EE250 Final Project

Inferno Tower

Description:

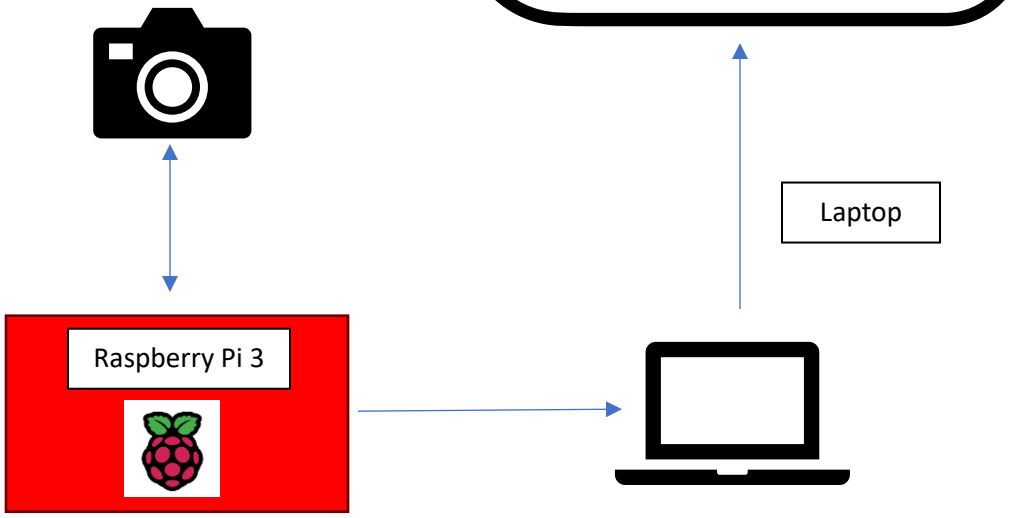
The goal of our project was to provide a home protection system by utilizing the YOLO detection system to determine if a human is in the camera frame. If a human is present, then it takes an image of the human present in the camera frame and posts it to our website for all to see.

The process starts with the camera taking a stream of images that the raspberry pi then encodes the JPEG into a base64 format. There are two parts to the encoding process. The CV2 encode is crucial because it compresses the JPEG into a size that can be sent. Then encode the JPEG image data into a base64 string, so the image can be safely transmitted over MQTT.

Then the Laptop listens for the MQTT channel and AJAX fetch request every half second to fill up a queue. The format of information in the queue is the AJAX fetch request along with a time stamp. Reading from the queue, an image is decoded into a byte array, converted into a numpy array, and then used to create an image object. These image objects are placed into another queue that is fed to OpenCV's YOLO model to analyze if a human is in frame. If a human is not detected, the next image is scanned. If a human is found, then the image is once again compressed using the same process and sent to our webpage. To show the information on the webpage, we switch to JavaScript to fetch the chosen image and add it to the page.

Raspberry Pi Camera :

The Raspberry Pi camera module is a small electronic device that connects to the Raspberry Pi's camera port, enabling the Raspberry Pi to capture images and video.



The Raspberry Pi runs a python script that uses the Raspberry Pi camera to take an image every ~.18 seconds and send it to a Mosquitto MQTT server. This Mosquitto MQTT server is hosted on the Raspberry Pi 3.

We use a laptop to read in the images taken by the Raspberry Pi camera module by reading in from the MQTT server, process the information using YOLO, and then produce results shown on the Webpage.

Conclusion:

The project was immensely enjoyable as it allowed us to apply the knowledge we had acquired throughout the semester and further expand upon it. Our primary challenge centered around transmitting images via an MQTT server. Initially, we opted for two different free MQTT servers; however, the bandwidth provided was insufficient for sending images of reasonable quality consistently. To overcome this, we decided to set up our own Mosquitto server on a Raspberry Pi 3, which significantly improved both the transmission lag and image quality.

The subsequent challenge involved efficiently processing the images on a laptop using the YOLO framework. To achieve real-time analysis, it was crucial to minimize the queue of images awaiting processing. Consequently, we reduced the frequency of image capture, allowing the system to maintain a manageable queue and enhance the 'listener' performance. This adjustment was essential for balancing the system's ability to process images swiftly and accurately.