**Intro**

For my final project, I used a Raspberry PI along with sensors, actuators, and accessories to create a website that controls a Raspberry PI Camera and a Servo motor. My Raspberry PI 3B+ runs Raspberry PI OS (previously called Raspbian), and my project code was written in Python 3. My code folder also contains HTML and CSS for my web interface. I connected my Raspberry PI to a solderless breadboard using a 26-pin ribbon cable and T-Cobbler breakout, used to ease GPIO pin access. For sensors, I used a Raspberry PI Camera Module V2. For actuators, I used a SG90 Servo motor positioned next to an automatic dice roller. During early iterations of the project, I also used a four-by-four membrane keypad and three LEDs which were all phased out for the final implementation.

**Proposal Recap**

“I would like to create a networked dice roller with camera capabilities. The idea behind this project comes from one of my favorite hobbies, playing Dungeons and Dragons with my friends. While most of the game is played with pencils, paper, and your imagination, the most fun accessories are the dice. Players buy dice in their favorite colors and grow superstitious about “lucky” dice. Some players even spend $50-$100 dollars buying dice made from hand-carved woods, gemstones, or metals. We bring a few sets of dice with us when we play at friends’ houses, usually our newest sets or a set that thematically matches the character we are playing. If this project goes according to plan, I will always have my favorite dice with me.”

**Design and Implementation**

Diagram, schematic

Description automatically generated

(Circuit diagram, image also included separately in project folder)

My project design centers around controlling a Servo motor and viewing a video stream over the internet. As suggested in class, I implemented this project in three successive stages of complexity.

Stage 1: Controlling the Servo

To begin, I recycled code from my first project to set up the Servo control. I programmed a 4x4 membrane keyboard to control the Servo and LEDs. During this stage, I ran into my first major hurdle in the Servo being capable of pressing the button of the automatic dice roller, more details can be found in the difficulties section at the end of this report.

Stage 2: Video Feed

Once the Servo was set up and capable of activating the automatic dice roller, the next step was the video feed. At this stage, I came across a Python library for streaming a Raspberry Pi Camera over the internet through a Flask server. After downloading Flask and learning the basics, I modified a version of this library to set up my video feed (<https://github.com/miguelgrinberg/flask-video-streaming>.)

Stage 3: Flask Server

This final stage is where the bulk of the project work was done. During stage 2 I had learned how to use Flask to stream my video feed and found that Flask was going to be an ideal utility for my Servo control as well. Flask ended up serving two purposes for my project. First, it served my website which was a convenient feature that saved me from having to use Apache, NGINX, or the less reliable built-in Python SimpleHTTPServer. Second, and more importantly, Flask acted as the interface between by front-end HTML and my back-end Python server scripts. Looking at the last line of the HTML below, we see that a button is defined with href “/roll/70”

Text

Description automatically generated

Next, in the Flask interface code below we see that it deconstructs this route into two variables “/<servo>” which is given in HTML as “/roll” and “/<angle>” which is given as “/70” After pulling these two variables out of the path, the interface called a server side script by concatenating the variables onto an os.system script call, as seen in the last line of diceRoller.py below.

Text

Description automatically generated

Finally, the snippet below of angleServoCtrl.py contains the function being called on the server.

Text

Description automatically generated

This design of HTML -> Flask -> Python script is also used for the video feed, as can be seen in the HTML screenshot above. In this case, the route “/video\_feed” communicates with camera\_pi.py to get a continuous stream of images from the Pi Camera for the video feed.

With these two pieces set up, my project consists of starting the Flask server from diceRoller.py which serves my index.html on the Raspberry Pi’s IP address over port 80. When I visit this site from another computer on my network, I can see a continuous video feed from the PI camera. If I then press the “roll” button on the site, the servo presses down on the automatic dice roller, and I can view the rolling and results live.

**Wireshark Analysis**

Below I have summarized several key frames of a Wireshark capture I used to analyze my server’s network traffic. First, you can see in frame 63553 that my server with IP 192.168.1.135 is sending my HTML data to my desktop computer on destination IP 192.168.1.104. The 31 lines returned match my index.html file.

A picture containing text

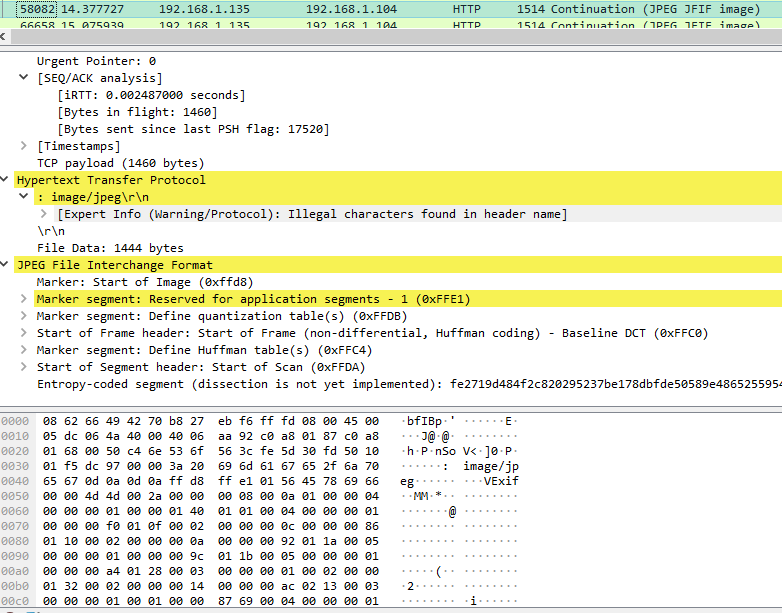
Description automatically generated

Next, frame 24164 shows the GET command that requests the video feed. This is the request that Flask uses to communicate with pi\_camera.py on the server to retrieve the image data from the camera.

A picture containing table

Description automatically generated

Next, we see in frame 58082 a case of JPEG image data being interpreted. Interestingly, we can see reference to the Huffman tables, which we learned in CSCI 484 is used to compress and decompress JPEG images.



Given the fast and plentiful image data being send in the video feed, we also see some malformed packets, as illustrated in frame 72328 below. Based on what I can see in the rest of the capture, “1514 Continuation” suggests this is part of the video stream. These only comprise a small portion of the tens of thousands of video feed packets.

Graphical user interface, text, application

Description automatically generated

Finally, we see in frame 51271 a GET command of /roll/70 which represents me pressing the “roll” button on the website. As explained above, this GET specifies the name of the Servo “roll” and the angle we should set it to “70” degrees.

A picture containing text

Description automatically generated

**Difficulties, Lessons Learned, and Extensions**

The greatest challenges of this project were mechanical rather than programmatic. The actuation force required to press the button on my automatic dice roller was greater than I anticipated. When I originally built the housing out of cardboard, the Servo pressing the button generated so much force that the Servo would not stay in one place. I had to switch to constructing the housing out of some old Legos I had. Even after switching to Legos, the Servo moved so forcefully that it would pop the Legos apart. Ultimately, I was able to solve this issue by super gluing several sections of Legos together. The other challenge I ran into was that the Raspberry PI camera’s ribbon cable is much shorter than I thought it would be. This forced me to place the camera so close to the dice roller that the image had a hard time picking up ambient light and focusing. I was able to solve this issue by shining a flashlight onto the dice rolling tray, but a better solution would be to get a ribbon cable extender.

I learned quiet a bit from this project. I had to refresh myself on writing HTML and CSS, learn how to use Flask, and construct an effective housing unit for the machine. After working with Flask, I really enjoyed it, and will be exploring other ways to use it for future projects. If I were to extend this project, I would add a second Servo to control the angle of the camera. Due to the short cable length on the camera, there were a few times when the dice were at a certain angle or position on the roll tray that made reading them difficult. I added those extension examples to my code to illustrate what I would need to do to implement this. Additionally, if I wanted to make the machine itself smaller, I could create my own dice roller. I used an automatic dice roller because I already owned it, and my emphasis on this project was programing rather than mechanic construction. However, I could improve the mechanical design by throwing out the Servo altogether and using a motor that is capable of continuous motion like a step motor. If using a motor like this, I would position it below a rolling tray and spin the tray by controlling the spinning motor directly. Finally, I plan to forward port 80 on my router to my Raspberry PI so that I can access my website outside of my local area network. I did not do this during my demo to not expose my public IP unnecessarily. Ultimately, I am glad I was able to design a machine and web interface that accomplished all the goals I set in my project proposal, and I am looking forward to using my machine during my next Dungeons and Dragons session.