Remote Control Canine

Tashler Dane Greene
University of Alabama in Huntsville

Remote Control Canine

by

Tashler Dane Greene

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Diploma
to

The Honors College

of

The University of Alabama in Huntsville

May 2, 2024

Capstone Project Director: Dr. Earl Wells

Tashler Greene	5/5/24
Student	Date
Earl Wells	5/5/24
Project Director	Date
Aleksandar Milenkovic	Digitally signed by Aleksandar Milenkovic Date: 2024.05.05 17:53:12 -05'00'
Department Chair	Date
Sean Lane	Digitally signed by Sean M Lane Date: 2024.05.07 10:28:03 -05'00'
Honors College Dean	Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College Frank Franz Hall
+1 (256) 824-6450 (voice) +1 (256) 824-7339 (fax)

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

Tashler Greene
Student Name (printed)

Tashler Greene
Student Signature

5/5/24
Date

Table of Contents

Abstract

My Honors Capstone project was to create a covert system that can attach to a military working dog and can securely send real-time video feed from the dog's perspective to an Android application used by the handler. The project utilizes a Raspberry Pi Zero 2 W to send the video over Bluetooth connection. This paper outlines the process and challenges encountered during the system's development, detailing configuration settings, testing methodologies, and numerous instances of error correction. Serving as a comprehensive guide, this paper addresses the lack of documentation found in online sources, offering valuable insights for interfacing with legacy technology. It also provides a valuable resource for future projects involving Bluetooth technology or video transmission.

Introduction

Have you ever wanted to create a security camera or know how one works? Have you ever wanted to build a system that can stream live video using cheap parts? The paper discusses the process and challenges encountered while attempting to build a system that can stream live video between a Raspberry Pi Zero 2 W and an Android phone over Bluetooth. This project aimed to enhance a system created for radio frequency transmission of commands between a military working dog handler and their dog by adding the functionality of streaming live video. This paper could serve as a valuable resource for individuals undertaking projects related to video transmission or Bluetooth technology. It could also be of relevance to anyone working on surveillance technology or wireless communication protocols.

Originality and Importance

The originality and importance of the work that I completed for this Honors Capstone project are tied to the product that the work enhances. My Senior Design group's project is Remote Canine Control, where we have created a system that allows a military working dog handler to covertly send commands and receive status information via radio frequency communication from a system attached to a military working dog that can be one hundred meters away and totally obscured. The handler is able to use an Android application to select commands and view status information such as GPS, battery levels, and signal strength.

My Honors Capstone project was to create a system that can attach to a military working dog and send real-time video feed from the dog's perspective to the Android application used by the handler. Both the Senior Design project and my Honors Capstone project are based around building systems that send information in ways that are already available and widely used, but using secure and covert procedures. There are very few, if any, devices that send and receive live video footage in a covert manner via Bluetooth. The work is also important because it could lead to better and more secure communication techniques for the military, and provide more options for military working dog handlers to choose from in the field.

Process

Acquisition of Hardware

The beginning of this project was the acquisition of the hardware. I asked Dr. Emil Jovanov to purchase a Raspberry Pi Zero 2 W, a Raspberry Pi Camera Module 3 - 12MP 120 Degree Wide Angle Lens, and a Raspberry Pi Zero Camera Cable Set by Arducam. Once I received those pieces of hardware, I began researching how to interface with the Raspberry Pi Zero 2 W. I realized that the Raspberry Pi Zero 2 W, which I had asked Dr. Jovanov to purchase, did not include a microSD card, unlike many other models. The microSD card is used to flash the operating system onto the Raspberry Pi, and without the microSD card, it is pretty much impossible to get the Raspberry Pi functioning.

Configuration of Raspberry Pi Zero 2 W

Once I acquired a microSD card and adapter to be able to write to the microSD from my computer, I was able to load the operating system onto the microSD and plug it into the Raspberry Pi. When the Raspberry Pi is given power, it immediately checks to see if there is a microSD plugged in to retrieve the operating system from. This meant that once I plugged the microSD card that had the operating system on it and gave the Raspberry Pi power, the Raspberry Pi would immediately load the operating system and begin working. I learned through repeated attempts that certain features needed to be added to the microSD card before plugging it into the Raspberry Pi. The most important of these features were enabling Secure Shell Protocol or SSH with password protection and providing the Raspberry Pi with the network information of the local network I wanted it to connect to. I had to do this so that the Raspberry Pi would be accessible via SSH on my local network. I did not use a micro USB or a micro HDMI to be able to write to and see the display of the operating system on the Raspberry Pi. Instead, I used SSH to access the Raspberry Pi and enable the Virtual Network Computing (VNC) server. From there, I could use a third party application like RealVNC to see the display of and use the operating system on the Raspberry Pi. Some other features I changed on the microSD before plugging it into the Raspberry Pi were changing the hostname and setting the local settings like the time zone. The hostname change made the Raspberry Pi easier to find on the network because when scanning the local network with nmap, I could search for the name I choose, "dogvideo", instead of the specific IP address of the Raspberry Pi. The local settings change was important because certain functions on the Raspberry Pi depend on the time zone and would not be reliable without the correct information provided to the Raspberry Pi.

Testing the Raspberry Pi Camera Module 3

Once I had the Raspberry Pi Zero 2 W configured and accessible, it was time to test the camera functionality. I connected the Raspberry Pi Camera Module 3 to the Raspberry Pi with one of the camera cables that I requested at the beginning of the project. After the camera module was attached to the microcontroller, I tried many different commands and encountered many issues. Upon initial research, I was led to believe that the command "raspistill" would be my best at testing the camera because the operating system I loaded onto the Raspberry Pi was a Legacy version. The "raspistill" command is an older command dealing with Raspberry Pi cameras. To use it, you need to allow the Legacy Camera to be used in "raspi-config". One weird issue that I never understood was that my command line "raspi-config" allowed me to toggle the use of the Legacy Camera, but my desktop version on the Raspberry Pi did not give me the option. Unfortunately, "raspistill" would only give me errors. I utilized many commands to determine what was preventing me from using "raspistill". One of these commands was "vcgencmd get_camera", which would allow me to see if the Raspberry Pi could see the camera module and if it could access it. After extensive research into why the errors that I was receiving were appearing, I determined that I would not be able to use “raspistill”, but instead tried “libcamera-hello”. Libcamera is a newer library that provides commands that replace the older Raspberry Pi commands. Libcamera is mainly meant to be used on the newer operating systems, but for some reason that was the only method that I could find that would allow me to access and use the connected Raspberry Pi Camera Module 3.

Hotspot Connection

The next step in my process was trying to get the Raspberry Pi to connect to my mobile hotspot so that I could work on the project anywhere, not just on the initial local network that I set it up with. My process for this was powering the Raspberry Pi off, removing the microSD, rewriting the operating system to the microSD with the network information for my mobile hotspot, restarting the Raspberry Pi with the new operating system, and waiting for it to connect. Unfortunately, the Raspberry Pi would never connect to the mobile hotspot. I learned later that there was a faster way to change the network information on the Raspberry Pi, which was by editing the file “/etc/wpa_supplicant/wpa_supplicant.conf” using “sudo nano”. I did attempt to use this method, but never had any luck with it, so I stuck to rewriting the operating system, since I knew that it worked when connecting to my local network.

Corrupt microSD

While attempting to connect the Raspberry Pi to my mobile hotspot, I made the grave error of using Windows Disk Management to delete the content of the microSD that I was using for the Raspberry Pi. I mistakenly deleted all partitions on the microSD, leading to the microSD becoming corrupted and unwritable. I spent time researching how to recover the corrupted microSD card, but after many hours, determined it would be a better use of time to buy a new one to continue my progress on the project.

Camera Issues

AFter setting up the Raspberry Pi with the new microSD card, I tested my system with "libcamera-hello" and, to my surprise, I did not get the same results that I had previously. Now, my Raspberry Pi was not presenting errors to me, but would try to preview the camera's view and would only show a black screen. When attempting "libcamera-vid", it seemed that the camera would flash between a black screen and the same image, even if I moved the camera to a different perspective. I researched this newfound issue, but found no documentation of the issue. I decided that instead of trying to focus on fixing a new issue that had no documentation, I would move on to a different aspect of the project and return to the camera issues at a later date.

Data Transmission

I then shifted my focus to data transmission, a critical component for the project's success. I began researching different video formats and what might be my best bet, but realized that after running into so many issues already, it may make more sense to focus on just sending a string of text between the Raspberry Pi and the Android device. I started searching for example projects online and code that may be similar to the code I would write.

I wrote a program for the Raspberry Pi to act as a Bluetooth server and a program for the Android device to act as a Bluetooth client. The Raspberry Pi would find the already established Bluetooth connection with the Android device and begin continuously sending strings to the Android device. The strings counted from one to ten so that I would be able to see the constant change in strings on the Android application or the lack thereof which would lead me to another issue in that case. Unfortunately, this is where a lot of progress ended. I ran into severe problems with the Bluetooth data transmission.

I used an older Android phone for my project because of convenience and backward compatibility, but this inevitably led to many problems. A friend of mine had an older Android phone that I could borrow and I decided to use that for the project because, theoretically, if I could get the program to run on an older phone, it would work for any Android device that was produced after the phone I was using. This proved to add a challenge to my work because the older Android device runs on Android Lollipop which is Android's fifth iteration of their operating system. To put that into context, the current Android phones are running the thirteenth iteration, so there are a lot of differences. When trying to load my application onto the older phone, I received errors saying the Application Programming Interfaces, APIs, and Software Development Kits, SDKs, for the Android application were too high to run on the older Android device I was choosing to use for my project. This meant that I wrote my code for a newer operating system and it would not work on the older device.

Once I fixed this problem, I loaded the application onto my device and began testing it. The code that would allow me to connect the devices or check the already established connection required a permissions called "BLUETOOTH_CONNECT". When checking if the permission was allowed in my code, I would receive errors saying that the device did not allow the permissions. After looking through the errors I was receiving, I found that this permission was not in existence when Android Lollipop was created, so the device did not know what permission to check and would always return that the permission was not allowed. This put an end to my progress, because I was unable to find a way around enabling "BLUETOOTH_CONNECT", which meant the program would never fully run or receive the incoming text from the Raspberry Pi.

Before I figured out that the issue was my permission check, I spent a lot of time researching how to write the program for the Raspberry Pi differently. I was a lot more confident in my Android application code because I had just recently been programming for an Android application project. I learned a lot about the different ways of transmitting data over Bluetooth, like using pybluez or “tee /dev/rfcomm0”. I even considered writing the Raspberry Pi Bluetooth server in java, but after some more research decided that the coding language did not impact the problems I was running into.

After the many problems that I ran into, I was unable to find a work around for the problem of data transmission. I considered purchasing a cheap, newer Android device, but with the time constraints I was facing, it did not seem like a worthwhile decision. The time spent fixing the many other problems I faced depleted the time that I had and led to an inability to finish the project.

Self-Assessment and Lessons Learned

Considering the resources and time available, I made significant progress on the project. I ran into many issues and with little documentation of these issues or their solutions, it consumed my time fixing them. I would have liked to dedicate more time to the project and am considering pursuing the end result after graduation.

I recommend that anyone working on a similar project or continuing mine be prepared for numerous obstacles. Tons of research upfront will alleviate obstructions and mistakes made, but there are only so many issues that can be predicted and prepared for. I would suggest utilizing online resources and YouTube videos because there is a lot of documentation on small DIY projects that are similar to this one.

Conclusion

While the project is definitely achievable, it presented many challenges and required extensive learning about data transmission and Bluetooth technology. I utilized an exorbitant amount of research and time to make the progress that I did, but this project definitely requires more. In the future, I hope to continue this project and make the final product that was intended.