Easy as Pi Vehicle Safety System

Chris Kaberle, Anthony Milani, Mira Yun, and Magdy Ellabidy

Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
{kaberlec, milania, yunm, ellabidym}@wit.edu

Abstract

With limited resources, knowledge, and experiences, our undergraduate classroom can be very dry and theoretical. However, by introducing existing example projects, our undergraduates can be motivated to find more real-world problems interested and solvable. Our Easy as Pi Vehicle Safety System (EaPVSS) is an effort of finding a real-world project and solution for undergraduate students. As one of undergraduate capstone projects, EaPVSS is proposed to increase the overall safety of drivers with off-the-shelf low cost devices. EaPVSS provides three main systems including parallel parking, DVR, and surveillance. EaPVSS shows how undergraduates contribute to the real-world projects and communities by adapting learned knowledge of computer systems and networking technologies.

Keywords – Undergraduate Capstone, Vehicle Safety System, Surveillance, Digital Video Recorder.

I. INTRODUCTION

As of May 1, 2018, new cars sold in the United States are required to have backup cameras installed in them [1]. While some cars sold before then have backup cameras installed, most do not, especially older cars. The introduction of backup cameras provided a huge assistance to drivers who have trouble backing up, for example in cars that are larger or cars that are new to them. Parking is only one of the areas that could use improvement. There are an average of approximately six million car crashes a year in the United States [2]. A person is at fault for most accidents, but proving it after the fact can be difficult. A lot of insurance companies have started issuing dash cams so that drivers can record accidents, which helps drivers prove fault more easily. Car related thefts have reached over half a million a year in the United States [3], and most current anti-theft systems will just make noise or flash lights. These only act as a deterrent and will not be able to help the driver after a theft has occurred.

As one of undergraduate capstone projects, *Easy as Pi Vehicle Safety System (EaPVSS)* is presented in this paper to increase the overall safety of drivers with off-the-shelf low cost devices. *EaPVSS* proves how undergraduates contribute to the real-world projects and communities by adapting learned knowledge of computer systems and networking technologies. By attaching two cameras to the car, *EaPVSS* makes parallel parking easier. One camera is attached to the back to check the distance from the rear of the car. Another camera is installed on the front so the drivers can know exactly how much space they have to work with. Additionally a digital video recorder (DVR)/dashcam functionality is implemented so the driver can have video proof should any be needed in the case of car related incidents. This feature makes drivers feel safer while driving because they know whatever happens will be recorded and can be saved as evidence if needed. Another objective is to add an internal surveillance system for when the car is parked so the drivers can have some peace of mind while being parked in an unfamiliar area or in an area that they feel their car may not be safe [4].

In the following of the paper, we introduce the design of *EaPVSS* in details, following by the implementation and testing of the system. At the end, we conclude the project and present potential future work.

II. EAPVSS

EaPVSS is a lightweight application that improves vehicle safety with affordable, off-the-shelf devices. There are three main physical components to the *EaPVSS*: a Raspberry Pi Model 3 B with a touch screen attached, and two Raspberry Pi Zero W. The first feature we present here is an enclosed wireless local area network (WLAN) so that the Raspberry

Pi 3 B would be able to communicate with the two Raspberry Pi Zero W units mainly used for their attached Raspberry Pi Cameras, as shown in Figure 1. EaPVSS construct a small /29 network that the entore system works off of 3 Raspberry Pis. To allow for communication between the three devices, the Raspberry Pi 3 B was set up as a wireless access point (AP) using a software called HostAPD, or Host Access Point Daemon.

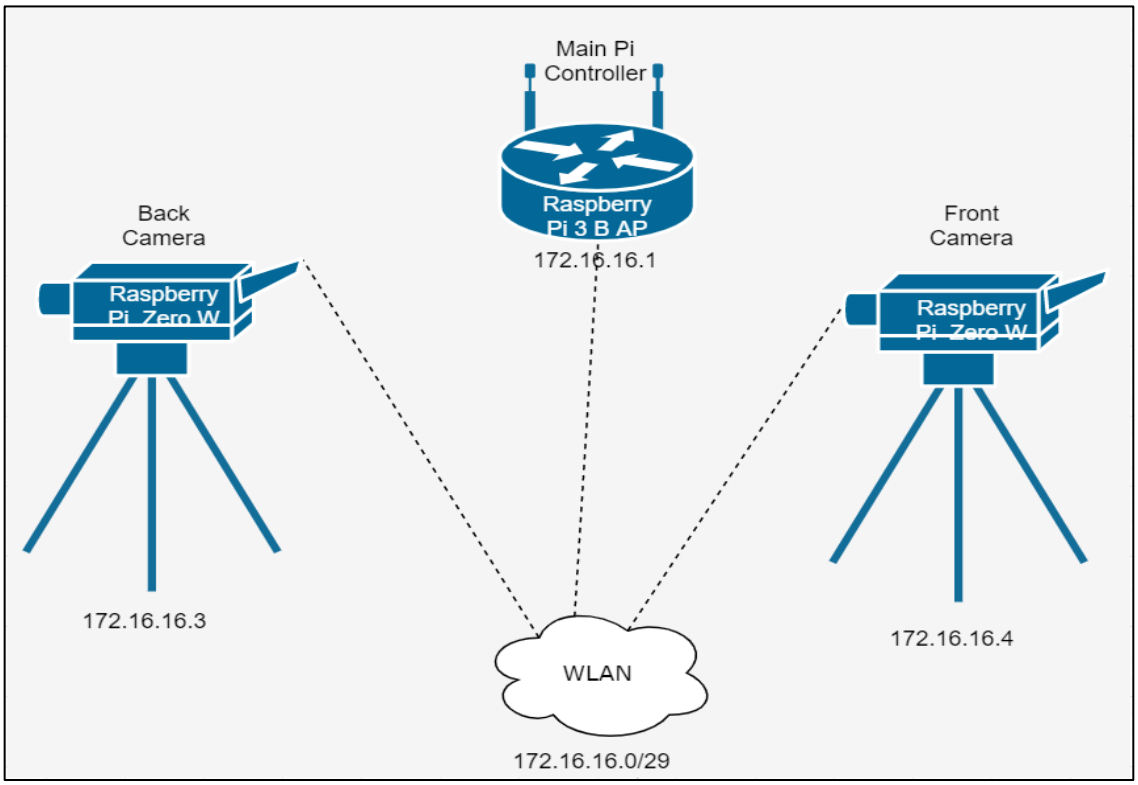

Figure 1: Network diagram for EaPVSS

The WLAN was secured with Wi-Fi Protected Access (WPA2) and the Service Set Identifier (SSID) was set to not broadcast. These security enhancements make it much more difficult for an attacker to capture the data. In addition to this, the subnet mask for the network was set to a Variable Length Subnet Mask (VLSM) of 255.255.255.248, which only allows for five usable addresses within the subnet. Three of these addresses are being used for the Raspberry Pi units, leaving two remaining. All three devices on the network were assigned static IP addresses, and the AP was not set up as a Dynamic Host Configuration Protocol (DHCP) server. This means that an attacker would have to discover the hidden network and guess the IP address information in order to infiltrate the network.

2.1 EaPVSS Parallel Parking System

EaPVSS parking system utilizes cameras at the front and rear of the vehicle as shown in Figure 2. The cameras are Raspberry Pi Zero W units with Raspberry Pi Cameras attached. They communicate back to the Raspberry Pi 3 B unit, which is located on the dashboard of the car. This Raspberry Pi hosts a website that can be accessed on that unit by the vehicle operator. The site takes the data from the web servers hosted on each of the Raspberry Pi Zero W units and combine it into one easy to use site. The case of this unit will have a touch screen that the driver can use to interact with the website. There is a button on the site that allows the user to switch between the two cameras.

Figure 2: Camera Locations on Vehicle

UserspaceVideo4Linux (UV4L) is a large component of the parking feature. UV4L streams real-time video to the website on the web server that will be detailed later. UV4L has a feature to set up a web server to stream the video from the camera, called UV4L-WebRTC. This feature allows a user to view the webpage by browsing to that web server from a web browser and the only thing on the webpage will be the stream. For the video settings, the resolution was set to 640 pixels by 480 pixels. This size is big enough to be clearly viewed on the small Raspberry Pi Touch Screen attached to the Raspberry Pi 3 B. The file format for the video was set to H264. This is a very raw file type that is easily streamable as there is little overhead in the file itself. Other file types such as MP4 or MJPEG were too much for the Raspberry Pi Zeros to stream, but H264 streamed very well on these units. The WebRTC feature had a web-based configuration page, but this would only change the settings until the unit was shut down. For a permanent change in settings, changes had to be made to the UV4L config file.

```
<div id="frontcamera">
    <!--takes the stream data from the front camera and puts it in this div -->
    <iframe src="http://172.16.16.2:8080/stream"></iframe>
    <!--form to run the frontcamsave.php script-->
    <form action="frontcamsave.php" method="post">
        <input type="submit" value="Save Front Camera Video">
    </form>
    <!--button to change the view between the front and back cameras-->
    <button onclick="changeView()">Switch Cameras</button>
</div>
```

Figure 3: UV4L stream embedded into the webpage

In order to transition between the two streams, a web server was needed on the Main Pi Controller in order to be able to switch between the two streams. For this functionality, Apache HTTP Server 2 was installed on the Main Raspberry Pi. An HTML file was created that included Javascript, as well as a CSS file for page formatting. An iframe tag was used in order to pull the data from the Raspberry Pi Zero's stream onto this webpage, as shown in Figure 3. To allow only one stream to be viewed at a time, CSS was used to hide one of the streams, and when the "Switch Cameras" button is clicked, as shown in Figure 4, a JavaScript script is used to change the CSS so that the hidden stream is showing and the viewable stream is hidden.

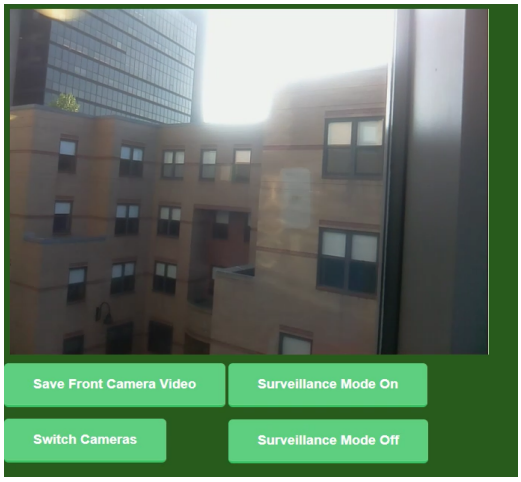

Figure 4: Webpage showing stream from front camera

2.2 EaPVSS Digital Video Recorder

The goal of the dashcam function is to record everyday driving and give drivers a recording of any incidents they may be involved with. This feature consists of a Python script that continuously records for a small period of time, constantly overwriting the last recording, as shown in Figure 5. A while loop constantly runs and records in sixty second intervals. Video records for sixty seconds and saves the video in the H264 file format. The buffer is then cleared and the recording stops, then the loop runs again, overwriting the previously recorded video. This runs until the script is manually closed by saving a video manually using the Save Video button, as shown in Figure 4. This video will be permanently saved instead of being overwritten. These decisions were made in order to save space on the small 16GB SD cards installed on the Raspberry Pi Zero units. This script was set to run on startup using Cron and continuously runs until the unit is shut down. This button runs a PHP script that will execute a shell script. This shell script uses SSH to run a command remotely from the Main Pi Controller to the Pi Zero units, and copies the currently existing video into a different location so that it is permanently saved and not overwritten.

```
import picamera
import io

#sets camera resolution
camera = picamera.PiCamera(resolution=(640, 480))
#Sets the stream object to record in 60 second intervals
stream = picamera.PiCameraCircularIO(camera, seconds=60)
#to make an infitinte loop
i = 1
while  i == 1:
#Starts recording in H264 format
    camera.start_recording(stream, format='h264')
    #Keeps it going in 60 seconds intervals
    camera.wait_recording(60)
    #name of video recording
    stuff_in_string = "dashcam.h264"
    # copys the recording (to overwrite)
    stream.copy_to(stuff_in_string)
    #clears the stream
    stream.clear()
    #stops the camera from recording
    camera.stop_recording()
```

Figure 5: DVR functionality implementation

2.3 EaPVSS Surveillance System

EaPVSS surveillance system is motion sensitive, and takes photos when motion is detected. This provides evidence to the vehicle owner in case the vehicle is broken into, vandalized, or otherwise impacted. This was done with a Python script that constantly runs in the background while the car is in surveillance mode. The code takes a test photo and then compares every photo taken after that to the original, as shown in Figure 6. In the motion function, the test image is captured to compare other images against. In a while loop, another image is taken and compared pixel by pixel to the test image. If a set amount of pixels have changed, then a photo is taken, timestamped and saved.

```
def motion():

    # Get first image
    image1, buffer1 = captureTestImage(cameraSettings, testWidth, testHeight)

    # Reset last capture time
    lastCapture = time.time()

    while (True):

        # Get comparison image
        image2, buffer2 = captureTestImage(cameraSettings, testWidth, testHeight)

        # Count changed pixels
        changedPixels = 0
        takePicture = False

        if (debugMode): #explained before, actually calling here
            debugimage = Image.new("RGB",(testWidth, testHeight))
            debugim = debugimage.load()

        for z in range(0, testAreaCount):
        # = xrange(0,1) with default-values = z will only have the value of 0 =
        #only one scan-area = whole picture
            for x in range(testBorders[z][0][0]-1, testBorders[z][0][1]):
            # = xrange(0,100) with default-values
                for y in range(testBorders[z][1][0]-1, testBorders[z][1][1]):
                # = xrange(0,75) with default-values; testBorders are NOT zero-based,
                #buffer1[x,y] are zero-based (0,0 is top left of image, testWidth-1,testHeight-1
                #is botton right)
                    if (debugMode):
                        debugim[x,y] = buffer2[x,y]
```

Figure 6: Surveillance functionality implementation

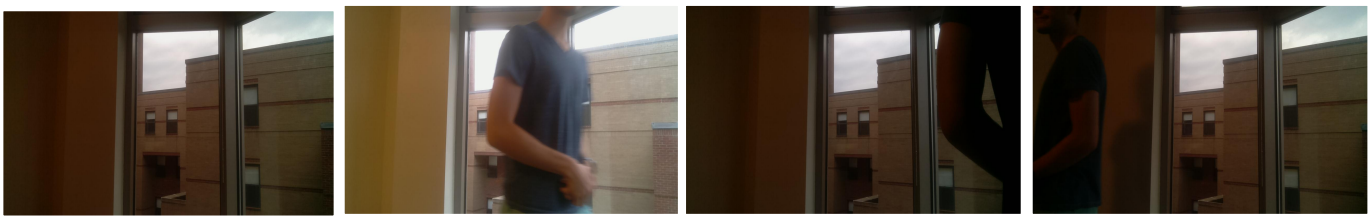

Figure 7: Surveillance Photos Taken by EaPVSS

The user is then able to get the pictures taken in the same way the retrieve videos from the DVR function. In Figure 7, four photos can be seen that were taken by the surveillance system feature. The first photo is the test photo that every other photo is compared to. The others are three photos that were taken that contain different pixels than the original. This difference can be as specific as an entire person in frame, as seen in the second photo, or a shadow of a person in the frame, as seen in the fourth photo.

III. TESTNG AND RESULTS

Throughout the implementation process, each aspect of the project was tested by itself first. To test and prove that the WLAN was successful, each device was checked to make sure they could communicate with each other over the network without being bridged to an outside network. This was done with a series of traceroute and ping commands.

The testing of the parallel parking feature happened in two parts. The first test was to confirm that the cameras worked on their own and there was no physical defects with them. A simple script with some test videos ensured that the Raspberry Pi Zero W with the camera module could both record and save video. This feature was tested again by activating both cameras via the UV4L application to ensure that both cameras could run and transmit video to the main Raspberry Pi at the same time. This test was successful as it was possible to switch between both the front and rear cameras with little to no delay and no loss in quality.

Testing the DVR function again had to be tested twice. The first test was on the individual cameras to ensure that the script would work as intended. The testing for this included recording a stopwatch for one minute and having it save a middle time frame to ensure that it worked like a real dash cam. The camera was set to record for a minute and only saving the video that showed the 30-40 second time frame on the stopwatch, proving that the script was working as intended. The next part of the test was to ensure that the script ran from the application. The same test was ran and provided the same results.

Testing of the surveillance feature was done in a similar, two stage design. The first test included running the surveillance scripts on the cameras. Testing for this included setting the camera up and having a person move into view. The test proved successful by taking a burst of photos when motion was detected catching the person from a couple different angles and saving the photos. This was also tested within the application and completed a similar test to prove the functionality of the feature.

The final test included running all three of the features from the application. The web application was opened like a user would when attempting to park their car and attempted to judge distances and ‘park the cameras’ without hitting any obstructions. After parking, the most recent recording was saved using the “Save Video” button on the application. Finally, the surveillance mode was activated and a person walked in front of a camera to trigger the script to take photos. After some testing internally, users were asked to test the system in its entirety. Users were able to successfully use the application with minimal assistance, and then users were asked to go look at the photos and videos to see what they captured. These tests all proved to be very successful and all three features worked together on one application.

IV. CONCLUSION

In higher education, bringing real-world problems into undergraduate classroom is very challenging. With limited resources, knowledge, and experiences, our undergraduate classroom can be very dry and theoretical. However, by introducing existing example projects, our undergraduates can be motivated to find more real-world problems interested and solvable. Our *EaPVSS* is an effort of finding a real-world project and solution for undergraduate students. *EaPVSS* provides three main systems including parallel parking, DVR, and surveillance. By using off-the-shelf low

cost devices, we presented how to design and implement those systems in detail. Users were able to use *EaPVSS* the way it was designed with little to no guidance.

REFERENCES

[1] Muller, David. “Backup Cameras Are Now on All New U.S.-Spec Vehicles.” Car and Driver, Hearst Magazine Media, 7 Mar. 2019, www.caranddriver.com/news/a20126570/backup-cameras-are-now-on-all-new-us-vehicles/

[2] “Car Accident Statistics in the U.S.: Driver Knowledge.” DriverKnowledge, www.driverknowledge.com/car-accident-statistics/.

[3] Facts + Statistics: Auto Theft.” III, www.iii.org/fact-statistic/facts-statistics-auto-theft.

[4] M. T. Rehman, N. Khan Swati, A. A. Chaudhary and S. Asim Ali Shah, "Live View Car Surveillance System," 2018 3rd International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST), Karachi, Pakistan, 2018, pp. 1-5