Raspberry Pi to a display via HDMI cable	The Raspberry Pi boots up and is observed on the display
2	Connect video camera to the video capture device via HDMI cable.	Connection is secured
3	Open the camera lens and check the camera's viewfinder	The camera lens is opened and an accurate image is displayed on the viewfinder
4	On the mobile application, enter some	The information is entered and

	information for a random sporting event so that the video can be organized	saved
5	On the mobile application, select the button to start the video capture from the camera	VLC player is observed being controlled by the developed program to start the camera capture.
6	Look around with the camera to different areas of the surround physical environment	The Raspberry Pi is still observed to be capturing video without and stutters or lag.
7	On the mobile application, select the button to end to the video capture for the current video clip	VLC is observed being controlled by the developed program to end the current capture.
8	On the Raspberry Pi, navigate to the expected destination folder for the video file	The video file is located in the expected directory
9	Wait for the script converting the video to H265 format to finish. Observe the format of the video clip.	This video has been converted to the H265/HEVC compression format.
10	Before the file is uploaded, open the video file for playback and watch the clip in its entirety	The video playback appears to match what was seen on the camera viewfinder during the video capture
11	After some time, confirm that the video file is deleted from the directory on the Raspberry Pi	The file is deleted
12	From a separate computer, access the destination server	The server is accessed
13	Confirm that the previously captured video clip is now located on the server with the previously entered game information	The clip is located on the server and it is organized by the game information

Table 7 – 2

8. Administrative Content

8.1 Milestone Discussion

Senior Design 1 Schedule

FALL 2016		
Description	**Duration**	**Dates**
Senior Design 1 Project Idea	1 week	August 22 - August 29
Project Discussion	1 week	August 29 - September 5
Initial Project Document	5 days	September 5 - September 9
Initial Project Document Due		September 9
Research and Design	2 weeks	September 9 - September 23
Develop Software	4 weeks	September 23 - October 28
Prototype Design	8 weeks	September 23 - November 18
Begin Rough Draft	1 week	October 28 - November 4
Table of Contents Due		November 4
Finalize Rough Draft	1 week	November 4 - November 11
Rough Draft Due		November 11
Begin Final Document	2 week	November 11 - November 25
Finalize Prototype	1 week	November 18 - November 25
Finish Final Document	1 week	November 25 - December 2
Review Final Document	5 days	December 2 - December 6
Final Document Due		December 6

Table 8 - 1

Senior Design 2 Schedule

SPRING 2017		
Description	**Duration**	**Dates**
Purchase Components	1 week	January
Build Prototype	8 weeks	January - March
Test Prototype	1 week	March
Confirm Design	1 week	March

Specifications		
Finalize Project	2 weeks	March - April
Prepare Final Document	1 week	April

Table 8 – 2

8.2 Budget and Finance Discussion

8.2.1 Budget Table

The prices listed in Table 8 - 3 are estimates from online research and quote from the manufactures. Therefore, the prices may be change in the future once the final product is implementing. Not included in this budget table is the soldering iron that was previously financed outside of the company.

Budget Table

Description	Quantity	Photo	Cost ($)
PCB containing wireless communications (4G LTE / 5G)	1		275
HDMI to USB Capture Device	1		200

Raspberry Pi 3	1		40
Battery Source	1		65
Plastic Case	1		50
Micro SD Storage Device	1		15
Total			645

Table 8 – 3

8.2.2 Budget Breakdown

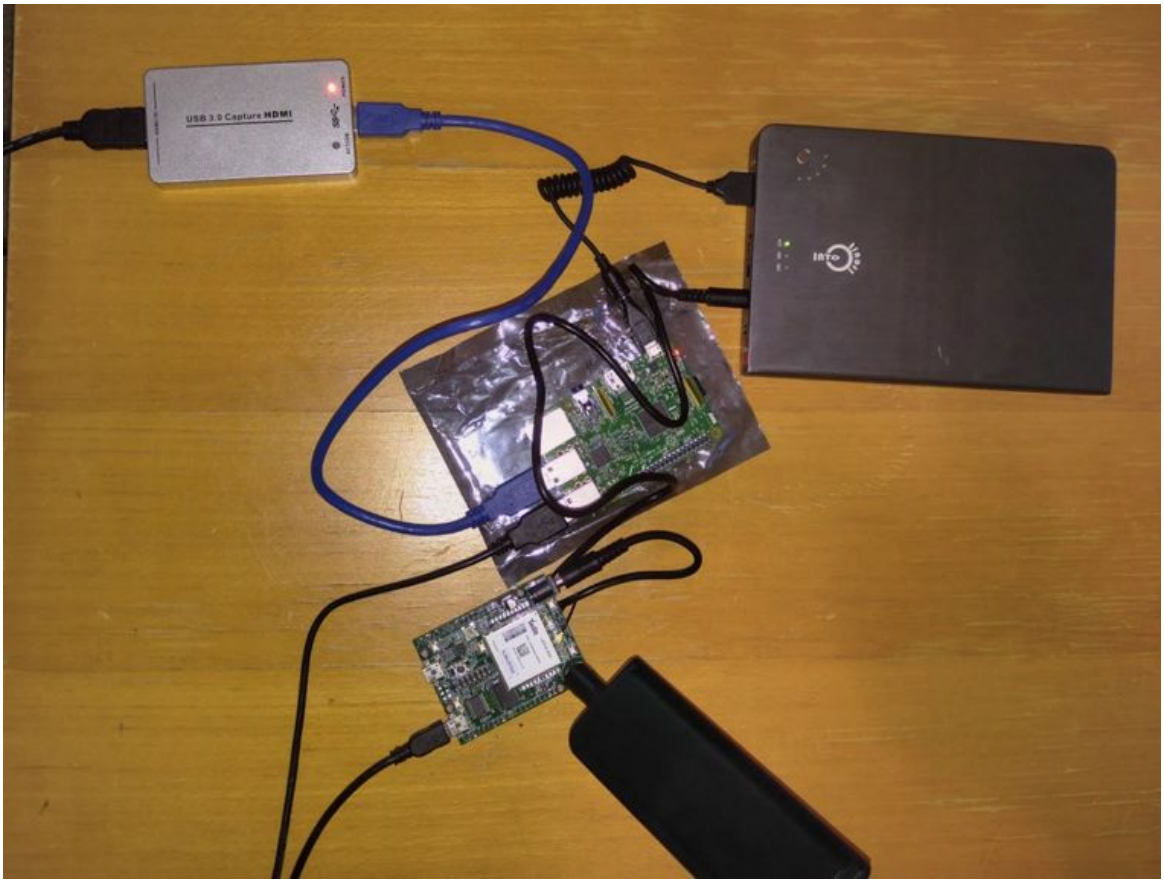
The key to this project is successfully encoding video data into a memory efficient codec and then steaming that data wirelessly to the video web server which allows clients to view and edit the HD video faster than current processes. The Raspberry Pi is the centerpiece of the system and will do most of the redirecting and transferring of data. Despite the significant role, a Raspberry Pi 3 only runs for about $40. The system will require enough storage to store decoded video which is to be upload. This storage will be on a 32GB MicroSD card, which costs only about $15. Once the system has the uploaded the video, a phone application will allow the user to decide on the name, date, time, and location the video was recorded. In order for this to happen the HDMI to USB capture device is needed so that the Raspberry Pi receives the stream. This device will cost about $200. This compressed video data is then transmitted using wireless communications to the webs server. These wireless communications will depend on the situation, but the system will have multiple options. The Raspberry Pi has Wi-Fi and Ethernet built in, so there's no extra cost there. Verizon cell reception is needed however when Wi-Fi and Ethernet aren't available. This will be implemented into a PCB design and has an estimated cost of about $75 - $100. Of course everything will need to receive power in order to function. A 26000mAh battery will provide this power for the entire system. It runs at $65 and will allow the system to operate for at least 9 hours on a full charge. Bringing it all together, a plastic case is needed to house all of the components. Potentially being 3D printed, this cost is estimated at $50.

8.2.3 Sponsorship

The sponsor for this project, Todd DeNoyer, is providing the funding necessary for us to develop a successful prototype. Every expense will be presented to him to explain the need for that specific cost in detail. It is also our responsibility to prevent any outlandish or unnecessary costs while not sacrificing the usability or performance of the QwikBox. The end cost of all the components of the project will be mostly dependent on Todd's needs for the quality of each part as well as the capacity for the battery, storage devices, and processing power.

9. Conclusion

Final Design Concept



Picture 9 – 1

Developing a project that uploads video recorded at sporting events in real time at a reasonable price will be highly beneficial to all high school and youth sports. Reviewing and studying game film is vital for many teams and individuals to make improvements and refine their skills. This becomes even more beneficial when the film is ready for viewing within a few seconds of being recorded. Additionally, having video uploaded to a server platform automatically as its recorded saves video cameramen and managers of game film platforms an immense amount of time and energy. As it exists at the time of writing his, cameramen must record a whole day of sporting events and save them to various SD cards as their recording. These storage devices must be properly labeled and stored until all filming is complete for the day. Once the cameramen go home, they must spend hours uploading their video and many times the upload process lasts until the next day or two when the cameramen can find time. This causes a headache for the game film companies as well as the sporting teams since they have to wait to see their game film.

Video and analytics company QwikCut created an idea to help themselves with the upload process and help the sports teams with getting their video in a much more reasonable amount of time. This is where the QwikBox comes in. Our project will take in a live stream in from a designated HD video camera using a HDMI to USB video capture device and save it to a raw JPEG file. The streaming software used is a command-line tool called MJPG-streamer. From there, the raw video file will use FFMPEG software installed on the Raspberry Pi 3 board to convert the video to a H.265 format. This video conversion and compression is a large component of the project since there is only a limited amount of space on the remote server platform. The H.265 codec provides a 40% - 50% bit rate reduction when compared to a video file of the same quality that is encoded with the h.264 codec, which has been the video codec standard since soon after it was approved in 2003. This bit rate reduction leads to a 40% - 50% in file size as well, which is the main concern for this project. After the conversion, the video files will be added to an upload FIFO queue to be uploaded one at a time to the server. Uploading one video file at a time in an order respective to which file was recorded the earliest is imperative to our project. This is because it contributes to making the total time between the end of a recording and the moment a file is available on the server as small as possible. A queue is needed in the case that the internet strength isn't opportune.

Our project will be operated using a mobile application on the video cameraman's phone. This application will allow for the user to input game information so that the video can be organized on the remote server. It will also allow the user's phone to be connected to the QwikBox using a Bluetooth interface and it will provide controls for starting and stopping the recording process for each video clip. The controls will be very easy to operate and this is important because the operator many time also has to direct the video camera to make sure quality game film is being recorded. The interface will also give the current Bluetooth connection status to assure there is no connectivity issues and no loss of video occurs.

The last major part of the QwikBox project is giving it the ability to communicate with the remote server through a cellular network connection. We will be using the NimBeLink Skywire end-device certified modem to establish a connection with the Verizon network. This modem will be connected to a custom PCB via a UART port and it will be developed using the help of a development kit provided by NimBeLink. The PCB will then be connected to the Raspberry Pi 3 using a USB interface. This ability to connect to a cellular network is very vital to the success of the QwikBox project. Many sporting events for youth and high school sports are played at venues that do not have access to wireless internet or Ethernet connections. This would cause major issues with uploaded video in real time without the use of a cellular network. However, this ability will allow for uploading capabilities at all types of venues so that the features developed for this project are accessible to everyone.

10. Senior Design 2

10.1 Intel NUC

A first we started designing QwikBox with a Raspberry Pi, it presented us with al lot of different issues in which slowed the development of QwikBox. Firstly, it was unable to power all the external systems we required it to power. The modem antenna couldn't obtain a signal from the satellites because of the low power provided by the Pi. This caused us to look at alternative powers sources such as another battery being used. Secondly, the Pi could not properly take in a video signal due to it's minimal processing power. This led to the third issue being that the Pi has couldn't convert videos to the H.265 codec because all the processing power was being used for video capture.

When looking at alternative solutions we decided that we could do the compression on a cloud service to free up processing power on the Pi. This would allow the QwikBox to be even more portable and would allow IoT all-in-one solution to the problem. The only issue with doing it this way that there would always need to be an internet connection and the videos would have to be processed again in order to upload them once they were pulled from the cloud. This waste time and overall can prove to be expensive depending on the type of cloud service used. On top of that there was still the issue of a power supply and the Pi not being able to power the PCB. These issues lead us to the idea of replacing the Pi with another type of system.

Intel NUC

Picture 10 – 1

After further research, the team decided to use the Intel NUC i5 model in order power the new QwikBox system. Right away, all our problems were solved with this more powerful CPU. The Intel NUC was able to capture live HD video and audio stream

from a camera without an loss in visual and auditory quality. This was a huge improvement from the Pi which produced video tearing and out of sync audio. With the video capture perfected and files viewable on the NUC, the next step was to encode each video and compress it to a H.265 codec file such as .mp4. These files were compressed using FFMPEG and have no quality loss and a much smaller file size. Since the NUC is powered by an Intel i5 processor, which features QuickSync technology and native H.265 encoding, the performance of the QwikBox suddenly fit the requirements of the project. Our overall time to capture, encode, and upload was no longer than 15 seconds per video. The new CPU also allowed us to power the PCB which gave our QwikBox Verizon 4G LTE access. The only drawback to the NUC was decreased power life but to our requirements was still satisfactory.

10.2 Software Updates

Originally, our design plan involved using a bluetooth connection for the communications between the mobile application and the central processing device. After switching the processing device from the raspberry pi to a Intel NUC, the bluetooth software that was originally developed was no longer compatible. Additionally, the Ubuntu operating system we installed on the NUC was causing authentication issues when trying to have it communicate with the mobile application. Therefore, using bluetooth for the communications was turning out to be a huge headache.

After exploring other options, we decided to build a python flask API to be hosted on the NUC. The mobile application communicates with the NUC by sending HTTP POST and GET requests to the API. Since the API runs locally on the NUC, it has the power to kick off the necessary scripts to start and stop the recording. Another benefit is that information about each clip can be sent from the mobile application to the NUC within a POST. This information includes the game ID, the teams playing, and the play number. This way each clip can be saved with an applicable file name and organized accordingly.

Since the python flask API runs locally on the NUC, it is only accessible to devices on the same network as it. Since it will be used at sporting events and the user most likely will never be on the same network as the NUC, we had to figure out a way to make the API publicly accessible. After researching potential options, a service called nGrok was decided to be exactly what we need. After installed on the NUC, it successfully forwards a localhost port to a public domain. Once the port and public domain were specified, the mobile application was able to seamlessly communicate with the NUC with very low latency.

The final step was to get the video automatically and quickly uploaded to an Amazon S3 server. A python script was developed to continually check a specified directory for new video files. Once a new file is discovered, it creates a place for it in the

S3 server and uploads the file. The python flask API moves the video file to the specified directory once the user stops recording a clip. This python upload script launches on boot along with the nGrok service and the API.

10.3 PCB Design Updates

When building the PCB board there were a few obstacles to overcome. With the custom PCB board and all the part necessary in hand the actual mounting of the individual parts was a tedious process. Many of the parts were surface mount which was a challenge at first but with a careful eye seems easier to put together than through-hole components. Another difficulty was making sure the individual parts were mounted properly. The data sheet for some parts never indicated on the actual part where each of the pins were located.

After the first attempt at building the PCB board we began testing. During this process the voltage regulator was not working properly and needed closer attention. In order to isolate the problem a second PCB board was built except only the voltage regulator part constructed. After testing this second board, receiving proper voltage drop from 16volts to 4 volts, with a close eye we detected the voltage regulator chip on the first board was improperly mounted. After a simple desoder and flip of 180 degrees the chip was mounted properly and the custom PCB board was fully operational.

Lastly with the final board design completed and a completely operational voltage regulator on a secondary board we decided to complete additional boards with excess parts. Two additional boards along with the first fully operational board have been completed. These additional boards have been field tested and released to the sponsor.

Appendix A

1. Prepare To Be Better." QwikCut » Video & Analytics » Game Film Automated Processing. N.p., n.d. Web. 05 Dec. 2016.
2. "Performance Analysis Tools for Sports Teams and Athletes at Every Level. | Hudl." Hudl. N.p., n.d. Web. 05 Dec. 2016.
3. "HEVC / H.265 Explained." X265. N.p., n.d. Web. 05 Dec. 2016.
4. @Raspberry_Pi. "Help Guides and Resources - How to Use Raspberry Pi." Raspberry Pi. N.p., n.d. Web. 05 Dec. 2016.
5. "Hudl Sideline - Wireless Instant Replay | Hudl." Hudl. N.p., n.d. Web. 01 Dec. 2016.
6. "VLC Media Player." VideoLAN - Official Page for VLC Media Player, the Open Source Video Framework! N.p., n.d. Web. 17 Nov. 2016.
7. Grinberg, Miguel. "How to Build and Run MJPG-Streamer on the Raspberry Pi." How to Build and Run MJPG-Streamer on the Raspberry Pi - Miguelgrinberg.com. N.p., n.d. Web. 03 Nov. 2016.
8. Grinberg, Miguel. "CompilationGuide/RaspberryPi." FFmpeg. N.p., n.d. Web. 12 Nov. 2016.
9. "Bluetooth - Installing and Using Bluetooth on the Raspberry Pi." The Pi Hut. N.p., n.d. Web. 28 Oct. 2016.
10. "SSH (Secure Shell)." SSH (Secure Shell) - Raspberry Pi Documentation. N.p., n.d. Web. 22 Nov. 2016.
11. "VNC (Virtual Network Computing)." VNC (Virtual Network Computing) - Raspberry Pi Documentation. N.p., n.d. Web. 22 Nov. 2016.
12. "Add Code from a Template." Android Studio. N.p., n.d. Web. 10 Oct. 2016.
13. Tutorialspoint.com. "Android UI Layouts." *Www.tutorialspoint.com*. N.p., n.d. Web. 15 Oct. 2016.
14. "Activities." *Android Developers*. N.p., n.d. Web. 15 Oct. 2016.
15. "Start Developing IOS Apps (Swift): Work with View Controllers." *Start Developing IOS Apps (Swift): Work with View Controllers*. N.p., n.d. Web. 17 Oct. 2016.
16. @SwiftLang. "Swift - Apple Developer." *Swift - Apple Developer*. N.p., n.d. Web. 18 Oct. 2016.
17. "Publish Your App." *Android Studio*. N.p., n.d. Web. 01 Dec. 2016.
18. Ziolkowski, John. "How To Submit An App To The App Store (The Right Way)." *Clearbridge Mobile*. N.p., 2016. Web. 01 Dec. 2016.
19. Bruce, James. "Intocircuit Power Castle 26,000mAh Portable Charger Review and Giveaway." *MakeUseOf*. N.p., n.d. Web. 04 Nov. 2016.
20. "300W Power Inverter." Bestek 300W Inverter - 12V DC to 110V AC. N.p., n.d. Web. 05 Dec. 2016.
21. "Intocircuit Power Castle 26,000mAh Portable Charger Review and Giveaway." MakeUseOf. N.p., n.d. Web. 05 Dec. 2016.
22. "Mophie Powerstation® - Smartphones & USB Devices." Mophie Powerstation® - Smartphones & USB Devices. N.p., n.d. Web. 05 Dec. 2016.
23. "HDMI." Wikipedia. Wikimedia Foundation, n.d. Web. 05 Dec. 2016.

24. "HDMI Video Capture,Rongyuxuan HDMI to USB 3.0 1080P Video and Audio Capture Device for Xbox PS3 PS4 DVD,Video Game Recorder Compatible with Win7/8/10 Mac Linux OS." Red Hot Video Games. N.p., n.d. Web. 05 Dec. 2016.
25. "PandaBoard - Linux on ARM - Eewiki." PandaBoard - Linux on ARM - Eewiki. N.p., n.d. Web. 05 Dec. 2016.
26. "Banana Pi - A Highend Single-Board Computer." Banana Pi - A Highend Single-Board Computer. N.p., n.d. Web. 05 Dec. 2016.
27. @nimbelink. "Skywire™ 4G LTE CAT 3 Embedded Modem - NimbeLink : Build Your IoT Device Faster." NimbeLink : Build Your IoT Device Faster. N.p., n.d. Web. 05 Dec. 2016.
28. "PiAnywhere - 4G, GPS and Power for the Raspberry Pi." PiAnywhere. N.p., n.d. Web. 05 Dec. 2016.
29. "H.264 Pro Recorder." Blackmagic Design: H.264 Pro Recorder. N.p., n.d. Web. 05 Dec. 2016.
30. "Matrox Maevex™ H.264 Encoders & Decoders." Matrox Graphics RSS Events. N.p., n.d. Web. 05 Dec. 2016.
31. "ARM1176 Processor." ARM1176 Processor - ARM. N.p., n.d. Web. 05 Dec. 2016.
32. "Skywire Development Kit." (n.d.): n. pag. Nimbelink. 4 June 2014. Web. <http://nimbelink.com/Documentation/Development_Kits/NL-SWDK/20002_NL-SWDK_Schematic.pdf>.