E-Maxx Mirage

Alexander Cromar, Mike Mercer, Murphy Rickett, and Kyla Kunz
Computer Engineering
University of Utah
Salt Lake City, USA

***Abstract*—A high-performance remote-controlled car has been retrofitted with a camera that streams video footage to a virtual reality headset. A Raspberry Pi interface is integrated into the system for video transmission and remote control. WiFi technology is utilized for high-quality video streaming and radio signals for precise vehicle control. The system creates a realistic driving simulation by integrating a steering wheel and pedals for remote-control.**

***Index Terms*—Wireless Communication, Radio Control, Virtual Reality, Stabilization, Raspberry Pi**

I. Introduction and Motivation

The Traxxas E-Maxx, a preexisting RC car, is the foundation for our radio-controlled virtual reality project. To explore distant environments and create a near in person driving experience, this remote controlled car has been modified as shown in Fig. 1. This is accomplished by having a raspberry pi 4 with a raspberry pi camera affixed to the chaise of the car. The raspberry pi 4 creates a local area network. An immersive and interactive experience has been created by enabling real-time video streaming through the local area network to a virtual reality headset.

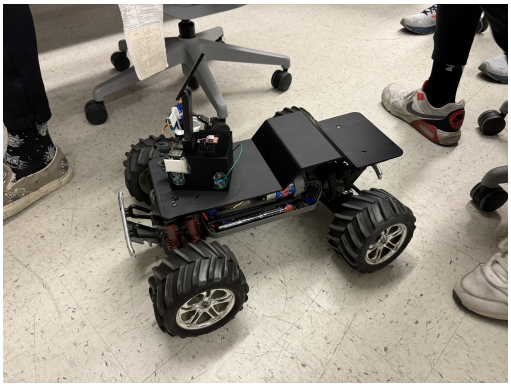

Fig. 1. E-Maxx car.

To help the experience feel realistic, a steering wheel and pedals are integrated for remote control. This not only enhances the user experience but also introduces a dynamic element to the remote control experience. This was accomplished by connecting the car's transmitter to digital potentiometers, which allowed the speed and turning of the car to be controlled through a Raspberry Pi. The Raspberry Pi then connects to the wheel and pedals through USB. Furthermore, the car includes a rotational platform for the camera to expand the degree of view. This approach was designed to mitigate motion sickness and provide users with an immersive experience. This project has established a system that finds applications in entertainment, education, and virtual training. With an inexpensive approach, this makes it affordable for anyone.

II. Related Work

A. Radio Control

A team at the RCC Institute of Information Technology constructed a remote-controlled car that is operated by motion gestures [2]. Some similarities to the E-Maxx Mirage include implementing a distinct control scheme for an RC-controlled car, employing the I2C protocol for various components, and utilizing a Raspberry Pi to interface input, which is subsequently transmitted to the receiver. One of the differences is the use of an encoder and decoder, whereas the E-Maxx Mirage uses the original RC transmitter and receiver, thus there was no need to construct an encoder and decoder. The control method is through an accelerometer, compared to the E-Maxx Mirage implementation which uses direct input devices such as a wheel and pedals. The implementation of the I2C protocol in this project is slightly different because RCC Institute of Information Technology uses I2C to control inputs going into the Raspberry Pi to then be encoded. E-Maxx Mirage differs by using the Raspberry Pi as the I2C master to control other devices.

B. Virtual Reality

A team of three engineers in Taiwan proposed using a virtual reality headset to view footage from an endoscope camera used in minimally invasive surgeries [3]. The camera would follow the movement of the surgeon's head allowing the surgeon to have more control over their surgical tools inside patients. Using a virtual reality headset, also called a head-mounted display (HMD), allows users to experience a full field of view which creates a better depth perception. To record the surgeon's head movements, two lighthouses tracked an HTC Vive headset allowing for high-precision manipulation of the camera inside the patient. A virtual reality headset that uses base stations or lighthouses will be used for E-Maxx Mirage because it is more accurate and easier to implement.

C. Wireless Communication

Raspberry Pi's communicate with other devices wirelessly. A common way to connect a Raspberry Pi to another device is through a local area network (LAN). Several guides exist on how to set up as a LAN [1]. This local area network is used

to transfer data between several different Raspberry Pis and other devices. This implementation uses a dedicated router and network switch as the main host and can be set up to access the outside internet or be completely offline. Raspberry Pis can also connect to other devices through direct radio control through a transceiver. A transceiver typically connects only two devices together, not groups of devices. Radio control modules work great for small amounts of data that need to be sent fast while wireless LANs work great for sharing large amounts of data with one or more devices.

III. E-Maxx Mirage Construction

A. The Car

The E-Maxx's reliability and durability made it a stable platform for our project. It was able to bear the load of the camera mount, enabling a focus on refining radio control methods and enhancing the virtual reality experience, rather than starting from scratch.

B. Radio Control

To both simplify and retain the original control protocol, the original radio transmitter for the car was utilized. This was done by using a Raspberry Pi and two digital potentiometers as well as the original transmitter as indicated in Fig. 2

Fig. 2. Diagram for RC car control flow.

1) Radio Transmitter: The radio transmitter being used utilizes four potentiometers to control the RC car. Steering and acceleration each have two. One for raw input and another for trim to correct imperfections in values. The input potentiometers were replaced while keeping the trim potentiometers for fine-tuning. The rest of the circuitry from the transmitter was reused and placed into a new 3D printed shell for better space utilization. The high voltage is 5V while the low voltage is connected to ground. This leaves the middle voltage to range from 3V to 2V with an initial value of 2.5V. These values are true for both of the control potentiometers.

2) Digital Potentiometers: To control the transmitter using digital signals, two DS3502 digital potentiometers are used. These potentiometers meet the voltage requirements needed for controlling the transmitter. These potentiometers require capacitors to smooth out the voltage as well as pull-up resistors on a custom PCB. In addition, 4 more potentiometers will be used as voltage dividers to shrink the voltage range from 5V to 1V allowing for more granularity.

$$V_{RW} = V_{RL} + \frac{WR}{127}(V_{RH} - V_{RL}) \tag{1}$$

The datasheet provides Equation (1) to calculate the voltage output of the potentiometer. A total of 25 values will determine the output voltage. This means 25 different points of turning and acceleration using these digital potentiometers. Communication between the DS3502 and Raspberry Pi will be done using I2C.

3) Controlling the Potentiometers: The device controlling the two DS3502 devices is a Raspberry Pi 3. This device runs a C program that correctly sets the desired values. This program also interfaces with a USB steering wheel and pedals such as the Logitech G25. This set of wheels and pedals provides a Direct Input protocol to the Raspberry Pi. Different modes are used to limit the speed of the RC car as it is too fast for the user, thus the need for multiple speed modes will be necessary. The custom-made PCB used to interface with all potentiometers and transmitter can be seen in Fig. 3

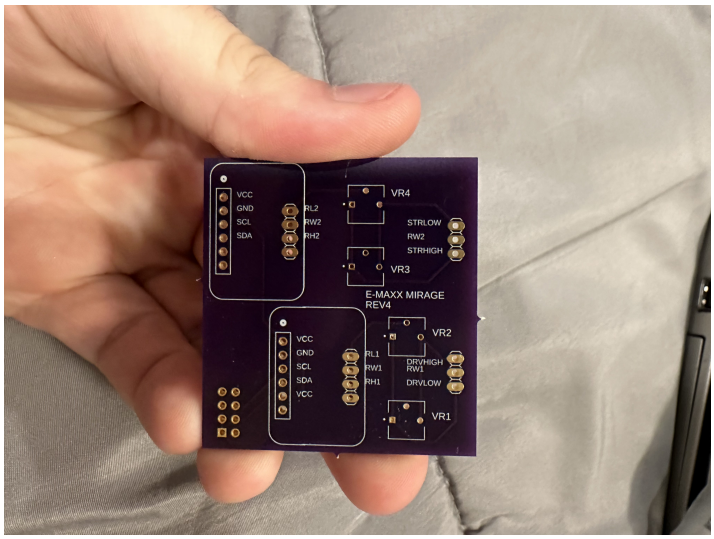

Fig. 3. Custom-made PCB used to interface with potentiometers, Raspberry Pi, and transmitter.

C. Force Feedback

The Logitech wheel used supports force feedback events to give the user an extra level of immersion. Four different force feedback events are used to provide the user with a more realistic driving experience. The first effect is a damping effect which provides resistance to the user. The second effect is a spring effect. This effect only activates when the user is driving either forward or backward and attempts to push

the wheel back to the midpoint. This spring effect has an intensity proportional to the speed of the car. The last two effects are rumble effects that happen when the user changes speed modes.

1) Enclosure: To contain all the necessary components a custom made 3D printed back plate was constructed as seen in Fig. 4. This enclosure housed the Raspberry Pi 3 with the custom-made board with potentiometers interfacing with the GPIO pins. The enclosure also holds a 12V to 5V USB converter so that the Raspberry Pi can be powered by the same source that powers the transmitter.

Fig. 4. Custom made 3D printed enclosure.

D. Wireless Transmission

The entire user experience is based on moving the camera and viewing its footage. The user can control the car effectively, by making decisions based on the car's current soundings. The camera can stream at 90 frames per second (FPS) preventing motion sickness. Fig. 5 shows a picture of the final camera module that is used.

Fig. 5. Raspberry Pi Camera Module 3

1) Support: The Raspberry Pi official camera module is capable of recording video in 120 FPS at 1536 × 864, (admittedly, a rather odd resolution). Several libraries already exist for this camera module, written in several different programming languages. The primary package used is the picamera2 module written in Python.

2) Encoding: The picamera2 module can encode raw footage from the Raspberry Pi Camera Module 3. The H264 encoder is used, as it supports the resolution and frame rate needed. The H264 encoder is implemented using the Raspberry Pi's built-in hardware, supporting 1080p at 90 FPS. However, we opted to use 720p. The command that implemented this on the Raspberry Pi is:

```
rpicam-vid --info-text "" --level 4.2
    --framerate 90 --width 1280 --height 720
    --denoise cdn_off -t 0 --inline --listen
    -o tcp://0.0.0.0:8000
```

Listing 1. Sample Command Line Code

This allows the camera feed to stream at 90 FPS in 720p. The stream is sent over WiFi using a TCP (Transport Control Protocol) connection.

3) Reception: The TCP connection is received by a desktop computer over a local area network WiFi connection. This allows for high bandwidth connection so frames are not lost and head rotation can be sent back across the same connection. Upon reception, the stream is decoded by FFmpeg, an open-source library for video transcoding, into a unity application that displayed the image in a virtual environment.

4) Moving the Camera: Moving the camera to mimic the user's head movement is done by two motors. A stepper motor is used to move the yaw of the camera and a servo motor is used to move the pitch of the motor. Roll is not used because the camera needs to stay flat and level with the ground to prevent motion sickness. First, the user's head movements are captured in unity, only the rotation vector, not position. Then the two rotation values, the x and y, are clamped between 0 and 360 degrees. These two values are sent over a separate TCP connection to the Raspberry Pi. In this case, the computer running Unity is the server and the Raspberry Pi is the client. The rotation values are received by the Raspberry Pi and the servo and stepper motor are moved accordingly. The Raspberry Pi is able to directly control the servo, so no external driver is needed, just a simple PWM output. To control the stepper motor, an A4989 driver is used. Three pins are connected from the driver to the Raspberry Pi, a connection for stepping one step, a connection for controlling the direction of the step, and a ground connection. Using a stepper motor was a poor decision due to its massive current draw which is discussed further in the next section.

E. Power

Powering the camera system that is mounted to the car proved to be difficult. A mobile device power bank was first used. The specs of the power bank seemed sufficient but when put to use, the internal circuitry limited the max current draw.

The Raspberry Pi kept resetting due to insufficient power and the motors didn't work at all with the power bank. Next, lithium-ion battery cells were used. These were extracted from old laptop batteries (CR18650). A buck-boost converter was connected directly to the cells which worked great until the cells were overdrawn and dipped below 1.5 volts making them useless. There was also no way to efficiently charge the cells using this setup. In the brief moment that it worked, everything ran great, the motors, the Pi, and the camera. The buck-boost converter was set a little higher than 5 volts, 5.2 to be precise, to give the pi just a little extra power to handle sudden current spikes. The external power for the stepper motor was connected directly to the cells. The next solution was installing a protective battery management system between the devices and the power source. The cheap 4-cell lithium-ion USB C charger that was used did not have power-over-draw protection. Another 4 batteries were killed. The final solution is a slightly higher quality BMS board that has cell balancing and overdraw protection. This works great.

F. Virtual Reality

The Oculus Rift, depicted in Fig. 6, was selected as the virtual reality headset for this project. Its compatibility with the Windows VR platform facilitated the development and streamlined the testing process of the virtual environment. A significant challenge encountered during the development was the conversion of the TCP data stream into a format compatible with the virtual environment. To address this, a C++ plugin was developed to interface Unity with FFmpeg, as Unity is based on C# while FFmpeg is implemented in C. Unity's support for native C++ plugins enabled the creation of an intermediary layer, which encapsulated the C-based FFmpeg functions. This layer is responsible for decoding the incoming image stream and rendering it in real time within the virtual environment.

Fig. 6. Oculus Rift

G. Car Mount

The final step for testing this project was to integrate everything together. This was done by using Shaper3D to create a 3D design that could both hold the camera and be placed on top of the car. We needed the design to be thick enough to hold the weight of the camera and the motors that move the camera, but thin enough to be able to place the pins over the car mount to give it more stabilization and hold it in place. Through this iterative process, the carriage design was refined and optimized by successive prototyping, ensuring that the camera could be mounted securely to the car. Notably, this design is mounted atop the E-Maxx undercarriage shown in Fig. 1, with no planned alterations to the current undercarriage. We were able to create a carriage that can be easily modified and attached to any type of remote control car, shown in Fig. 7.

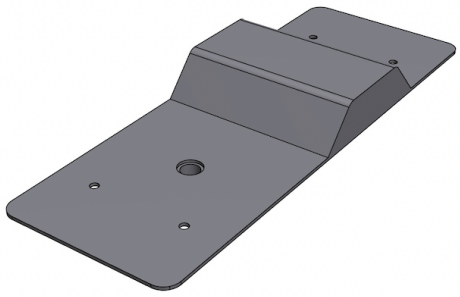

Fig. 7. Car Mount

H. Demonstration

During the demo, the baseline requirements were successfully proven. The car drove smoothly, controlled by the Xbox steering wheel and pedals integrated into the transmitter. The camera remained securely mounted, providing a stable feed. The VR headset connected to the camera, allowing a clear view of the car's path. Overall, the car was easy to use and worked as expected.

IV. Conclusion

A remote control car add-on has been developed to capture live footage from an HD camera and transmit it back to the control station. The control station includes digital potentiometers that allow the connection of the RC car's original transmitter to a Raspberry Pi which is then controlled by an X-input controller. A wireless 2.4GHz radio capable of sending packets at 12,144 kbps is used for transmitting video footage live from the camera to the control station. At the control station, live camera footage is received and displayed in the VR headset, creating an immersive experience.

Acknowledgment

We acknowledge the University of Utah and Ken Stevens for the help and resources provided for this project. The guidance and instruction were instrumental in helping us achieve our final result.

References

[1] J. W. Jolles. Broad-scale applications of the raspberry pi: A review and guide for biologists. 2021.

[2] S. K. Pal, A. Bano, A. Jana, M. Kundu, and A. Ghosh. Gesture controlled robot-car using raspberry pi. In C. K. Chanda, J. R. Szymanski, A. Sikander, P. K. Mondal, and D. Acharjee, editors, *Advanced Energy and Control Systems*, pages 251–259, Singapore, 2022. Springer Nature Singapore.

[3] K. Zinchenko, O. Komarov, and K.-T. Song. Virtual reality control of a robotic camera holder for minimally invasive surgery. In *2017 11th Asian Control Conference (ASCC)*, pages 970–975.