# Automated Vehicle Anti-Theft Security System

Adam Hadiji, Sean Perraud, Benjamin Hochstadt, Gerardo Pineda

### Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2450

***Abstract* —** **This document explains the design of the solar powered anti-theft car security system, including components used and overall implementation. The project takes advantage of Android user interface, solar charging, MPPT/PWM charge controllers, Raspberry Pi communication and two-step sensor detection. A large chassis that sits on top of a car will house four raspberry Pi modules that will communicate with each other, record video and detect motion with two different types of sensors. A solar panel will charge the system and will be controlled using a charge controller. This document also explains how the design was created and how each component works.**

***Index Terms* — MPPT/PWM charge controller, Android user interface, Raspberry Pi communication, two-step sensor detection, solar panel, solar charging**

### I. Introduction

We have created a prototype for a solar powered car anti-theft security system. This system can send notifications, live stream video, watch for car theft and act as a dashcam from each side of a car. Our design consists of a curved box with cameras and sensors that can be put on top of a car. For the charging portion, a solar panel is used to power the design. A large capacity battery is connected to the solar panel through a charge controller to safely control the charging of the system. A load is connected through a printable circuit board to the battery to step down the battery voltage from 12 volts down to 5 volts.

To make the user's car as secure as possible, motion sensing technology has been implemented into the design that can detect movement even when the car is completely off without using the car's battery. Once motion is detected a camera will turn on and take pictures until motion is no longer detected. The car owner will also be notified through phone by a generic message when the car is shaken and motion is detected making them aware of the incident at the same time it has happened.

Very few current car security systems implement these technologies, we are not aware of any that currently send real time alerts to the car owner when motion has been detected [1]. Our design has multiple modes that can be easily controlled via phone application. When our design is set to Alert Mode, the system first checks for threats and sends alerts. The system can also be placed in dashcam mode allowing for the user to record from each side of the car constantly. Another mode that can be used is a live streaming mode, where the user can select the side of the car they want to view and watch a live stream from that side.

### II. Overview

For our project, we decided we wanted to build a system that was both battery powered and wireless allowing for our device to be used anywhere. This was essential due to our device being placed on cars which are always on the go. In order to accomplish this, we set out a few major requirements that we must follow:

* Device no larger than the roof of a car
* Powered using solar energy
* Coverage of all sides of the vehicle
* Capable of sending wireless alerts

Our design ended up being 48” x 30” x 8” which is a good fit for most compact cars. On the top of our chassis is a 100W solar panel. This panel is used to charge the battery that powers the entire system. The solar panel is connected to a charge controller PCB. This charge controller regulates the power given to both the battery, and from the battery to our load. Our load in this case is our voltage regulator, which steps down the voltage for our microcontrollers. The brain of the project is four Raspberry Pi that are clustered together. This means that one acts as a head node while the others are communicating as slave nodes. This cluster is what processes and stores the video feed from the four separate cameras. A combination of both motion detection and vibration detection is utilized for determining when a potential intruder is at the car. From these detections, a notification is sent wirelessly to an application on a user’s smartphone. This notification allows the user to then open live feed to see what exactly is approaching the vehicle. From there, the user is able to then send an audio alert back to the car to try and ward off the intruder. If the user is instead driving the car, the device can be set to dashcam mode. In this mode, all detection is paused and instead the cameras all start recording and saving video internally. As more video is recorded, files start to delete themselves to make room for newer footage. The app has several features on it including seeing the video feeds and also changing all the configuration settings like dashcam video length. Altogether, our device works as a wireless car security system.

### III. Peripherals

Our project relies heavily on the use of different peripherals. Because of this, our device needed a microcontroller able to process all the different peripherals and stream all while also being able to store lots of pictures and video feed. The only microcontroller that fit our needs was the Raspberry Pi 3 Model B. Compared to most other microcontroller, the Raspberry Pi is more of a microcomputer. When compared to our next best option for a microcontroller, the Arduino Mega 2560, the Raspberry Pi excels in every category.

|  |  |  |
| --- | --- | --- |
| Board Name | Arduino Mega 2560 [2] | Raspberry Pi 3 [3] |
| CPU | 16 MHz ATmega 328P | 1.2 GHz Quad-core ARMv8 |
| RAM | 8 kB | 1 GB |
| Storage | 256 kB Flash | MicroSD (32GB) |
| Network | N/A | 10/100 Ethernet, 802.11n/Bluetooth 4.0 |
| USB ports | 1 | 4 |
| GPIO | 54 x Digital I/O, 15x PWM digital, 16 x Analog | 40 x GPIO pins |
| Operating Voltage | 5V | 5V |

Table 1 – Microcontroller Comparison

The Raspberry Pi having wireless network capabilities integrated also saved money and time. For motion detection, we ended up going with simple PIR sensors. These sensors are cheap and super simple to use. They utilize infrared light and change in temperature to determine if there is movement. If there is, the output is set to high, if not it is low. In combination with the PIR sensor we used an IMU for vibration detection. An IMU is an Inertial Measurement Unit which consists of both a gyroscope and an accelerometer as well as others we didn’t use. The gyroscope is used to measure angular velocity while the accelerometer measures acceleration forces. Together, they help is determining whether the vehicle was being tampered with. We went with BerryIMU which is a unit designed to work with our Pi. For the cameras, we ended up going with the Raspberry Pi Camera Module v2. This is a camera module made for the pi. It makes the software side for dealing with video feed much simpler. It also has a streaming resolution of 1280 x 720 at 60 fps. This was more than we could have asked for. The Pi will also have a GSM module. This will be used for sending text notifications. The last peripheral used are a set of speakers. These are cheap speakers that are powered through the USB port on the Pi.

### IV. Hardware

1. *Charging System*

For our design, we decided to go with a 12-volt battery with a 10 amp hour capacity that would be connected to a 100 watt solar panel through a charge controller. The load consisting of four Raspberry Pi modules, an ethernet hub and a USB hub will be connected to the battery as well. However, the load operates at 5 volts so we will put a switching regulator circuit in between the battery and the load [4]. The switching regulator will be used as a part of our PCB design. After doing research, it was found that switching regulators can be very sensitive to noise, so we used a separate switching regulator for each output including the Raspberry Pi modules and the ethernet/USB hub along with isolated ground planes for each output. Here is a diagram of the general layout of the power system for our design:

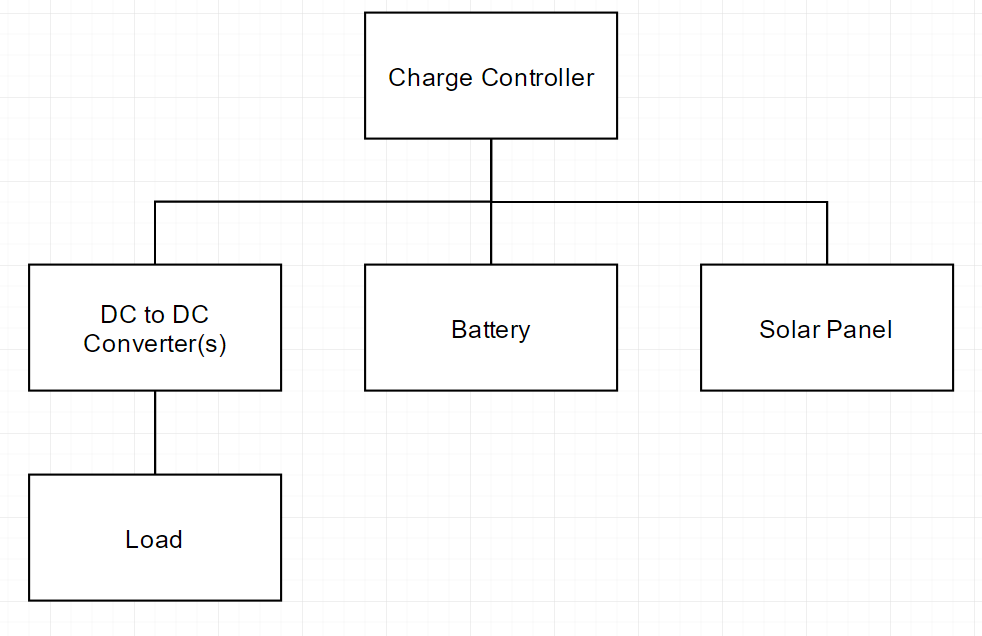


Figure 1 General Layout of Solar Charging System [5]

This design allows for easy charging of our system without having to add various wires running outside of main chassis box through the car. It was also found that the power needed for the raspberry pi modules with the 12 volt to 5 volt to step down converters is much less than what was found when testing the power needed with the AC wall adapter. This means are charging system should perform exceptionally and exceed in power needs for the design. The solar panel is also flexible which also allows us to bend the solar panel to match the contour of the chassis box.

1. *PCB*

*i.Voltage Regulator*

For our first PCB we knew we needed to somehow regulate the voltage coming from the battery to be able to work with our devices. The devices we needed to power are four Raspberry Pi, an ethernet Hub, and a powered USB hub. Each of these devices has an operational voltage of 5V. Both of the hubs have a maximum current rating of around 500mA. The Raspberry Pi each have a maximum current rating of around 2A. Because of this, we wanted to make sure that each regulator could safely pull 2A. Because of this, we decided on designing a PCB using the LM2592 5V set output switching regulator. [6] This device has internal resistors and a feedback loop to maintain 5V. We also knew we needed at least 5 of these, one for each Pi and an extra to be used for the extra hubs. This ensures that each doesn’t pull over the maximum current.

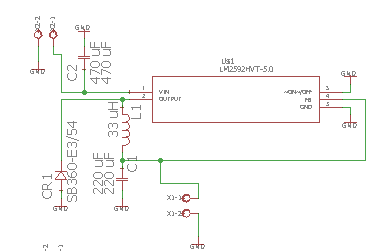


Figure 2 – Voltage Regulator Schematic

Above is the schematic designed in Eagle. Five of these schematics were placed in parallel on one board in the final design. As can be seen, there are no external resistors used, as all resistors are in the LM2592.

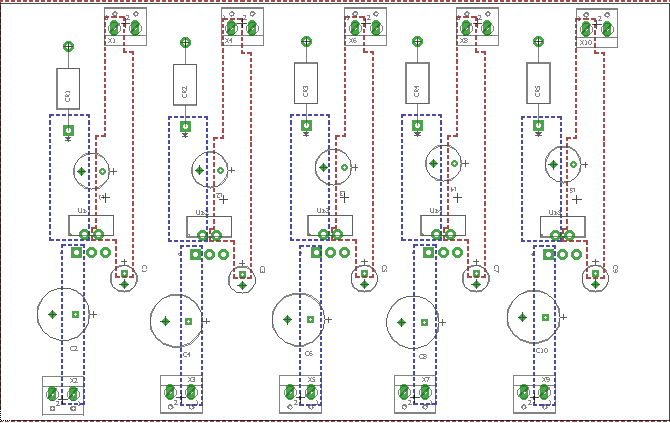


Figure 3 – Voltage Regulator Board Layout

We used planes instead of wires for the entire design. This is useful in designs that have the potential for a lot of current. This makes sure that the circuit doesn’t overheat. When testing the voltage regulators, each gave a steady 4.96-5.06V even when pulling up to 1.3A.

*ii. PWM Controller*

Another PCB we built is a PWM charge controller. A PWM charge controller is a Pulse Width Modulating charge controller. It uses pulse width as a switch for controlling voltage between the panel and the battery. We used Julian Ilett’s design for the PWM controller. [7] This design uses a Arduino nano microcontroller as the brain of the charge controller. The controller measures both the battery voltage, the input, and also the Pulse width to determine when to cut off.

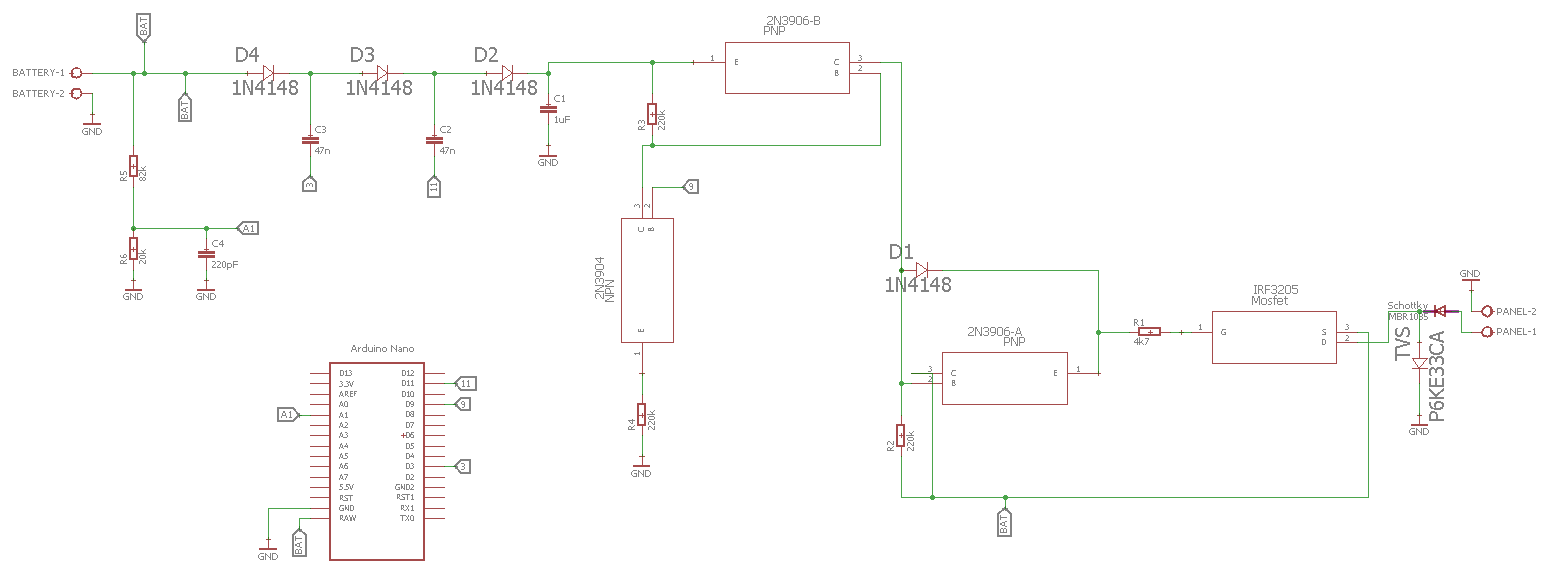


Figure 4 – PWM Schematic

The design seen above uses both MOSFETs and diodes for limiting the flow of current in one direction, from the panel to the battery. This design was fitted onto one small compact board.

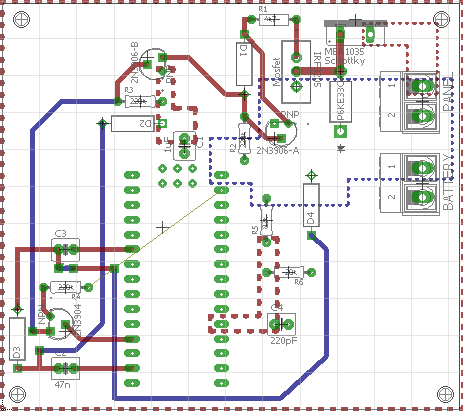


Figure 5 – PWM Board

One small issue when testing the board, we ran into is that one pin was connected on an isolated ground. This was an easy fix with an external wire. Afterwards, our board worked perfectly in cutting off voltage when battery was at determined set point. We also designed another one of these to work between a battery and a load. In this case, the battery will cut off from the load when its voltage drops too low.

*iii. MPPT Controller*

A PCB of an MPPT charge controller was created however inconsistencies in the voltage levels of both the battery and the load terminals lead to the chances of an unsuccessful system and so the PWM design charge controller was used instead of this design. However a significant portion of our time was put into making this charge controller and so we will provide details on making this charge controller. The schematic was designed using a drawing from the Texas Instruments website [8]. Eagle was used to create the following: schematic, component packages, and the board component layout. Below is the board layout used to create the MPPT charge controller.

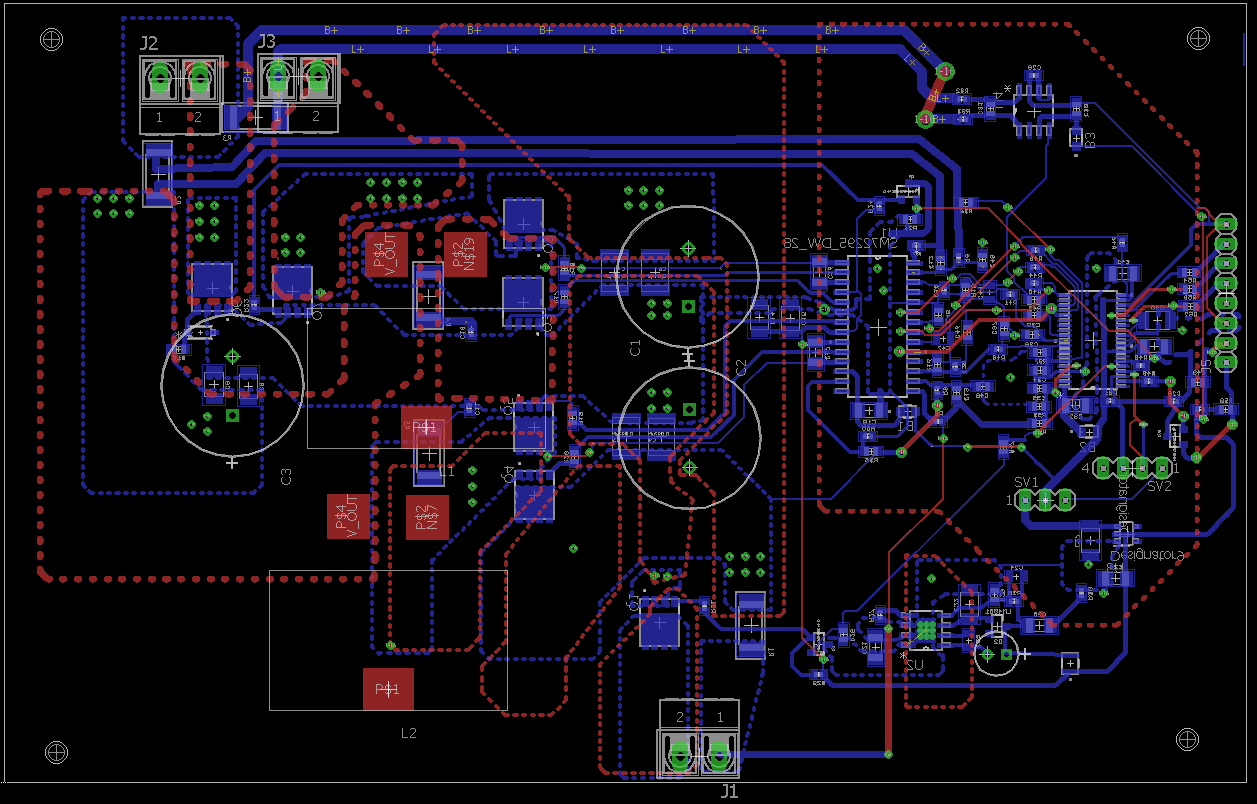


Figure 6 – MPPT Board Layout

Once we received the board and components, each component was soldered onto the board via surface mount. There is an embedded MSP430 that is used to control the charge controller and this was programmed using code from the Texas Instruments website [4]. The code came with a GUI that was installed and an MSP430 Launchpad with GPIO pins was used to connect the charge controller to the GUI. The MPPT charge controller was able to successfully connect to the GUI however various circuit protection problems kept occurring. The GUI can be used to detect the voltage and current levels of both the battery and solar panel. The onboard MSP430 also be programmed to run without the constant use of a GUI.

The MPPT charge controller allows the battery to continue to draw less and less current as the battery nears maximum charge. Also, the solar panel voltage is stepped-down to the battery voltage to allow for maximum charging efficiency. Several months were spent making this charge controller, including time spent soldering components and designing the layout for this charge controller and verifying the circuitry before having it manufactured. Video is used to demonstrate proper usage of this charge controller when it is connected to the respective GUI.

### V. Software

The software development component of the project is comprised of two subsystems. The first subsystem will be the software infrastructure to program the Raspberry Pi. The second subsystem will be the Android mobile application. These two subsystems were developed and tested simultaneously in order to stay on track with project milestones. The programming languages being used in this project were Python for programming the microcontroller units and performing the server-side scripts. PHP was used to integrate the Python and bash scripts and allow the application to communicate with the security system. Java was used for the development of the mobile application

*A. Raspberry Pi Cluster Internal Communication*

The cluster is a system of 4 communicating Raspberry Pis connected via ethernet. At the front is the head pi, which holds primary responsibility for controlling system behavior and the activity of each worker pi. In order to manage the behavior of each of the Raspberry Pis as well as the system function in general, a system of SSH communication over ethernet is set into place. Each worker node communicates to and from the head node. The data being transferred varies between operation flags and commands (such as setting dashcam mode, alert mode, and initiating live stream sequences), to picture files and video data. The most common avenue of control that is utilized, is in the case of the head pi accessing the bash terminal of a worker pi and executing existing python scripts.

Untitled Diagram (8)

Figure 7 - Pi Cluster Communication [9]

Communicating to and from the nodes can be accomplished in most cases by a direct python script call, however often there is an acknowledgement protocol to handle with complex actions. At this point, utilization of *expect* scripts were very helpful in device implementation. The two primary uses of expect scripts in this project is handling SSH permissions and certificates as well as utilizing the GSM module to send text messages, which will be discussed in a later section.

*B. System Operational Modes*

The major operational modes that exist on the device are as follows:

* Dashcam Mode – Each camera will record footage to the pi’s local storage. The video clips will record for a certain amount of time before they end and immediately start to record to a new file. This allows the device to record indefinitely, deleting older footage as storage fills up.
* Alert Mode – In this mode each node will monitor their own PIR sensor and when motion is detected their camera will capture a single photo and send the data to the head pi. At the head pi there exists an IMU sensor which will monitor the minute positional changes of the entire device in order to detect any and all vibrations and tampering. Once these two detection sequences are triggered within a user-defined length of time (default of 5 seconds) a text message alert will be sent to the user, as well as a picture and notification to the mobile android application.
* Livestream Mode – The user may select any camera in the device to view the live footage currently being captured. This will be streamed directly from the device to the mobile android application.
* Standby – This will be the mode where no activity is currently being executed by the device.

A major challenge when designing and implementing the behavior of the device was managing the sequencing and time sensitivity of each task. Management of race-conditions, semaphores, and shared data between 4 separate processors was a hurdle in and of itself. By utilizing a universal configuration object, the head pi was able to ‘know’ the current behavior of each of the worker nodes. This allowed us to implement mode changes seamlessly.

*C. External Communication Process*

A large motivation for us when designing this project was to be able to have an intuitive interaction between the user and the device. The most flexible design pattern was the implementation of a webserver to be hosted by the head raspberry pi which will communicate to our mobile android application through the use of HTTP requests. The LAMP (Linux, Apache, MySQL PHP) stack is installed onto the headpi which hosts its local webserver files into any network interface available to it. Technically the other worker nodes would be able to access the head web server via its ethernet address, however using SSH is a much quicker and direct technique for those uses. Following our pattern of primary python scripts and bash terminal execution methods, the primary control technique used between the head node and the user’s android mobile application is calling php scripts hosted on the web server from the mobile application. These php scripts in most cases call

Untitled Diagram (10)

Figure 8 - External Communication Path [9]

A GSM USB modem was used in our project to give the device text message capabilities. When in alert mode and both detection sequences are triggered, an expect script is run to interface with the modem and send a pre-defined text message. The command “cu” is used from the shell which gives us the capability to initiate a dial in terminal to the device. Once connected, the acknowledgement and command sequence is automated via the expect script by defining the action, number, and message to send.

*D. Streaming and Alerts*

To accomplish our specification requirement of live video streaming to the user, we were able to use two key techniques, VLC player and the android VideoView class. From the raspberry pi’s end, the VLC player plugin was installed to the operating system and direct footage was piped from the “raspivid” command directly through VLC’s media server capability at an outgoing port on the pi’s webserver. Following, the android application listened to the pre-defined IP address and port for a video stream. Due to the process of transferring video data directly through PiCamera -> Raspivid -> VLC media -> webserver -> Android VideoView a moderate delay was present between action and response. For our purposes however, the delay was manageable and expected in such a data translation process.

updated app

Figure 9 - Video Stream

Another aspect of user-device interaction that our device had the capability of managing is detection alerts sent to the mobile application. When the device is presently in alert mode and both detections are triggered, a picture that is captured from the offending pi is transferred and hosted temporarily to the head pi’s web server. The head pi then uses this picture to send an alert notification to the android application via a PHP POST message. Once confronted with a detection alert, the user is given the option to sound an alarm or ignore the alert, as well as the ability to download the photo to the phone’s local storage.

*E. Mobile Application*

The mobile application is the main source of communication between the user and the vehicle security system. The user will be able to interface and control the behaviors of the system through the Android Mobile Application. The application was developed using Android Studio IDE. This IDE allowed the developers to fully implement the app, as well as debug and test the app through the Emulator feature. During all the stages of the development phase, the application was tested on both the Android emulator and physical device.

*i. Client-Server Communication*

The mobile application communicates with the vehicle system by communicating to the Linux, Apache, MySQL and PHP (LAMP) Web Server that is located in the Raspberry Pi. The application initializes the communication by creating a HTTP connection between the web server and the head Raspberry Pi. The HTTP URL Connection class was used as well as the Asynchronous task class to establish the connection. This connection allows the application to send information to the head pi in order to be processed. The mobile application sends PHP calls that the web server processes and executes upon request of the application. The PHP scripts are executed on the head pi and alter the behavior of the security system to comply with the user requirements.

*ii. Main Menu*

The Main Menu window displays all the features that the application possesses. The first two buttons display the operation modes of the security system. The first button allows the user to enable dashcam mode in the system. This mode records videos of the surroundings of the car, stores them in the local storage of each of the pis. These videos are sent to the application for the user to view if desired. The second option enables alert mode. This mode activates the PIR sensors and IMU. These sensors will constantly seek for motion detection or vibration. Once vibration is detected, an alert will be sent to the user in order for him or her to take action by either watching the video stream or sending an alarm to the security system through the mobile application.

The two buttons found in the middle allow the user to see the surroundings of his/her car while being away. The view streams option streams video of any of the four cameras and displays it on the application. This button will take the user to another activity in which he/or she decides which camera to view. The live stream of the camera is displayed once the user selects the desired camera. The dashcam videos option of the application allows the user to browse across the videos that are recording during dashcam mode. The user will have the opportunity to see the video directory of each of the four cameras and will be able to go through them and play any video that is selected.

The last two buttons found in the Main menu activity are for system configurations and for the user to take action in case the car is in danger. The Send Alert button allows the user to take action once he/she sees his/her vehicle in danger. This button sends a PHP call to the web server that will execute the given script and sound an alarm through the USB speakers that are connected to the vehicle security system.

The Settings option in the main menu will provide the user with the opportunity of configuring the dashcam mode settings of the system remotely. The settings configured through the user interface will be recorded and sent to the web server for processing. The head pi will process the configurations and update the system settings accordingly. On this option, the user has the opportunity to select the amount of videos that each camera directory will contain as well as the length per video that the camera will record. The figure below displays a screenshot of the Main Menu Activity of the Vehicle Security Android Application.



Figure 10 - Main-Menu Activity of Mobile Application

*iii. Video Streams*

Once the View Video Streams option is selected on the main menu, a new activity will be presented to the user. This activity will allow the user to select the stream that he/she wants to view. This activity has four buttons that help the user know the location of each of the cameras and give the user the opportunity to pick which side of the car he/she wants to view. The buttons are labeled front, right, rear and left to differentiate the different streams. Once a side is picked. The video stream of the camera will be displayed on a separate activity within the application for the user to view. Once the user decides to go back to the previous page, the stream is stopped and the previous Android Activity is displayed. The figure below shows a screenshot of the Video Streams menu.



Figure 11 - Video Stream Activity of Android Application

### VI. Conclusion

This Senior Design project proved to be a very challenging and rewarding task to complete. Designing and implementing such a complex system gave each of us useful experience in creative problem solving and allowed us to learn about many new subjects that we had not dealt with prior. The culmination of our under-graduate engineering career was put to the test during this projects course.

Not only was the group tested by the difficulties of the direct content and implementation of the project requirements, but the often-overlooked skills such as time-management, project planning, and long-term implementation planning were challenged as we made our progress through our final two semesters.

From the software perspective, we were forced to overcome several challenges that concerned both completing our set tasks (such as being able to configure video clip length and GSM modem communications) but also higher-level system interactions, causing entire system redesigns. There were many points in our project that we were forced to take a step back and reevaluate our plans and expectations for the project as we were met with time and functional restraints. An example of this was the desire to stream video footage across a GSM connection, once the interface was setup, we were confronted with a large loss of bandwidth availability which would not be able to sustain a constant video feed, regardless of any possible compression techniques.

### VII. Acknowledgements

The development of this project was one that was very complicated and involved. The completion would not have been possible without the additional guidance of our professor, Dr. Wei. We would like to thank Dr. Chan for his support when designing the PCBs utilized in this project, as well as Anirudh Pise and Dr. Bataresh for their help troubleshooting the MPPT charge controller design in Dr. Bataresh’s lab. We would also like to thank Adam Hadiji as he came up with the initial concept and design for the project. Adam Hadiji introduced the concept of a solar powered car security system with cameras on each side of a car that would be able to send wireless alerts. Adam Hadiji wanted a security system that could fit inside a box that could be placed on top of a car that was completely independent of the car.

### VIII. Biography

Adam Hadiji is graduating in May 2017 with a degree in electrical engineering from the University of Central Florida. He has strong interest in pursuing a career path in electrical engineering and has accepted an offer Electrical Design Associates and will start when he graduates.

Sean Perraud is a senior student at the University of Central Florida. He is graduating with his Bachelor’s of Science in Electrical Engineering in the Spring of 2017. His interests are in embedded and power systems. After graduating, he hopes to work in either the embedded systems field or power systems field.



Ben Hochstadt is a 22-year old Computer Engineering student graduating Spring 2017 with an interest in embedded systems. Following graduation he’ll be starting his career as an embedded software engineer at Lockheed Martin, MFC.

Gerardo Pineda is a 22-year old student from Venezuela raised in the United States. He is pursuing a Bachelor’s of Science in Computer Engineering. He enjoys working with people and plans to pursue a professional position in software consulting, systems engineering and/or networking.

### IX. References

[1] "Omnivue 360° (ORCA)," Camos-Multimedia, [Online]. Available: http://www.camos-multimedia.com/en/products/360omnivue. [Accessed 30 November 2016].

[2] "Arduino - ArduinoBoardMega2560," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMega2560. [Accessed 07 November 2016].

[3]"Raspberry Pi - Raspberry Pi Hardware Guide requirements | Raspberry Pi Learning Resources," [Online]. Available: https://www.raspberrypi.org/learning/hardware-guide/components/raspberry-pi/. [Accessed 07 November 2016].

[4] "How Much Power Does Raspberry Pi3B Use? How Fast Is It Compared To Pi2B?," RasPi.TV, [Online]. Available: http://raspi.tv/2016/how-much-power-does-raspberry-pi3b-use-how-fast-is-it-compared-to-pi2b. [Accessed 4 November 2016].

[5] "Flowchart Maker and Diagrams Online," Draw.io, [Online]. Available: https://www.draw.io. [Accessed 26 November 2016].

[6] Instruments Incorporated, Texas. "LM2592HV SIMPLE SWITCHER® Power Converter 150 KHz 2A Step-Down Voltage Regulator Check for Samples: LM2592HV."*LM2592HV SIMPLE SWITCHER Power Converter 150 KHz 2A Step-Down Voltage Regulator (Rev. C)* (n.d.): n. pag. Web. 30 Mar. 2017. <http://www.ti.com/lit/ds/snvs075c/snvs075c.pdf>.

[7] Ilett, Julian. "PWM Charge Controller." *256.uk*. N.p., n.d. Web. 06 Apr. 2017. <http://256.uk/>.

[8] "TIDA-00120 Solar MPPT Charge Controller," Texas Instruments, [Online]. Available: http://www.ti.com/tool/TIDA-00120.

[9] "Teach, Learn, and Make with Raspberry Pi." Raspberry Pi. N.p., n.d. Web.