“On Air” - Zoom Mic Indicator for Quarantine

ECE 508 - Introduction to Python Project

Chuck Faber

Objective and Motivation

The objective of this project was to create a sign that would indicate to my husband when my microphone was on during a Zoom meeting. The motivation for this, is that since the COVID breakout, we've all been under quarantine, and for the first time for many of us, most of our work and courses are now done at home, often in shared spaces with the rest of our family.

These close quarters often lead to certain stresses, including my husband walking in during a live meeting with others and causing disruption, or vice versa. I want my husband to know when my microphone is active so he knows when not to disrupt, and vice versa so we might communicate at times that both work for us.

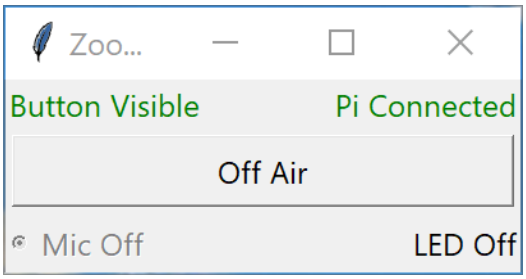
Implementation and Organization

The idea I had was to connect a sign to some LEDs which would turn on or off based on whether the mute button was active or inactive on my Zoom meeting window. I wanted to do this wirelessly with a Raspberry Pi Zero W. My initial idea was to tell directly if the microphone was muted or unmuted using the Zoom API but this proved too difficult to figure out. So I opted instead for trying to scan my screen for the Zoom mute or unmute button to tell me whether or not the mic was on.

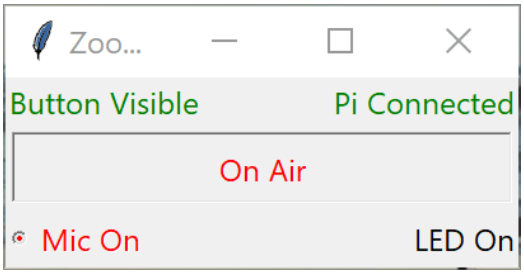
I ran into some issues with a library I was using and so decided to create a GUI that would turn the Zoom mic on or off, and then send the signal to the Raspberry Pi. I eventually decided to use a different library for the features of finding and clicking the Zoom mic button that upon retrospect could eliminate the need for a GUI at all, but for this project I created the GUI. I decided it would be good practice for the concepts taught to us in this class, and wanted to to take on the challenge of doing it.

So currently, what my code does is open a GUI that has one button and 4 status labels. One which tells the user if the mic button is visible, one which tells the user is the Raspberry Pi is connected, one which tells the user if the mic is on or off, and the last which tells the user if the LED sign is on or off.

The GUI looks as follows with the Mic Off:



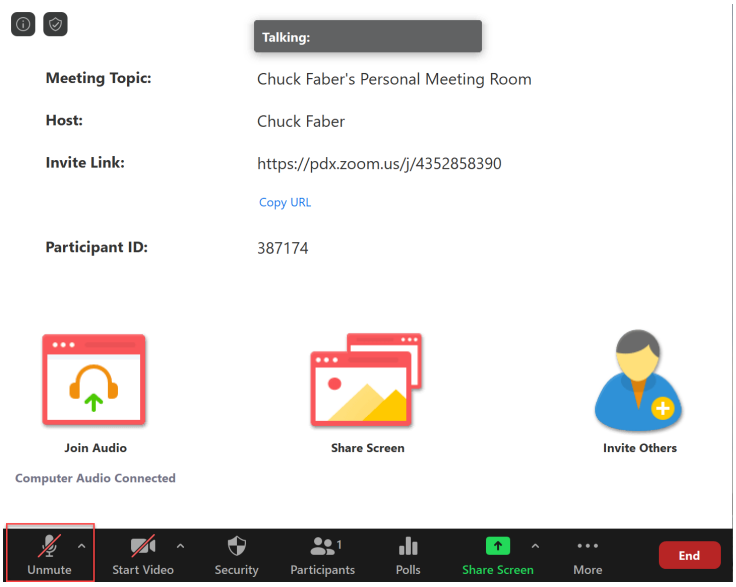
And as follows with the Mic On:



Functional Limitations

Some limitations of the script are as follows:

1. The Zoom meeting window must be open and the mic button must be visible and unblocked by other windows or objects. (see below)
2. If using dual screens, the Zoom meeting window must be on the primary screen.



Organization

My code is organized into two scripts. A client script which runs on my PC, and a server script which must run on my Raspberry Pi Zero W. The two scripts communicate with each other via a TCP socket.

Server-Side Script (server.py)

This script is the simpler of the two, so I will discuss this one first. This script first does some setup of the GPIO pins on the Raspberry Pi, setting five pins as outputs and initializing them to LOW.

The script then creates a socket, and configures it to listen for any IP address communicating on port 12345. Then it awaits a command on this socket. The possible commands it may receive are ‘on’, ‘off’, and ‘quit’. All other commands will return the ‘unknown command’ reply. The ‘on’ command will set all GPIO pins high. The ‘off’ command will set them all low. The ‘quit’ command will close this connection and continue listening again for another connection.

Client-Side Script (client.py)

This script declares a class ‘onAirGUI’. The __init__ function first initializes and sets up all of the variables we will use in this class. The first items are the filepath to the mute and unmute images that the script will search for on the screen. These will have to be changed accordingly for anyone else running this script.

Next are the HOST and PORT definitions. The PORT can remain unchanged, but the HOST IP must be changed to the IP address of the Raspberry Pi on the network.

The state variables are all for changing the GUI display or particular labels.

The script then creates a tkinter object, and initializes the string variables that display on the GUI itself. Then all of the tkinter widgets are initialized using the grid organization method. There are four labels, and one button, and one indicator (for the mic).

A queue is initialized for communicating commands over the socket, (because this communication must take place in a separate thread), and the socket itself is created.

Two threads are initialized, one for communicating over the socket, and one for periodically checking whether the Zoom mic button is visible.

Then a continuously running updating method is called to keep the GUI updated with state changes to the state variables. A protocol is also defined for executing the ‘close’ method if the GUI window is closed to properly join all threads, and close the socket.

The methods in the onAirGUI class are as follows:

- close(self) - method which is called when GUI is closed so that all threads join properly and the socket is closed.
- checkZoom(self) - Method which continuously run in a separate thread to check if the Zoom mic button is visible on the screen to update the GUI.
- socketListen(self) - Method which communicates via the socket in a separate thread when the queue has a command waiting. It sends the command and listens for a reply.
- updateLabels(self) - Method which alters the GUI configurations and widgets to accurately display the current state of the script and sign, changing the text and colors as needed.
- buttonPress(self) - The button callback when it is activated on the GUI. It will change certain state variables, try to find and click the mute button on Zoom, and place commands in the queue to be sent over the socket to the Raspberry Pi to turn on the LEDs.

Running Code

Dependencies

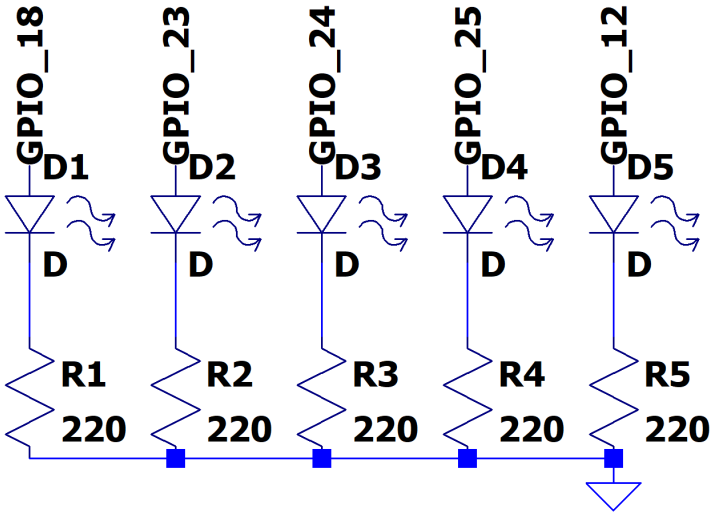
The following may need to be installed through pip.

- Python 3
- tkinter
- pyautogui
- openocd (for the confidence parameters in pyautogui.locateOnScreen)
    - Note: not imported into script but needs to be installed.
- time
- threading
- socket
- queue

How to Run

Hardware Setup

1. Create the following simple LED circuit below and tie the appropriate GPIO pins on the Raspberry Pi to the appropriate LEDs.



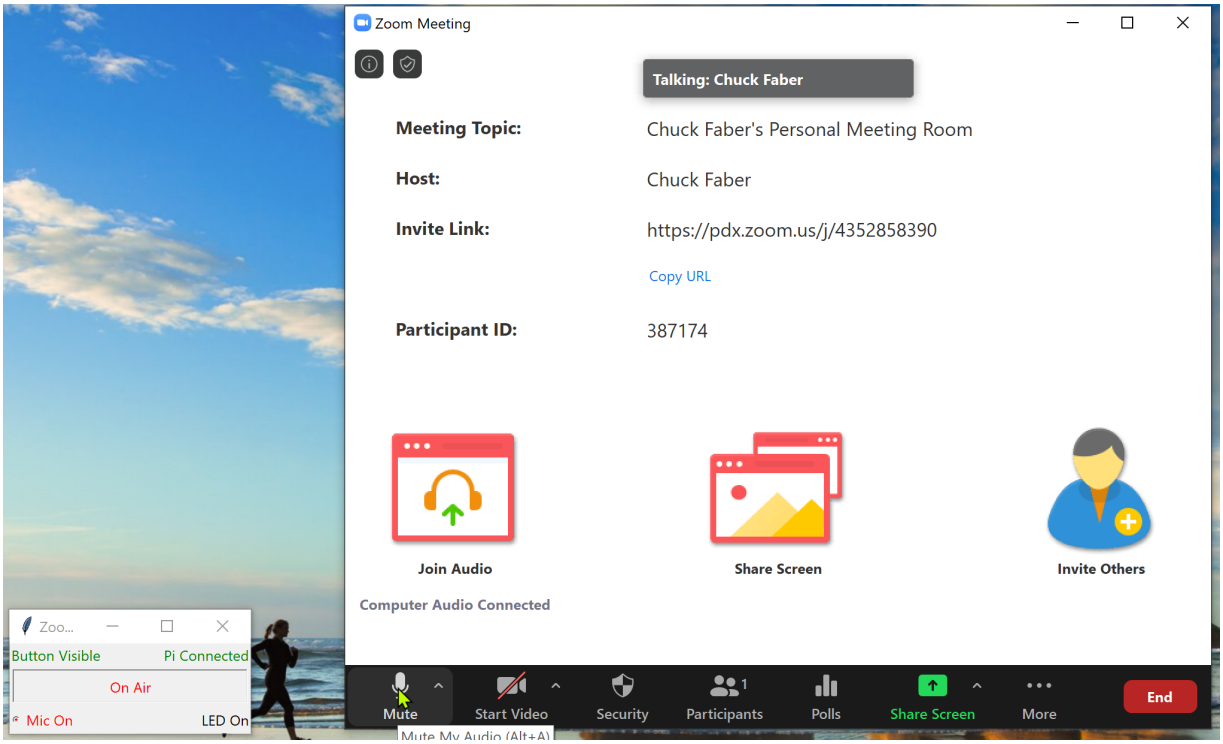
2. Make sure Raspberry Pi is set up, powered, and connected to WiFi. Setting it up for ssh communication is useful also.
3. Get the Raspberry Pi IP address using `ifconfig`.

Software Setup

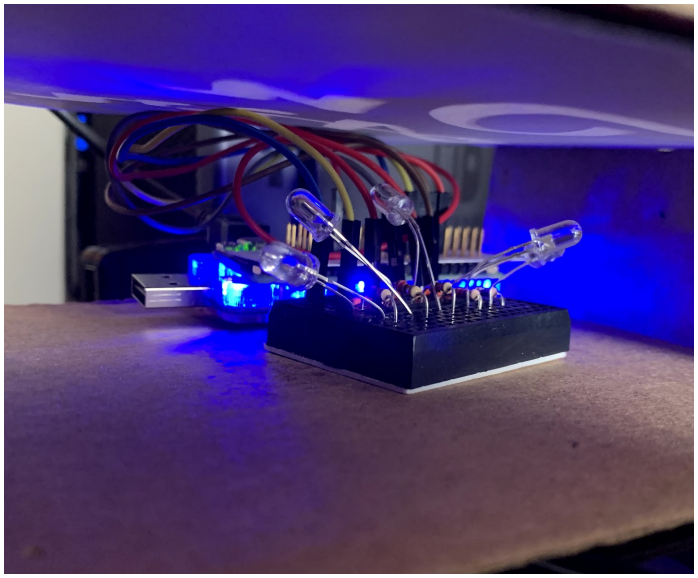
1. In the client.py script, change the file path to the mute and unmute images as needed.
2. In the client.py script, change the host IP for the Raspberry Pi to what is appropriate for the Raspberry Pi on your network.
3. Access your Raspberry Pi and run the server.py script with `python server.py`.
4. On your PC run the client.py script with `python client.py`. The GUI should open up.
5. Open a Zoom meeting, and connect with audio. The GUI and the Zoom meeting should both be visible with no objects obstructing the Zoom mic button.
6. On the GUI click the “Off Air” button to go on air, and turn on the sign. To go off air, click the “On Air” button to toggle the mic off.

Results

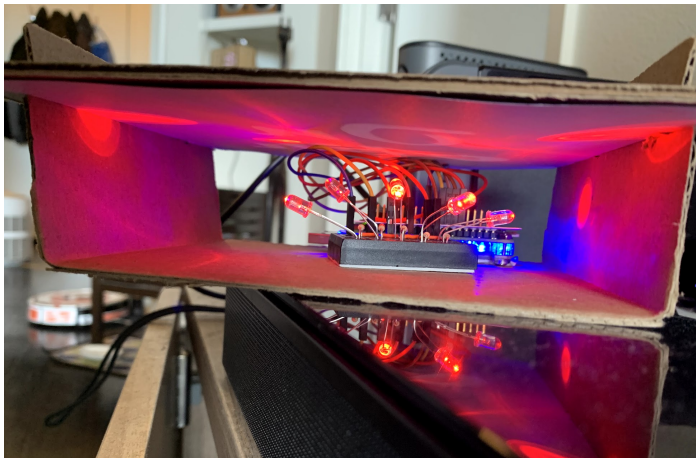
The final product is functional, if not attractive. Pressing the “Off Air” button on the GUI finds the Zoom Mic button, and clicks it (if it is visible) and sends a command over the socket to turn on the LEDs wirelessly. The button toggles to red “On Air” text, and the label text and indicators change on the GUI. Clicking the “On Air” button turns the LEDs back off, and re-mutes the Zoom mic.



GUI with Zoom mic button active (after click the 'Off Air' button to toggle to 'On Air')



Sign while 'Off Air' (ignore the blue LEDs which are just part of my RPi case)



Sign while ‘On Air’ (note: sign is not as bright as I would like)

Some screenshots of the terminal when button is clicked (not super exciting):

```
(base) C:\Users\Chuck\Insync\kpfaber@gmail.com\Google Drive\Portland State University\Spring 2020\ECE 508 - Python Workshop\ECE508-Project>C:/ProgramData/Anaconda3/python.exe "c:/Users/Chuck/Insync/kpfaber@gmail.com/Google Drive/Portland State University/Spring 2020/ECE 508 - Python Workshop/ECE508-Project/client/client.py"
led on
Failed to find button.
led off
```

Client Side Terminal

```
pi@raspberrypi:~/ECE508-Project/server $ python server.py
Socket created
Socket awaiting messages
Connected
I sent a message back in response to: on
I sent a message back in response to: off
```

Server Side Terminal

Video of Script Functioning with LEDs (please watch): https://photos.app.goo.gl/e8khEEnTAMUZQLhi9

Discussion

The effort required for this project was higher than I anticipated it would be but the challenges were not insurmountable. I feel I have put in at least 3-4 hours a week for the last 4 weeks or so, so perhaps 9-16 hours in total on this script.

Challenges

One of the challenges I encountered was having to figure out how to determine if the Zoom mic was active or not. Attempting to parse and use the Zoom API was a dead end, but after speaking to some people on a Python

forum, they floated the idea of using an automation library like AutoPy or PyAutoGUI. I first tried using AutoPy, because I found some scripts out there where people used this to take a screenshot of dual screen monitors and thought I could use this to make this script work with my two screen setup. However this was more difficult than I anticipated, and the modules they had available to search for an image within a screenshot ended up being incredibly slow and would take maybe 5 minutes to determine if the image was in a screenshot or not, and was frequently unreliable in doing so.

So I opted for the simpler PyAutoGUI which was much faster in determining the presence of a reference image on a screen, even though it only worked with a single screen.

I had to figure out how to communicate and send commands to the Raspberry Pi wirelessly. I had used a sockets library once upon a time in C, but I barely understood it even then. I figured out how to use it after much research and looking at example scripts.

I also had to figure out how to create a GUI on tkinter. The slides we went over in class were helpful for much of it, though some of it I had to figure out with further research and trial and error.

In order for my script to work, I need the GUI to run continuously while also sending commands and listening for replies on the socket, so I also had to figure out how to implement multithreading in this script project. I had some experience with threading from a prior class but it was in C, so this was new, and actually much simpler, but I didn't have to worry about using locks or anything either because each thread only changed one variable which wasn't changed by anything else.

My hardest challenge, interestingly, was trying to get everything organized into a class object. That resulted in a lot of headaches, but also lessons. Also I learned that when you describe a callback function in tkinter with parenthesis (ie. function()) it will actually run the function even before the button is pressed.

Future Work

As mentioned, I've since realized that I can probably eliminate the GUI altogether and just go with a script that uses PyAutoGUI to periodically search for the presence of the mute button on a Zoom meeting and lights up the LEDs when it finds it. This would be simpler, and probably more user friendly.

Though it would still have the same problem that the Zoom meeting window couldn't be on a secondary screen, minimized, or behind another window. I don't know how I would get around that without more research. I'd possibly have to write my own functions to find the button over multiple screens or find some way to work directly with the Zoom API.

In the future I will probably make it so that the server.py script automatically runs on the Raspberry Pi on boot. I would also like to make a more attractive sign and maybe use an LED strip for more attractive and diffuse illumination.

My GitHub repo for this project includes all of the test scripts I created or found leading up to final scripts in the 'development' folder. Feel free to peruse them if you wish: https://github.com/faberc/ECE508-Project