Final Project: Turret target system

EECS 3215

Michael Angelo Masangkay
219005867
Mixhael
Submitted: December 12 2023

Executive summary:

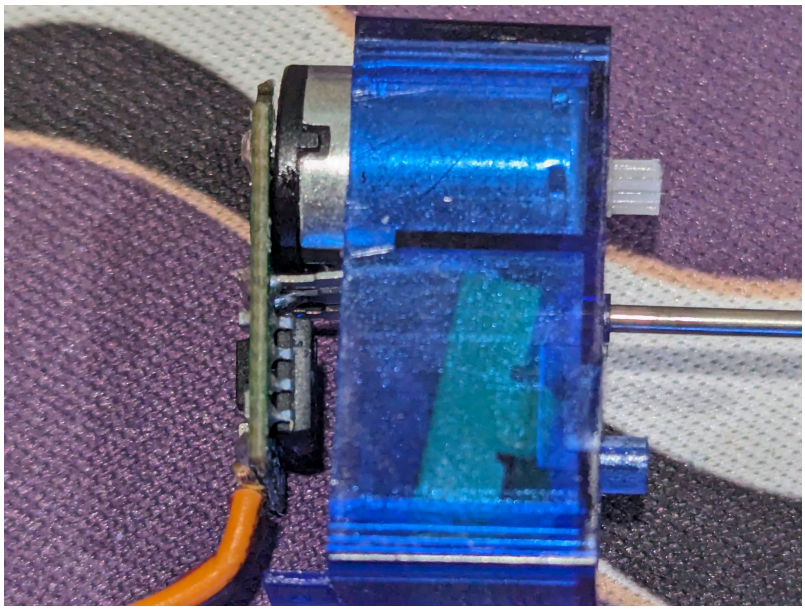
Objectives

- Create something that resembles a turret
- Servo spins 360 degrees
- Controller servo with arduino
- Finding cars in the image and displaying green square
- Setting up camera
- Video feed to pycharm
- Camera controlling servo
- Camera object detects car

Methods:

Servo Control

To make the servo motor spin 360 degrees, I had to take apart the motor and make sure that the potentiometer is ignored and not affecting how the servo motor spins.



Green part is the potentiometer, and as you can see it is no longer clipped on and doesn't affect the servo motor.

I've noticed now that the potentiometer is no longer in effect, that you can no longer control where the servo motor will spin to. For example a normal servo motor, with the command "servo.write(45)", would rotate the servo motor to point 45 degrees. Now that the potentiometer is no longer in affect, this function doesn't work anymore. I've noticed that as I set the write angle to approach 90, it slows down and after 90 it will spin in the opposite direction. Which I learned was because of the PWM in the Arduino. The Ardruino uses a 20 ms pulse cycle and on the standard servo a 1.5ms pulse would correspond to 90 degrees, and shorter pulses would spin the servo in one direction and longer pulses would spin the servo in the other direction.

Learning this was helpful to now help control how fast the servo would spin and the direction of control

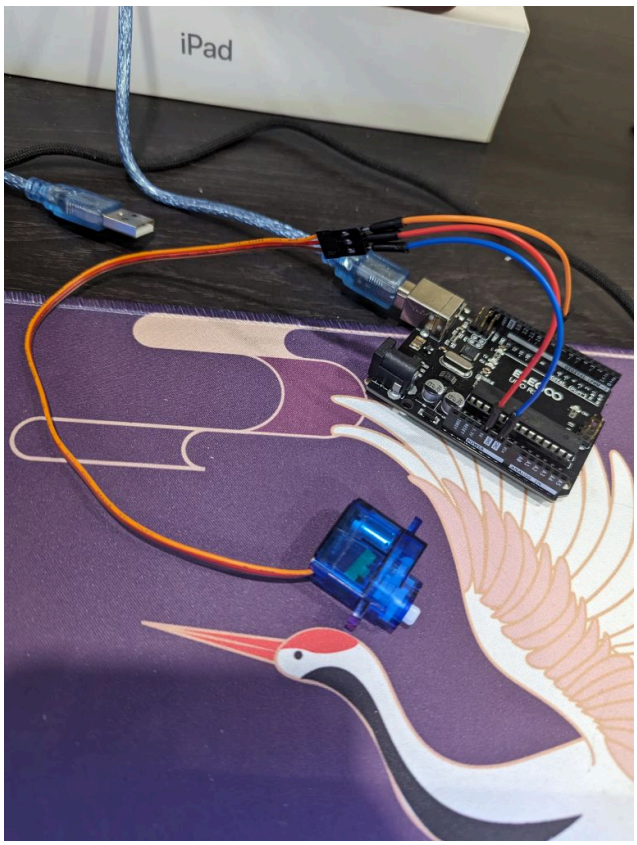
Controlling servo with Arduino:

To control the servo with an Arduino I had to connect it with 3 wires to the 5v, ground and digital 9 pins. Then after that I just had to connect the Arduino IDE to the correct port, com4 and then upload the following code:

```
#include <Servo.h>
Servo myservo;
void setup() {

    Serial.begin(115200);
    myservo.attach(9);
```

This code uses the built in Servo library. I create a servo called myservo and in the setup, i set the baud rate to 115200 for fast transmission. After that we attach the myservo to pin 9, which



As you can see red is on the 5 volt, blue on ground and orange on digital pin 9.

Camera Setup

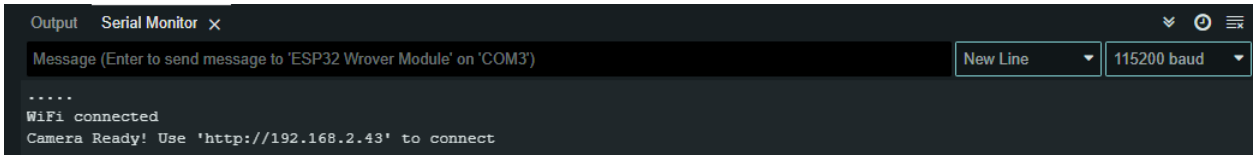
To connect the ESP 32 cam via wifi to my computer, I just used the open source CameraWebServer example which was built into IDE, but with a few adjustments.

First I had to change the model to the correct model, which is the ESP32 wrover cam

```
#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
```

Then I had to enter my WiFi credentials so the Camera can actually access the wifi, after that I kept the given code the same. After that I had to check the serial monitor on the IDE to get the address to access the camera.

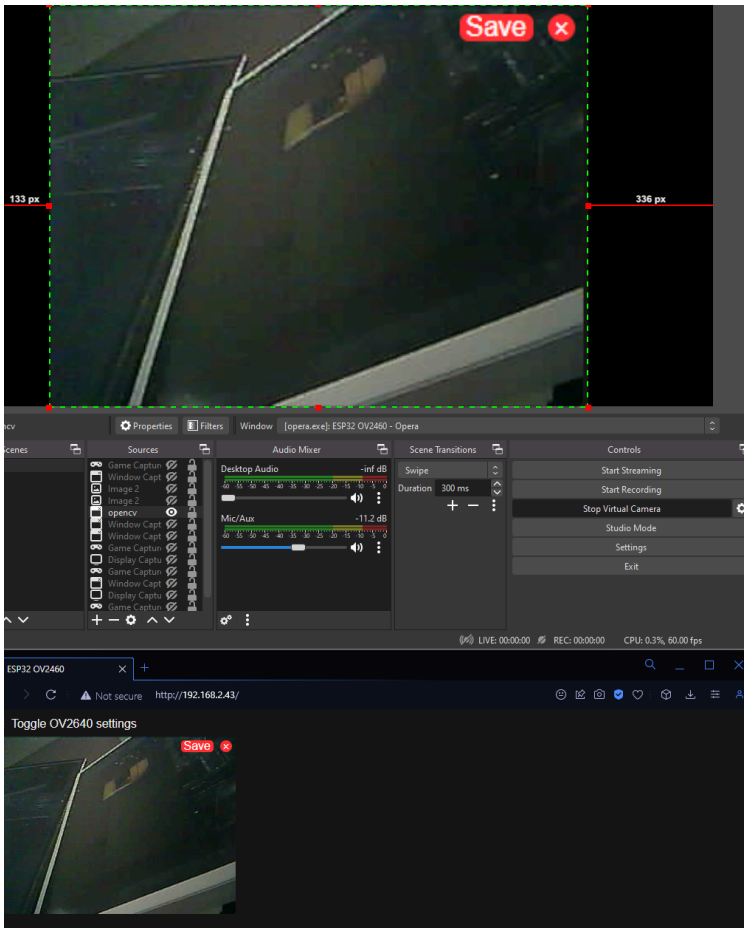
After uploading the code



This is what you see, and you can access the video feed by putting the address above into your browser. Something I learned the that is important to know is that your device has to be connected to the same wifi as the camera, or else you won't be able to see the feed. This makes sense as if it was accessible by the whole internet, anyone connected to the internet could access your camera from anywhere around the world. So this is a safer option.

Connecting the video feed to Pycharm

I had some trouble connecting the video feed from the website because connecting to the website directly doesn't work, so I had the idea of using OBS, which is a recording software to work as a virtual camera and set the video feed of that to the video feed of OBS. And because I can set it up as a virtual webcam, I am able to use the opencv cv2 video capture function which easily gets the video feed.



Here is what it looks like, on the top is the OBS window and on the bottom is the browser with the camera address. As you can see I cropped out all the other things in the window, and only kept the videofeed

```
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()
```

This will access the OBS feed, because I have no other webcams, OBS counts as webcam 0.

Object Detection

For this I utilized more of opencv, specifically using a built in function called CascadeClassifier which is a well documented object detection method. All I had to do was enter an xml file of what I wanted, which in this case was cars.

```
car_cascade = cv2.CascadeClassifier("A:\haarcascade_car.xml")
```

Next I had used cap.frame to get the frame from the OBS videofeed.

```
ret, frame = cap.read()
```

Here I got ret and the frame. Ret is just a boolean checker whether or not there was a return, so if you get a frame or not, and frame is the frame that was returned.

Then I had to grayscale the frame using this command

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

And then got the object detection using the detectMultiScale method of opencv, with the parameters of the gray frame, the scale factor, minNeighbors and the minsize

So the gray frame is the frame that will be searched for the image, the scale factor compensates for objects that might appear smaller or larger depending on the distance of the camera, and I set this value to 1.1,which would enable it to detect smaller objects. The minNeighbors parameter tries to ensure that there are a lesser chance of getting false positives. It works by seeing how many rectangles are there around a detected object, and I set this value to 5. Next is the minSize, which sets the minimum object size, which also helps with not detecting really small objects that are irrelevant, and I set this value to (25, 25).

```
cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(25, 25))
```

After this all I had to do was loop through cars, and print the rectangle.
To do this I used a for loop.

```
for (x, y, w, h) in cars:
```

For each iteration of the loop, it gets its x value, y value, width and height.

To print the rectangle, I used another cv2 function called .rectangle. I fed it the

```
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

So we gave it the gray frame, (x, y) which would represent the top left corner, (x+w, y+h) which is the bottom right corner, with x+w giving x value of the right edge and y+h giving the y value of the bottom edge. 0, 255, 0 sets the color of the rectangle in RGB, which in this case is green. Lastly the 2 represents the thickness of the rectangle.

Lastly we have the system to detect where the object is and move the servo accordingly.

So whenever a rectangle from the above code is made, I check where the x value of the object is, and compare with the bounds of the frame.

```
center_area = (190, 320)

if center_area[0] < x < center_area[1]:
   counter_stop += 1
   counter_goright = 0
   counter_goleft = 0
   if counter_stop >= 15:
       print("wrote stop")
       send_command("stop")
       counter_goright = 0
       counter_stop = 0
       counter_goleft = 0
```

Here I set the center area to x values of 190 and 320, so if the rectangles x value is in between these values, it will count once. As you can see if another counter is incremented instead of this stop counter, it will reset the other counters to 0, ensuring the object seen is actually in the center frame and not left or right to it. If this counter reaches 15, we count is as an actual good object detection and send the stop command.

```
def send_command(command):
   arduino.write(command.encode('utf-8'))
   time.sleep(0.1)
   print(f"Sent command: {command}")
```

This short method sends the command to the arduino using serial communication.

Serial Communication

To set up the serial communication I used this

```
arduino = serial.Serial('com4', baudrate=115200, timeout=0.1)
```

It connects to com4, with a baud rate of 115200, which is important to match the arduino's baud rate. With this all I had to do was arduino.write("commad") and to encode it to utf-8. After this command is sent, the arduino controls the servo.

This is a snippet of whats in the arduino code

```
void loop() {
  if(Serial.available() > 0){
    String command = Serial.readString();
    if(command.equals("stop")){
      myservo.write(93);
    }
```

If the serial is available, it reads the string given and if that string is equal to stop, it servo writes 93, which I've explained is the value to get it to stop. This seems very simple, but I had many complications of where this did not work.

Challenges:

- Serial communication
    - Problem: Writing to the arduino would not change the movement of the servo
    - Solution: making sure only one command is given at a time
    - Problem: pycharm could not access port4
    - Solution: because arduino IDE is already accessing port4, closing the application worked
    - Problem: everything working fine, all commands from Arduino IDE and pycharm correct, but servo still not changing
    - Solution: having to set the baud rate to be the same value in both programs.
- Object recognition
    - Problem: lots of false positives that are not cars
    - Solution: setting a counter to count if the given object is in the center, left or right frames, and when that counter reaches 15, actually send command
    - Problem: counter not resetting, and then sending many commands to arduino
    - Solution: no matter what, after sending a command, set all counters to 0
- Servo function
    - Problem: no longer can rotate motor at precise angles
    - Solution: stopping the rotation when object is in the center
    - Problem: servo not spinning 360 degrees
    - Solution: taking apart servo and making the potentiometer not affect the ac motor
- Camera:
    - Problem: program not outputting the address, and failing after everything is correct
    - Solution: making the partition scheme bigger, so all the code can be transferred
    - Problem: couldn't access camera when camera was connected to hotspot and library computer connected to school wifi
    - Solution: both have to be connected to the same wifi.

Sources:

- https://www.youtube.com/watch?v=zZGkkzMBL28 (modify servo from 180 degree to 360)
- https://apmonitor.com/pds/index.php/Main/CascadeClassifier#:~:text=The%20OpenCV%20Python%20package%20has,a%20face%20is%20not%20detected. (open cv documentation in python)
- https://github.com/souravdeyone/OpenCV-Reference/blob/master/Haarcascades/haarcascade_car.xml (car xml file for object detection)
- https://docs.arduino.cc/resources/datasheets/ABX00080-datasheet.pdf (arduino uno documentation)