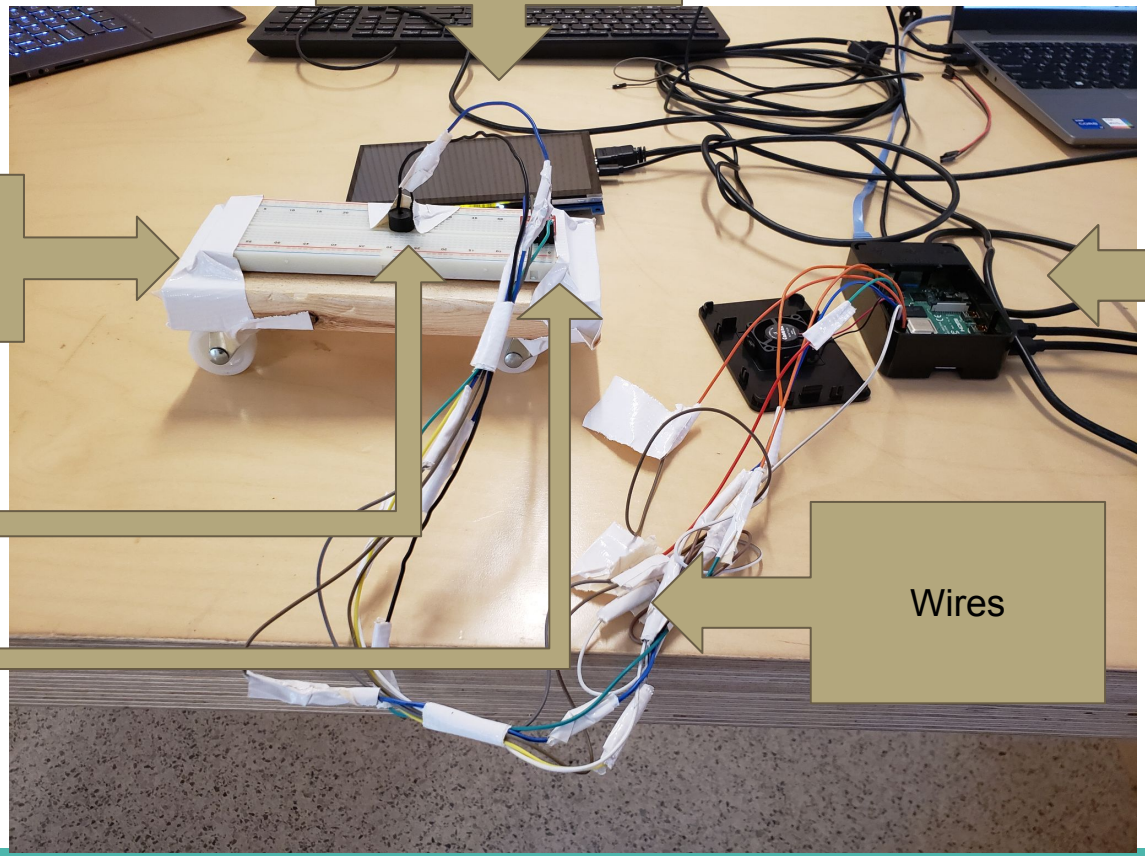
Music and Acceleration

MakeUofT Hackathon Presentation
Feb. 19, 2023

Steven Li, Kalvin Cui, Andre Rodrigues

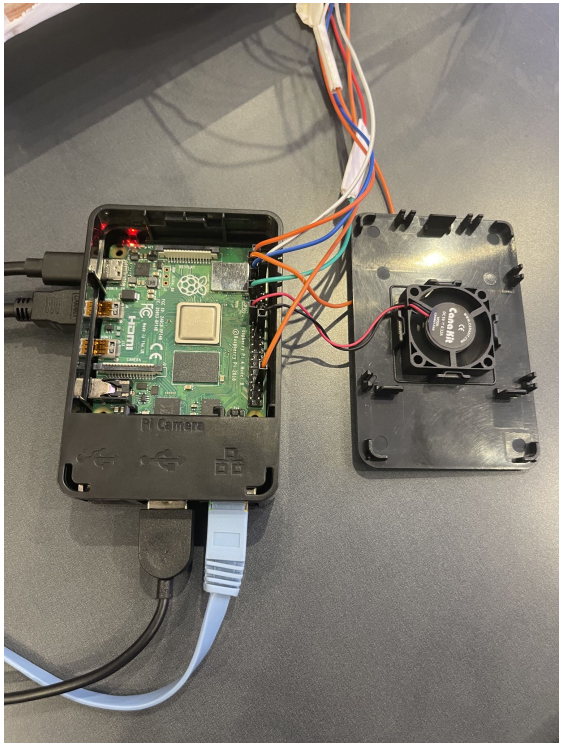
Our Product


Monitor
Moving board
Raspberry Pi 4
Buzzer
Accelerometer
Wires

1. Setting up the Raspberry Pi

1.1 Flashing Raspberry Pi

1.2 Bypassing Internet Connection using Ethernet



1.1 Flashing a Raspberry Pi

We flashed a raspberry pi for the first time.

It took much longer than expected. Our first Pi has a random system installed on it, since it was our first interaction with this machinery, a lot of time has been used to figuring out how to navigate a random system.

We eventually figured out that the Pi is not running on the correct system after chatting with a mentor.

1.2 Bypassing Internet Connection using Ethernet

Since UofT blocks device sharing on their internet, we have to come up with our own method of connecting our devices to the Pi.

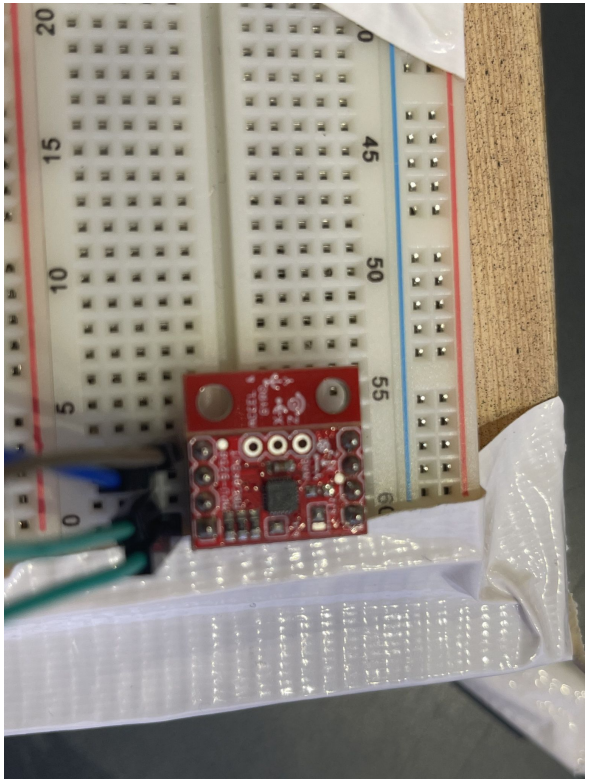
Therefore, we found an ethernet cable in one of our design teams, and used it to connect our Pi to the monitor using remote SSH via VNC Viewer.

We were able to code the Pi henceforth.

2. Testing the Accelerometer/Gyroscope

2.1 Accelerometer/Gyroscope

2.2 Code



2.1 Accelerometer/Gyroscope (with code)

Model number: MPU9250

Challenge: Lack of code, general knowledge and pinout diagram

There does not exist a pinout chart for this particular model anywhere on the internet. An arduino was used instead of a Raspberry Pi. However, after chatting with a mentor, we were able to figure out and learn different functions of pins on a Pi.

We also learned how to install and work with different libraries on a Raspberry Pi.

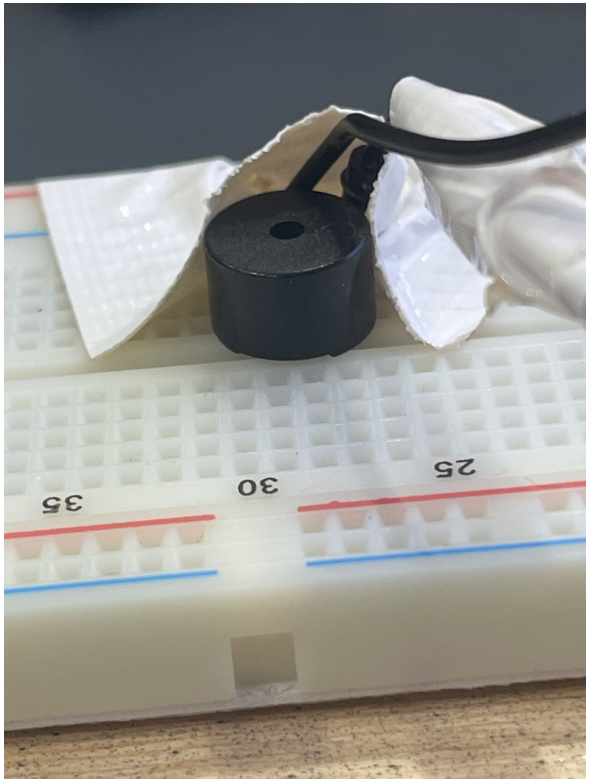
In addition, we learned the basics of using an IMU

```
import os
import sys
import time
import smbus

from imusensor.MPU9250 import MPU9250

address = 0x68
bus = smbus.SMBus(1)
imu = MPU9250.MPU9250(bus, address)
imu.begin()

while True:
      imu.readSensor()
      imu.computeOrientation()

      print ("Accel x: {0} ; Accel y : {1} ; Accel z :
{2}".format(imu.AccelVals[0], imu.AccelVals[1],
imu.AccelVals[2]))
      #print ("Gyro x: {0} ; Gyro y : {1} ; Gyro z :
{2}".format(imu.GyroVals[0], imu.GyroVals[1],
imu.GyroVals[2]))
      #print ("Mag x: {0} ; Mag y : {1} ; Mag z :
{2}".format(imu.MagVals[0], imu.MagVals[1], imu.MagVals[2]))
      #print ("roll: {0} ; pitch : {1} ; yaw :
{2}".format(imu.roll, imu.pitch, imu.yaw))
      time.sleep(1)
```

3. Testing the Buzzer

3.1 Buzzer Implementation

3.2 Demonstration



3.1 Buzzer Methods and Interpretation

- We were able to use and test out several versions of Piezo buzzers
- After obtaining a functional setup, we experimented with different buzzer frequencies and start buzzer parameters
- We experienced some issues with inconsistent pitch when varying the buzzer frequency
- We attempted to tune the buzzer, however, it became evident that iPhones could not pick up mechanical noise.

```
#Libraries
import RPi.GPIO as GPIO
import time
from time import sleep
#Disable warnings (optional)
GPIO.setwarnings(False)
#Select GPIO mode
GPIO.setmode(GPIO.BCM)
#Set buzzer - pin 23 as output
triggerpin = 17
GPIO.setup(triggerpin,GPIO.OUT)
#Run forever loop
#buzzer.changeFrequency(1000)
buzzer = GPIO.PWM(triggerpin, 784)
duration = 0.5
freqCycle = 1

c = 2093
d = 2300.32
e = 2537.02
f = 2793.83
g = 3135.96
a = 3520.7
b = 3900.07
c8 = 4186.01


def playNote(freq):
    buzzer.ChangeFrequency(freq)
    t_end = time.time() + duration

    while time.time() < t_end:
        buzzer.start(freqCycle)
    buzzer.stop()
```

3.2 A Demonstration

https://youtube.com/shorts/Zp2Mf-PjUoc?feature=share

4. Building the Physical Model

4.1 Soldering

4.2 Drilling and Taping

5. Final Product

- The final product was able to produce tones of varying pitch according to the accelerometer reading
- However, there are still some gaps in functionality, namely due to the change from positive to negative acceleration values, which result in lower pitches even when the cart is moving fast

Try Rolling the