Qwikcut Game Film

Mark Escott, Steven Little, Kevin Brown, Santiago Alvarez

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The purpose of this project was to create a new system for QwikCut. This system is to implement a camera that can pan tilt and zoom via a remote input. This project utilizes power over Ethernet to allow for a one wire connection to the camera. The whole system is battery powered and must last up to twelve hours. For controlling the camera, a custom software was written in Java to read a controller and talk to the camera while displaying useful information for the operator. The camera communication is handled via an industry standard called Open Network Video Interface Forum.**

***Index Terms* — Power Over Ethernet (POE), ONVIF, Battery Management System (BMS**

I. Introduction

QwikCut is a video and analytics company based in the central florida area that is contracted to go to high school athletics such as Football games, Soccer matches, lacrosse matches, etc and their objective is to record, stream, and relay these games in real time from their current camera and tripod setup. The current setup as listed above has problems with its three cables system in which these cables are always needing to be replaced which is a great expense, the motorized head that is used at the top of tripod does not pan fast enough for sports and uses 4- AA batteries as a power unit, and the HDMI that is currently used does not carry audio while streaming the video. These items are to be addressed and improved upon.

Created to solve the problems of the existing recording system. Qwikcut has reached out to recreate their system of recording and streaming sports games from the sideline. Designed as a part of a computer engineering and electrical engineering project, this new system hardware and software solves current issues and satisfies required specifications by the sponsor. The goal for our sponsor is to create a portable, lightweight, and overall simple system that can be taken to a wide variety of schools, athletic events, and locations around the central florida area and be used by the QwikCut team in order to make their recording and streaming of games as efficient and easy as possible. These benefits are achieved mainly through a new and improved power over ethernet system, pan-tilt-zoom camera, custom PoE Injector, Laptop, Battery storage systems for power and custom software.

II. System Components

Qwikcut Game Film design can be split into multiple components, which will be introduced here. Below, a system components diagram can be seen.

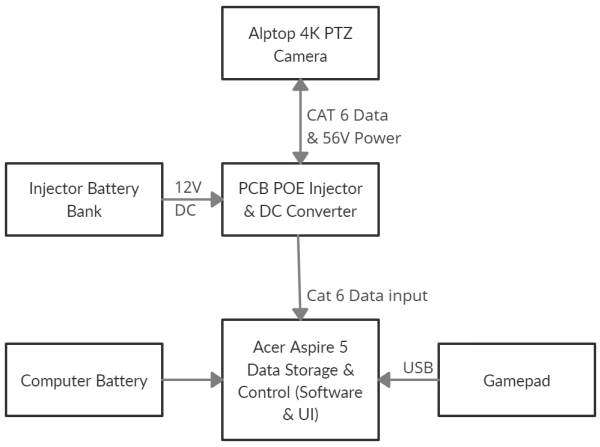

Fig 1. General Hardware Flowchart

A. PCB PoE Injector

In order to achieve an efficient system we needed to be easily able to power a camera at the top of a 25ft tripod. One request from our sponsors at QwikCut was to power this system via a singular cable that runs from the camera back to the storage container/main control point that the connection will be run to. After some deliberation and research we looked into many different methods and designs possible to us in order to power this system we ended up choosing Power Over Ethernet (POE) as our main power source to our camera.

Using Power Over Ethernet allows us to power a camera and control this camera over a singular cat5 or above ethernet connection that can simply run from the camera to our laptop. However, a standard laptop or computer would not be able to power a camera alone and would thus require a separate power source to meet its power requirements. Due to the nature of the POE this required us to use a POE Injector to power up our camera and transfer the feed that runs back up from the camera to our laptop all in one small package and simple connection.

POE injectors supply a constant 48-56V voltage to many different cameras and electronics in the world today and must adhere to IEEE 802.3 standards in order to make sure that these electronics are powered safely and in a

responsible fashion to save the products from overloading and causing issues to customers and consumers. Creating a POE Injector to meet these standards and power a camera in a system that can be simply assembled in the field was our main goal in this endeavour and ended up being our choice to tackle this task from our sponsor.

B. Injector Battery Bank and Computer Battery

The battery for the injector is being designed with 18650 lithium ion batteries. Using the lithium ion battery will be the best benefit to the customer mainly for maintenance, size, and ventilation for more waterproof applications. It is one of the most mature lithium ion formats available and is produced in high volume with a low cost per watt hour. This battery bank will be providing the PoE injector with the power it needs to keep the camera running for up to 12 hours.

The battery for the computer can also be created using 18650 batteries like the injector. On the other hand, the existing qwikcut box which stores all the batteries for each system includes a 50 amp hour battery that can cover the needs of the laptop for an extended period of time. Using this will save the sponsor money by reusing useful existing hardware, especially when it comes to buying more systems for manufacturing more of these on a larger scale in the future.

C. Computer

The computer needed for this project had three main requirements. The computer must have a battery that can last a minimum of five hours under continuous use, it must have a standard RJ45 Ethernet port, and it is able to run modern applications. One of the main requirements of this project was being able to operate in all weather conditions, however, due to the cost associated with water resistant or ruggedized laptops this requirement could not be met while remaining in budget. The response to not being able to meet this requirement is that the laptop can operate up to one hundred meters away from the QwikBox that supplies power to the camera without any loss of communication. We hope that this ability allows for QwikCut to find a suitably safe environment for the laptop in the event of inclement weather.

Surprisingly finding a modern laptop in the price point of under seven hundred dollars that had the features we needed was difficult. We ended up using an Acer Aspire 5 as it was the only laptop in the price point that met the requirements and was in stock.

D. Camera

For the camera it was decided to use an existing product rather than designing a custom solution. For this option security cameras were deemed the best solution as they can have the pan, tilt, zoom feature built in, have decent image quality and are relatively affordable compared to equipment that is specially designed for this task. Most modern PTZ security cameras have implemented an industry standard to make the communication easier and more efficient.

The process of finding an affordable camera that also meets the image quality and feature set requirements was one of the biggest obstacles about not making a custom solution. The initial concern was about finding a camera that could move fast enough for QwikCut's needs. However, the camera that we chose first lacked the ability to zoom in enough. Afterwards we found another camera that after messing with settings met all requirements and feature set needs, this camera is the Alptop 4K PTZ camera.

E. Software

Although existing software solutions exist that could be used to move the camera through standardized communication protocols none of these meet the requirement that the input be read from a gamepad style controller. Because of this we have designed and made a custom piece of software that makes the camera easy to control and presents the end user with any additional information needed to stream and record the video feed.

During the process of researching how we can make a piece of software talk to the camera we found that there is an industry standard that was designed to make this process easier, Open Network Video Interface Form, or ONVIF allows us to make simple POST and GET calls to perform the actions needed for the camera to move..

III. System Concepts

As a system, our Qwikcut design works by using PoE. It is a medium for carrying electric power over traditional data cables (CAT cables). With the PoE specified devices, we are passing a current through the ethernet cable along with the data that is also carried over by ethernet cable. These power over ethernet devices supply power according to the IEEE 802.3 device generation. Electrical current flows in a loop, so we need at least two conductors to carry our electric current. This is obtained by the PoE treating each pair as a single conductor, so it will utilize either the two spare pairs or even the two data pairs to carry that current. Power over ethernet is injected onto the cable at a voltage between 48 and 56 volts DC to the

camera. This configuration allows for us to use one connection for power and data.

The computer is the central part of the data stream. The computer collects user input from the gamepad controller and processes the data generated before telling the camera to move according to the input. The computer also handles all the video capturing and processing. QwikCut already uses 3rd party software to accomplish this and it will be reused. The 3rd party software called Open Broadcasting Software (OBS) handles the recording and streaming of the video feed, while the camera records a copy internally to ensure QwikCut never loses footage.

IV. Software Details

In this section the description of the software and its components will be broken down more and discussed in more detail. The software created for this project is broken down into three main parts, the UI, the controller logic, the camera logic. Since this software requires the monitoring of the controller and the communicating with the camera to happen simultaneously and continuously without causing the UI to be affected the software uses multi-threading to ensure it appears very seamless to the end user. The software also utilizes many prebuilt libraries to make the development easier and reduce the need for creating custom code that would be harder to keep updated in the long run.

A. Language choice

One of the first decisions that was made was which programming language to use for this project. Since this project has a UI we wished to make sure the language we decide to use has a system to easily make the UI. The next decision point was how to talk to the camera, this included a brief discussion of making a custom PTZ implementation or using existing ones. Once settled on using existing security based cameras with ONVIF being available as a method of communication we had to make sure there was a simple way to utilize ONVIF. The next limiting factor was determining which languages have an easy to implement method of reading USB inputs. After these decisions we were left with using C#, C++, Java, and Python. This list was driven by which languages were familiar to the team along with abilities. The final decision was to use Java over the others as it was the most familiar and can easily implement the needed features. C# was the alternative if Java ended up not working as it has a large community which would allow us to quickly learn how to accomplish things. Python was on the list since a large number of people develop simple libraries and these covered all aspects of what we needed to do, however, Python is not as optimized as other solutions so we were concerned that it would utilize too many resources and affect the battery life of the computer.

B. Controller Input

The first main aspect researched and handled for this software was how to collect, handle, and process the input from a gamepad style controller. The method of choice was using the Jinput library which makes accessing and reading inputs from Universal Serial Bus (USB) devices very simple.

The sponsor left the requirements for the control very open for interpretation. The requirements that were decided are as follows, the controller should be able to be replaced easily, the software should be able to handle multiple styles of controller, and some tuning of the inputs should be adjustable to the end user.

Since QwikCut has already purchased many of the same logitech controllers we elected to make that style of input the default. Currently due to time constraints and availability of other styles of controller we have not tested or implemented any other styles of controller that may map the inputs to different buttons. The controller also cannot be replaced while the software is running as the software only scans the USB devices on startup and once selected the controller is assumed to be connected until the software closes.

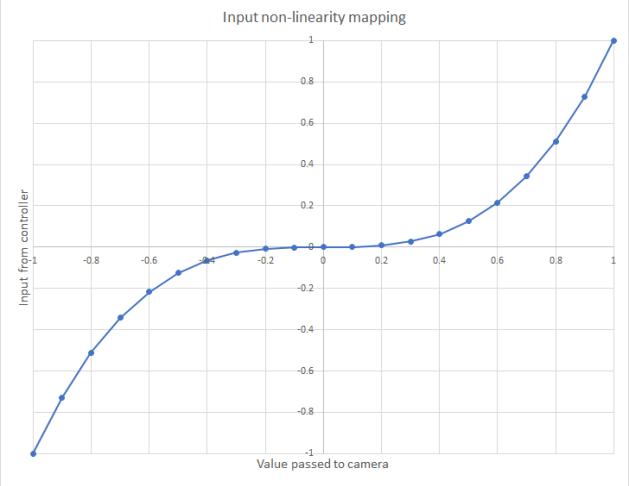

Fig. 2. Controller non-linearity curve.

One of the control features that we determined would be beneficial was the addition of having the input be not linearly mapped to the speed of the camera. This allows for the operator to have more slow speed control over a

larger area of the input but retain the fast movement speeds on the extremities of the input. The curve we decided to implement is shown in figure 2 and the equation is listed in equation 1. This equation is applied as soon as the input is read and the specific axis is flagged as non-linear. This setting can be changed for all three axes independently.

$$Camera\ value = (\text{Controller input})^3 \quad (1)$$

Eqn 1: Controller non-linear mapping equation

The current setup for the controller allows the user to pan using the left joystick in the horizontal direction, tilt with the right joystick in the vertical direction, and zoom with the right and left bumpers for zooming in and out respectively. This setup came from using pre-existing controls from other software, so as to be more intuitive to a new user. Using the continuous nature of a joystick for movement can allow users to be more precise than they would otherwise be able to with discrete buttons. Furthermore we currently have implemented speed limits and dead zones when it comes to reading the user input. The maximum speed of the camera far surpasses the need of the sponsor, as such we have created a way to limit this speed so as to be more precise within the spectrum needed. Dead zones were implemented to handle any small input that occurs from degradation of the controller's joysticks. This allowed us to ignore values beneath a user defined level as to avoid unwanted stutter. The last feature concerning the controller input is the addition of a speed increase button. To deal with the need to move the camera quickly at times, the right bumper allows the user to double their current speed. It takes the user's set speed limit and doubles it so as to allow for fast movement, this led to a potential issue where the speed limit set could exceed the maximum value, but cases such as those were handled. Future potential features include being able to map the inputs to different user defined buttons or for the user to be able to change their own movement curve to best suit their needs. Lastly user profiles could be added to set the multitude of settings to set default values depending on the profile selected. To ensure the camera and software meet our specifications those features have been left for future development time.

C. User Interface

The next aspect that was tackled in the development process was the design and creation of a UI that was simple and effective. Since this software was never to handle the processing or displaying of the video feed of the camera we took the approach of having a small simple UI that gives enough control and customization without taking up too much screen real estate. To achieve this we used Java AWT and Java Swing to make four main UI's that each handle an aspect of the program with one main UI driving the other three.

Since Java Swing handles the UIs in a separate thread to the rest of the program we had to ensure we made all the data needed for the UI thread safe. This was done via interfaces and getter and setter methods that had the information populated as soon as it was available. For example we display the controller input on the screen live, to allow for this we redraw the main UI when it is active and read the last controller input that was collected via a 100ms polling loop.

D. Camera Control Logic

The last aspect of the software was the most complicated to get to work. Since the camera selection was limited to security cameras to allow for QwikCut to easily mass produce the system our software had to implement a way to talk to security cameras regardless of manufacture. ONVIF is the most widely used industry standard that spans multiple manufactures.

One of the main issues we faced was the availability of updated and open source libraries that allow for an easy implementation of ONVIF in Java. Many of the libraries were either not documented or were poorly documented so understanding how to use the API was a challenge. The one that ended up working was onvif-java-lib [1], however it has not been maintained and most of the supporting libraries it used have been depreciated and replaced. Along with this most of the documentation was in german and some aspects had to be translated to be able to understand what the function was and what was required to use the function. Because of this we have heavily modified the library to use modern Java libraries and versions along with an update of the ONVIF standard used to match a more recent version. The library uses Simple Object Access Protocol (SOAP) to translate the XML the camera uses over HTTP into Java objects that can be used in the software to perform the necessary actions.

Since the ONVIF library was not able to be fully updated without rewriting a large portion of the library we elected to slightly limit our camera selection to those that allow for the access of features without having HTTPS communication. During our search for a camera we found that some cameras allow for the authentication step of ONVIF to be ignored via a setting. As the camera will be the only device on the network outside of the controlling

computer and this computer will have a separate IP space for internet access this security vulnerability is acceptable.

Once the libraries were updated the software was able to control the camera and read needed information such as the camera's video stream URI and supported functions such as movement methods. However, some challenges were faced in the connection step; typically ONVIF navigates of port 8080 or 8000 and some of the cheaper cameras we tested used port 80 which forced us to adjust how the user provides the connection information to allow for a port to be specified.

Along with the connection challenges it was found that for every request for movement the cameras came to a complete stop before moving again. Because of this we had a limitation of only being able to use the continuous movement method. We had to adjust how we requested movement to only send a request when the demanded movement value from the controller exceeded a threshold. To achieve this we calculated the change in input value every ten milliseconds and if it passed a predefined threshold it would then allow the camera to pause and change its speed or direction. This threshold is scaled to the maximum value that can be demanded to allow for a seamless change when the maximum camera speed is limited.

During a meeting with the sponsor the camera's movement speed was found to far exceed what was needed in application and it was requested we find a way to easily slow the camera down. When this slow down was demonstrated it was also requested we have a method to temporarily increase the speed when a button is depressed on the controller. To achieve the request of slowing down the camera we applied a scale to the input values that is adjustable by the user. Since the camera and controller values are float values between -1 and 1 we simply scale the controllers input by a percentage to limit the value the camera receives. This scale is then overrode when a specific button is depressed.

E. Software packaging

Since we used Java we could not natively make an exe file from our runnable java. Although we could have simply distributed a runnable jar file if for some reason an end user wished to run the software on a computer without java installed or with an improper version there is no easy way to inform the end user of this. We found in researching a solution to this problem that we could wrap the runnable jar into an exe and have error messages with links built right into the exe for both an invalid java version and missing java. Along with this the wrapping allows us to apply a mutex to the software to make sure only one instance is ever open at a time. This mutex is necessary since the Jinput library requires specific files to be accessible and if an instance is already running those files will be locked and the program will not function. To wrap our jar file we use a software called launch4j [2] and specify all the requirements needed.

V. Hardware Details

The hardware details of this system consists of more in depth components touched upon earlier. We have a Camera, Custom PoE Injector, Custom Injector Battery Bank, Computer, Computer battery, and Qwikbox for water resistant storage.

A. Camera

As discussed earlier we needed to design a system that could easily be powered and controlled via a singular cable. The most efficient way of doing this was to use either a cat5 or cat6 ethernet cable that could power the existing PTZ camera we chose. The Alptop 4K PTZ Outdoor camera was the perfect choice as it met some of the most important categories that we needed for this project. This camera choice was chosen for the following features it possesses:

Quality: Our sponsors at QwikCut requested that our system be able to achieve a minimum quality of 1080p High Definition picture. This PTZ camera possesses 8 MegaPixel camera quality or 4K HD as its image processing component. This would give our sponsor more than adequate video quality and definition to stream and record games for customers and any patron interested in the QwikCut products. As well as having 30 frames per second capabilities we are able to see clearer pictures in a smoother transition and video feed versus a choppy and non immersive camera feed that other brands and models on the market today can offer.

Pan: As a group constraint we needed this camera to have a minimum of 20 degrees per second horizontal rotation. This camera has the ability to pan up to 65 degrees per second and has a full 360 degree field of motion. This is perfect for looking at a full field of view for sporting events and other atmospheres that our sponsor may need.

Tilt: Another group constraint consists of a minimum of 10 degrees per second in the vertical direction both upward and downward. The Alptop 4K camera is able to move at 45 degrees per second and has a vertical field of motion from -5 to 90 degrees making it a suitable choice in the movement category.

Zoom: For a suitable camera that will work for our sponsor we needed a camera that had a minimum of 20x

zoom in order to achieve the correct level of focus and distance coverage needed for covering up to 300 ft of field. This camera is able to zoom up to 30x magnification and has great picture quality when zoomed in all the way.

POE Capabilities: In order to meet our sponsors desires of having a singular cord we needed to have a camera that was POE/POE+ capable. This camera has POE+ capabilities and is able to be powered through a singular ethernet cable and an injector to power the camera. This falls under the 802.3 at /af standards and is certified under these guidelines.

IP66 Waterproofing: Due to the nature of our sponsors' needs and everyday use we needed a product that would not be damaged by any wind, rain, or solar exposure that it may be presented with over the course of a normal field test or everyday use. This camera is IP66 waterproof meaning that it is dust proof and will not be damaged by any normal weather exposure and conditions that it may face out in the field and when in use.

Pricing: A big factor in this redesign and implementation of this project was cost. A main objective in our new design was to make a system that would come in at an overall lower cost than that of the previous system. This camera comes in at a price point of around $330 on many online retailers, this price although may be higher than the previous camera in use will still let us be able to come in under budget in the overall design giving room for a more expensive unit. Also, after comparing many makes and models that many retailers are presenting today this camera presents a clear balance between the price paid and the features we are given at this price point. Many models above this price level do not give our design a better benefit of a higher quality of features and many units at a lower value would not suffice in meeting the overall specifications and constraints needed by our project.

ONVIF compatibility: This allows for controlling the camera and retrieving information from the camera easier. In order to make sure we can control the movement of the camera via our laptop and game controller, ONVIF compatibility is vital to the overall success of our design. This camera is ONVIF certified and has plenty of accommodations that can be made in order to assist our sponsor if any tweaks or alterations to the movement or communication between components is needed.

Overall, This camera is more than sufficient enough to be put into our design and has great specifications that are crucial in our design.

B. PCB PoE Injector

In order to power our PTZ camera through a singular ethernet connection we needed a POE Injector that could remotely power it and send that data back to the laptop all in a small and effective package. This led us to design and construct our own PCB Injector that can be used to power POE and POE+ applications autonomously and efficiently without causing any problems with other components in the system. This PCB design would need to adhere to IEEE 803.3 standards and be capable of handling the "handshake" process that is done between Powered Device (PD) and Power Sourcing Equipment (PSE). This communication between the powering device and power source allows them to exchange information and let the other know what power requirements are needed to safely function within the perimeters of the device.

IEEE 802.3 standards dictate that for standard 802.3af or standard POE applications that a nominal 48V is necessary for powering a device efficiently and without causing the device to be underpowered. This range can be slightly lower or above as seen in the Power breakdown table table below in *Table 1*. However, for 802.3at models and equipment a nominal 52V voltage is needed in order to power a device efficiently. The difference in power needs is where the difference in category.

Property	PoE	PoE+
Standard	IEEE 802.3af	IEEE 802.3at
Type	Type 1	Type 2
Max Power at Powered Device	12.95 Watts	25.50 Watts
Max Power delivered at Switch	15.40 Watts	34.20 Watts
Voltage Range at PD	37.0-57.0 V	42.5 – 57.0 V
Power Management	Class 0 (unclassified) Class 1, 2, 3 negotiated @ initial connection.	Class 4 at initial connection 0.1 Watt steps negotiated continuously.
Cabling	Cat 3 & Cat 5	Cat 5

Table. 1. POE/POE+ Power Breakdown

In order to achieve a IEEE802.3 at/af POE Injector we developed a PCB with the following characteristics: A autonomous voltage regulator to convert the 12V DC input into a 52V or larger voltage supply to meet POE standards, A autonomous PSE Controller that can regulate the input voltage that our DC power supply can generate, another voltage regulator to power our autonomous controller with a 3.3V voltage, and finally a set of transformers to regulate the voltage over the correct pairs so that the voltage is efficiently going into our PTZ camera over the correct pairs.

Generally for many POE designs there are two standard modes and pairs used in power generation. Mode A is the commonly used one uses pairs 12 as positive and 36 as negative. Alternatively, Mode B used 45 as positive and 78

as negative. Either method can be used and both models were designed as suitable options in our project.

Overall,This POE Injector will power our PTZ camera and will relay the connection and feed given off by the camera to our laptop.

C. PoE Injector Battery Bank

We created a battery bank using 18650 lithium ion batteries. The specifications of these battery cells include 3.7V and 3400mAh. The POE camera includes a draw of 600mA at 48V which would give us 2400mA at 12V which is our desired voltage. We can place these batteries in a series of 3 to reach our 12V voltage range for the battery bank. Our system is thought of to be around 80% efficient after transmission which would give us a draw of 600mA at 48V still at the camera but coming from the battery would give us 2880mA at 12V desired. This is an 80% conversion which increases our battery power from 28.8W to 34.56W. We know we need a 12V battery bank, so looking at the specs of our 18650 batteries, we need them to be in a series of 3 which would increase our total voltage to 11.1V on the low end. The capacity of the battery is now calculated to make a battery bank that will last the specified time of 10-12 hours to meet the specifications.

$$Bank\ Ah = Batteries\ Parallel \times Battery\ (Ah) \quad (2)$$

Eqn 2: Amp Hour Battery Bank

$$Bank\ Hours = Bank\ Ah \div PoE\ Draw(mA) \quad (3)$$

Eqn 3: Battery Bank Hours Provided

After finding out the battery bank is best to have 10 18650 cells in parallel while being in a series of 3, we reach a total time of the bank being 11.81 hours or 11 hours and 48 minutes.

The Battery bank also needs protection while charging and discharging. Our protection involves the use of a microcontroller to compensate for temperature rise, adjust the charge current and charge with time according to the battery specifications. Using a system like this drastically extends the battery life and is most commonly used with the Lithium ion battery types. That is a main reason why we are using a battery management systems (BMS) type microcontroller for our system. A microcontroller like this can be fitted externally to the charger or the batteries themselves. The 3S BMS Board that manages the lithium ion battery pack. This offers an overvoltage range of 4.25-4.35V and over discharge voltage range of 2.3-3V maximum operating current of 0-25A making the complete name a 3S 25A BMS board.

For connection, our BMS had four soldering pads, a B-, B1, B2, and B+. We first connected the first parallel group. We will connect the negative terminal bus to the B- and the positive terminal bus to the B1. We also connected the third parallel group negative terminal bus to the B2 and positive terminal bus to the B+. Spot welding the nickel strips to the PCB will provide a sturdy connection where we can solder on our 2.1mm DC output port and battery level indicator so the user can see the level of the batteries.

The physical size of our battery bank requires a container of 243.2mm x 84.2mm x 72mm. We created a custom 3D printed container that has designs made in .stl format. This design keeps the battery from moving around, while also giving the BMS enough room to wire in a battery level indicator and DC output.

D. Computer and Battery

The computer we have selected for this project best fits in the form of a laptop. This is due to it being more lightweight and easy to use. The drawbacks to using a laptop instead of a normal tower PC is a lack of customization. The laptop we are using for our design is the Acer Aspire 5 Notebook. It meets all the specs we need in the forms of storage, ports for connections and battery life to help us get the most time possible on the field. For power specifications, the laptop contains a 4-cell lithium ion battery along with a 45 watt AC adaptor for charging when not in use. With this, we have a baseline battery life of around 5 hours. The existing G3 Tiger K2 battery that is being used in the QwikBox has a large capacity of 50,000mAh. Using a 3 amp per hour draw from the laptop at most, we would be looking at just over a 16 hour battery life for the laptop just using the existing G3 Tiger alone. This is a sufficient amount of time for our goal of 10-12 hours of uptime for the system. Not only will this be cheaper for QwikCut because these units are already in the box meaning they will not have to re-purchase a battery for the system, but this is a battery everyone is already familiar with.

E. Qwikbox

Used in the existing QwikCut system currently is a plastic type Kobalt with metal latches. The tongue and groove construction keeps water out and adds to the strength of keeping out anything that may be harmful. The

main function of the Qwikbox is the storage it currently provides. There are multiple sets of batteries, chords, the camera and wall adaptors. This new Qwikbox is meant to keep the majority of the power components inside the box at all times, minimizing clutter and keeping batteries out of the weather. It would be of benefit to us and a benefit to the customer to keep this Kobalt box known now as the QwikBox for the storage of our design which now has plenty of room. The QwikBox has no known issues of leakage, overheating, or not being able to protect the batteries and devices. This will save money in the budget, as long as it saves the time of replacing all the QwikBoxes that are already owned and in place.

F. Camera mounting

The last piece of hardware needed is a custom mount that is to adapt the existing tripod QwikCut has to a wall mount security camera. Since the existing tripod is designed to accept a standard camera its method for attaching is a standard ¼-20 thread rod that is inserted on the end of the tripod. Because of this we can simply have our custom mount have a threaded hole or nut attached to secure the mount to the tripod. The mount's design is to be as simple and easily manufactured as possible while still protecting the camera and being easy to attach. The design in mind is a simple ninety degree bent plate of steel that centers the camera body over the attachment to the tripod and has the needed whole to pass the wire through and attach the PTZ camera. Depending on the stability additional gussets added to reduce swaying of the tripod and camera as much as possible.

VI. Conclusion

The QwikCut redesign improves upon all aspects of the previous design in that it features more adequate and updated technology that can be used to better serve QwikCut themselves but also their customers. As a group we feel as if we have met those goals and are proud to present Qwikcut with something they can expand upon and produce as their own.

Meet the Engineers

Mark Escott is currently a senior at the University of Central Florida as an electrical engineering student who is taking a job with SGM Engineering which is an MEP Consulting firm in Orlando FL, as an Electrical designer.

Steven Little is a current senior at University of Central Florida as a Computer Engineering Student

Santiago Alvarez is a current senior at University of central florida as a Computer Engineering Student

Kevin Brown is a current senior at the University of Central Florida as an Electrical Engineering student who is looking for employment in the central florida area.

Acknowledgement

The authors wish to acknowledge the assistance and support of the QwikCut staff and sponsors along with all of the professors and advisors throughout our tenure at UCF.

References

[1] stevenlittle-UCF, "stevenlittle-UCF/onvif-java-lib," GitHub. [Online]. Available: https://github.com/stevenlittle-UCF/onvif-java-lib.

[2] Kowal, G. (n.d.). *Cross-platform Java executable wrapper*. Launch4j. http://launch4j.sourceforge.net/.

[3] "Advantages and limitations of the Different Types of Batteries - Battery University", Batteryuniversity.com, 2021.[Online].Available: https://batteryuniversity.com/learn/archive/whats_the_best_battery.

[4] "Power over Ethernet—Supply of Ethernet Devices Via Data Lines | Analog Devices", Analog.com, 2021. [Online]. Available: https://www.analog.com/en/technical-articles/power-over-ethernet-supply-of-ethernet-devices-via-data-lines.html

[5] "PoE+MeansMorePowerToYou!" PathSolutions.com.2