Name: Jorge Gonzalez

Username: jgonz096

Class: CS179J

Date: 6/9/2017

Write-up for Automated Mouse Trap

1. Overview:
    a. This project is an automated mouse trap designed by myself using the Raspberry Pi model 2b+. This trap is built around an enclosure as the base for the trap. Attached to this enclose is Floor Sensitive Resistor (FSR), a servo motor, and a Camera. The trap works by having bait placed behind the FSR and waiting for some external force to act upon it. This then sets off the servo-motor to shut the door to the trap, trapping the creature inside. At this moment the camera takes a picture of the creature inside, then it sends the user a message via SMS and via Email. The email will contain the image that was just captured. The user can control some key functions with buttons on the side. There are 3 buttons to use. One button turns off the device, another button makes a request to a server that contains a database of food items, and the last button opens up the door on the device when it is closed. The button for requests is done over TCP connection, where the trap acts as a client and the user will input the creature he or she wants food for, then the server will return what food would be good as well if there is any in your refrigerator or not. As an added feature, the user can download an app called RaspiCAM that will he or she to view a live feed from the camera over Wifi, letting the user get a view of what is in the trap.
2. Hardware:
    a. Raspberry Pi 2b+
    b. Solderless Breadboard
    c. Raspberry Pi 7-inch Display
    d. Adafruit MCP3008 8-Channel 10-Bit ADC SPI Interface
    e. Arducam Sensor OV5647 Mini Camera video module
    f. TowerPro SG90 Micro Servo
    g. Interlink 406 square Force Sensitive Resistor (FSR)
    h. Wifi and Bluetooth Dongles (since RPi 2b+ does not have Wifi or bluetooth capabilities)
3. Software:
    a. Rapsberry Pi Debian OS
    b. Python-based code only
    c. Mobile App: RaspiCAM Remote
4. Useful Links:
    a. Learning how to control servo motor
        i. http://www.toptechboy.com/raspberry-pi/raspberry-pi-lesson-28-controlling-a-servo-on-raspberry-pi-with-python/
    b. Learning how to use MCP3008
        i. https://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/script
    c. Learning how to use the FSR
        i. https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr

        ii. https://acaird.github.io/computers/2015/01/07/raspberry-pi-fsr
    d. Learning how to send Email over SMTP while bypassing Google's security protocols
        i. https://iotbytes.wordpress.com/programmatically-send-e-mail-from-raspberry-pi-using-python-and-gmail/
            1. note: Code obtained for my project for emails were obtained from Github links in this link. They are:
                a. https://github.com/pradeesi/email-from-raspberry-pi/blob/master/settings.ini
                b. https://github.com/pradeesi/email-from-raspberry-pi/blob/master/email_handler.py
                c. https://github.com/pradeesi/email-from-raspberry-pi/blob/master/demo.py
        ii. http://naelshiab.com/tutorial-send-email-python/
    e. About RaspiCAM
        i. https://play.google.com/store/apps/details?id=com.pibits.raspberrypiremotecam
    f. Attaching Image files to emails
        i. https://stackoverflow.com/questions/13070038/attachment-image-to-send-by-mail-using-python
    g. Youtube Video Link
        i. https://youtu.be/9-b1mAIkLSo
5. Test Cases during Creation:
    a. Test Servo Motor by itself
        i. Led to discovery that Servo motor has to be constantly told to be in one position. The motor can't be set to one position and be expected to stay there without being set to that position repeatedly over and over, thus the reason why it seems to stutter.
    b. Testing Email
        i. When testing the email, I knew I had written basic SMTP code for google and made sure it was correct by visiting various links that told me so. For some reason however, Google would block me from sending them. It turned out their security was blocking me, probably due to my 2-step security verification. I downloaded some code someone wrote that bypassed this issue and that fixed the error. Link is in the above section.
    c. Testing SMS
        i. Testing SMS was done using the Twilio python libraries. For some reason it wouldn't work at first but that was because I needed an account and the tutorial I followed was somewhat outdated. Turns out the way I was importing the library was wrong. Fixing that fixed my issues there.
    d. Camera
        i. Using the camera itself wasn't so hard. The main bug I had with it was that the camera itself wouldn't turn off once it was turned on, thus hogging up resources until python would crash the program. I didn't catch this until after testing the camera several times in succession because it would usually work on one trial. The fix was to delete the camera object in the python code once it finished running.
    e. Client/Server Testing

i. The client and server tests weren't that difficult to implement. The code is basic but what didn't work was connections over different LANs. I tried connecting to my partner's device over a another network and it would not work whereas when we would connect to the same network the devices would work just fine.
ii. Note: As a result, if my code is tested for any reason by anyone other than my partner and I, DO NOT run the client and server code unless the HOST is set to '', meaning "localhost", or else you'll likely make the client hang and stop the device.

f. DC motor Test
   i. Implementing the DC motor was simple enough but finding out that I could not accurately move it was what caused me to switch to a servo motor. Attempts to time the DC motor just right failed during testing so switching was mandatory.