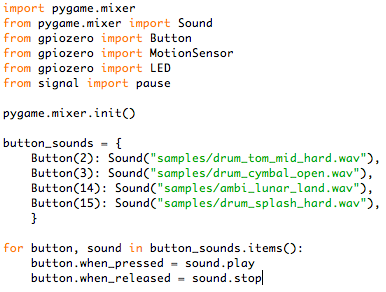
Individual Final Project Write-Up

The name of the project is *GPIO Music Box*. The group members in this project were Julio Lama and myself. Essentially what we had made was a music box with a Raspberry Pi and a breadboard using the GPIO pins on the Pi. We utilized Python, a the PyGame module, and GPIO Zero to create a mapping between each button and a different sound to make a musical box. To push this project even further, we added four buttons that each play their own unique sound. Also, we utilized an RPI motion sensor as well that played a sound whenever motion was detected. The code for the buttons and the sensor were stored in separate Python files.

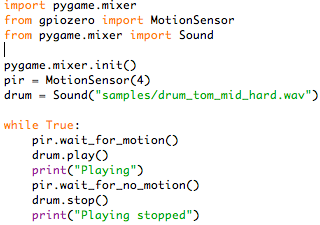
The list of equipment that we used was:

* A Raspberry Pi with an SD card
* A full-sized breadboard
* 3 male to male jumper leads
* 10 female to male jumper leads
* 4 tactile buttons
* A RPI motion sensor
* A speaker connected to the Pi via an auxiliary cable

Before even getting into the Python script we utilized for this project, one thing we needed to install on the Pi was a software called *avconv.* Now called *libav,* this is software that produces libraries and programs for multimedia data. Once that was installed on Pi (which took a while, it was a rather big library), we had to import sound samples from *Sonic Pi*. *Sonic Pi* is an open-source programming language that was developed for creating sounds. Once we had the library of sound files we copied them all over to a directory on the Pi. The sounds from the *Sonic Pi* library were in a .flac format, which works great for *Sonic Pi*, but in Python that wouldn’t work out to well. So, using *avconv,* we converted all the files to a .wav format which was usable in Python. Then it was time to create the Python script. For the tactile buttons, here’s the Python script:

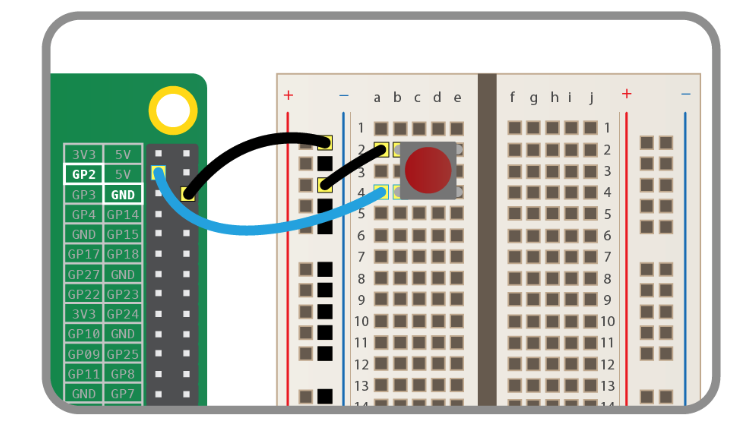


and here’s the Python script from the RPI motion sensor:

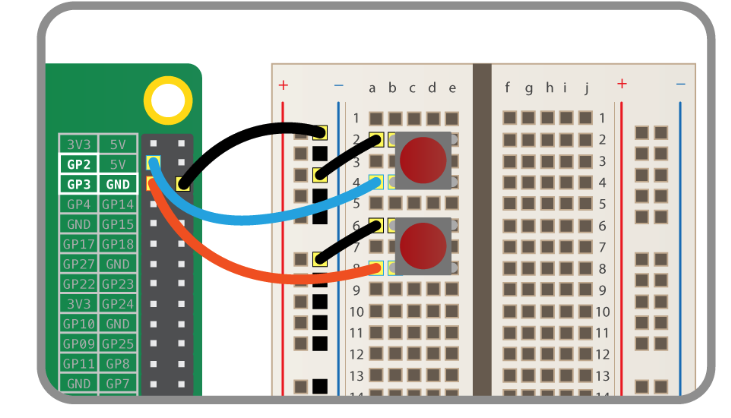


Then it was on to creating the first button. Here’s a breakdown of how we added one button:

1. Attaching a male to female cable from the GND pin of the Pi to the ground rail of the breadboard
2. Attaching a male to female cable from the GPIO pin 2 fo the Pi to the breadboard. Basically, here’s how it looked visually:

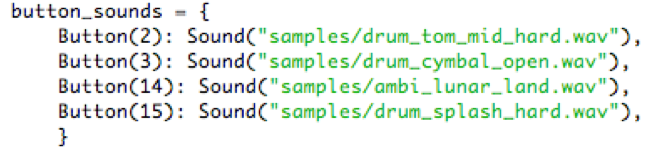


1. To add another button, here’s how it looked like:

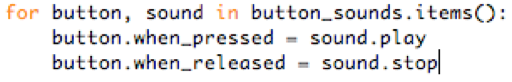


1. This setup the same when adding the third and fourth buttons.

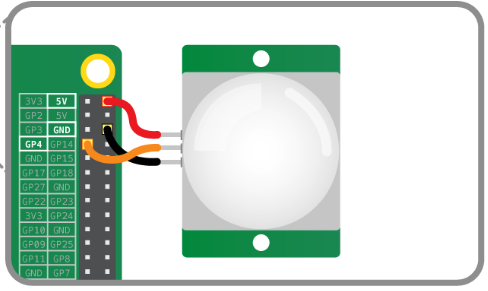
As you can see from our Python code, to store all these button sounds we had created a dictionary to map them, making the code a bit cleaner.



What this did was create instance variables for each button. Mapping each button to the proper GPIO pin on the Pi and each then played a unique sound. Then to tell each button to play a sound, we inserted this for loop into the Python script:



The *when\_pressed* line tells the button to run the sound’s *play* function when pressed. The setup for the RPI motion sensor was like so:



Simple, nothing too complex with this one. To get this to work, we had just taken the Python script from the GPIO music box and just tailored it to work for the sensor.

Moving on over to the difficulties faced, there were really none from the hardware side or group interactions. Really the one challenge we faced was from the software side. The problem with the button script that we had was once the button was pressed, the sound kept on playing and playing. It wouldn’t stop. And each time we pressed the button again, the same sound would play again over the previous sound. Then we had two sounds playing at once. So, after doing some light research on the button class and pir class in Python, we finally inserted this code into our Python script for both the button script:



and this code for the sensor script:



What this basically did is stop the sound from playing.

Overall, it was a very fun and interactive project and class. I definitely learned a lot from this lab class and am happy that I got to practice my Python skills a bit more.