Smart Tuning Assistant

Problems

01

Humidity causes top of the instruments to expand affecting sound. The bridge and strings can push upwards creating tension in the neck of the guitar, destroying the finish. Strings exposed to high levels of moisture may absorb the water, which will deteriorate the strings quicker.

02

High temperatures cause strings to loosen, lowering pitch of the instrument and not all of the strings are affected the same. Heat can also fade the finish of your instrument and warp the body. Cold weather causes sharper tunes.

03

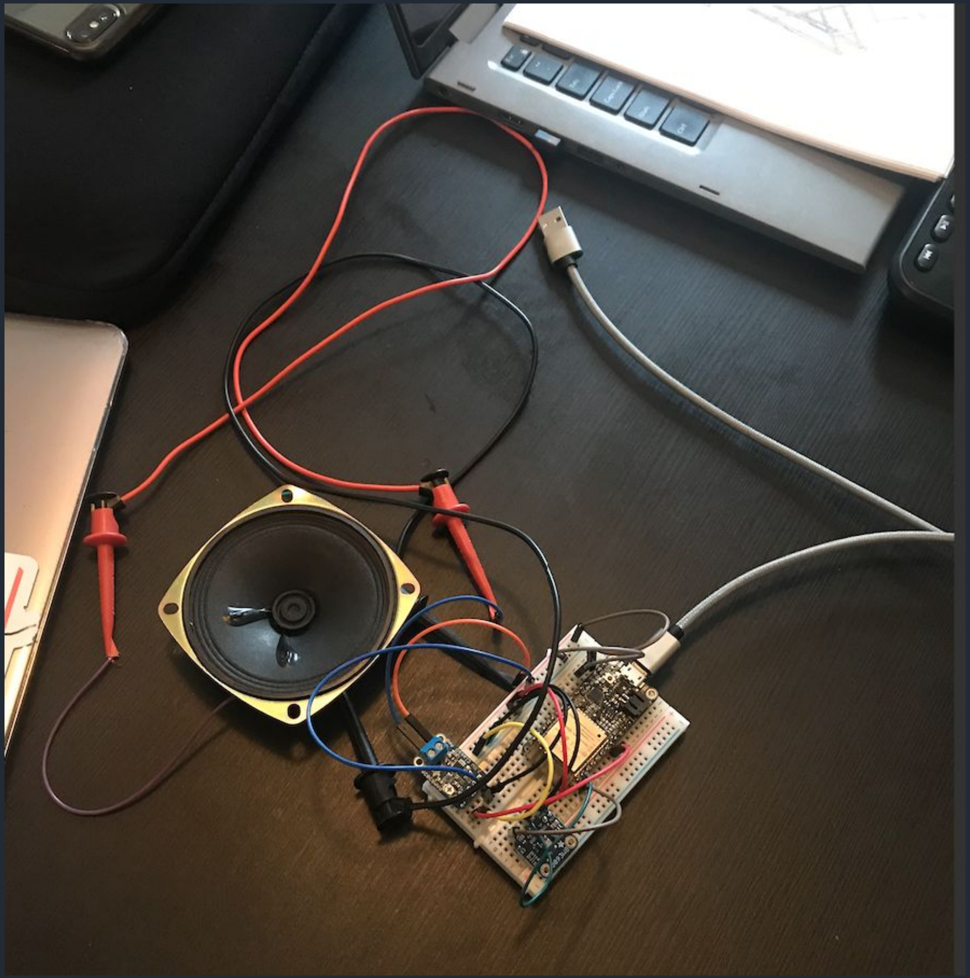
New instrument users may not know how to tune their guitar or what notes they should achieve.

Create a device that:

01 Has temperature and humidity sensing capabilities

02 Allows users to check frequencies of different notes and compare when tuning

03 Sensor display changes colors (from green to red) if ideal conditions are not met for tuning and storing

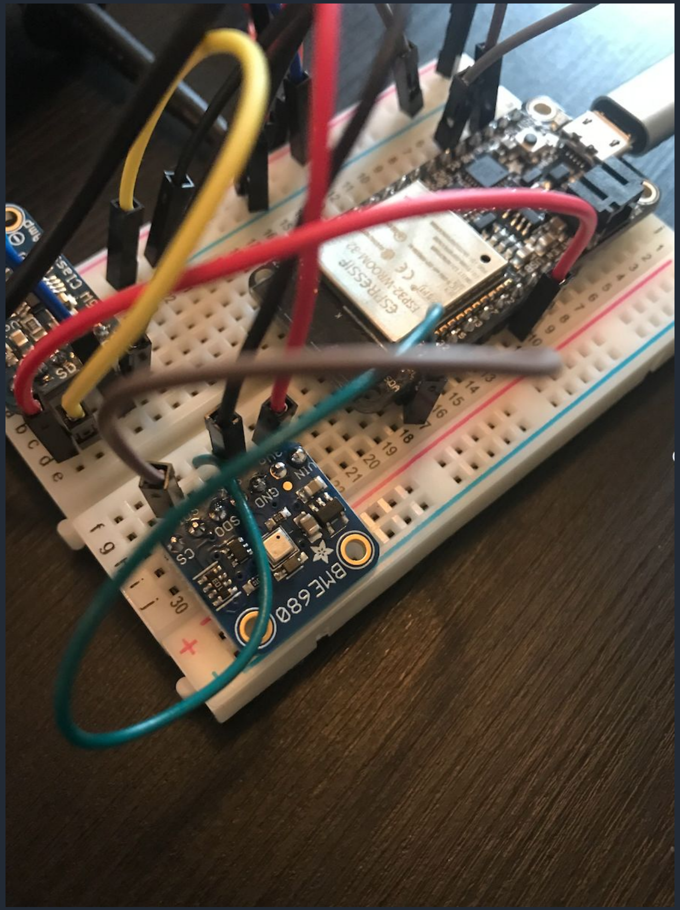


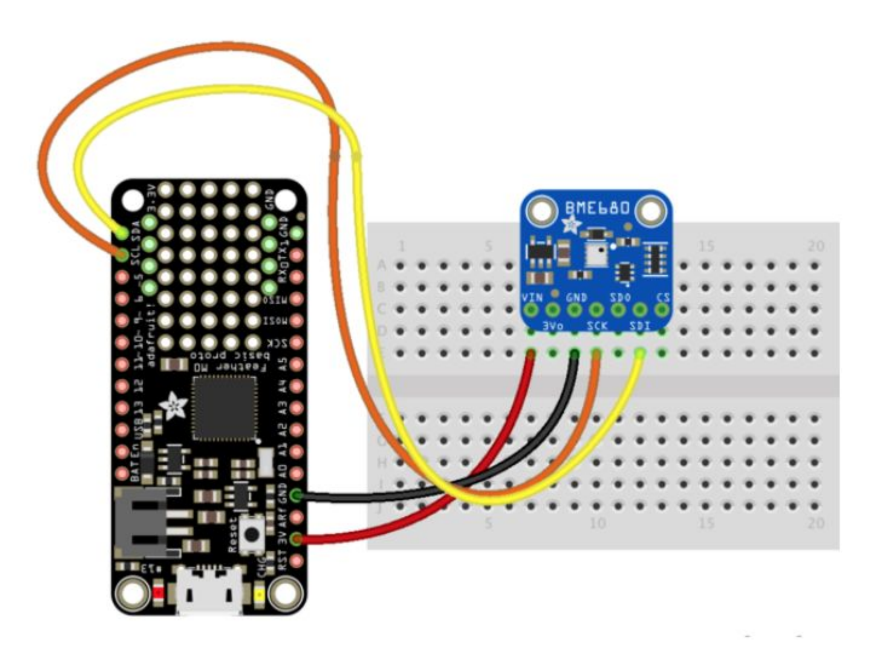
Hardware

Set Up (sensor to board):

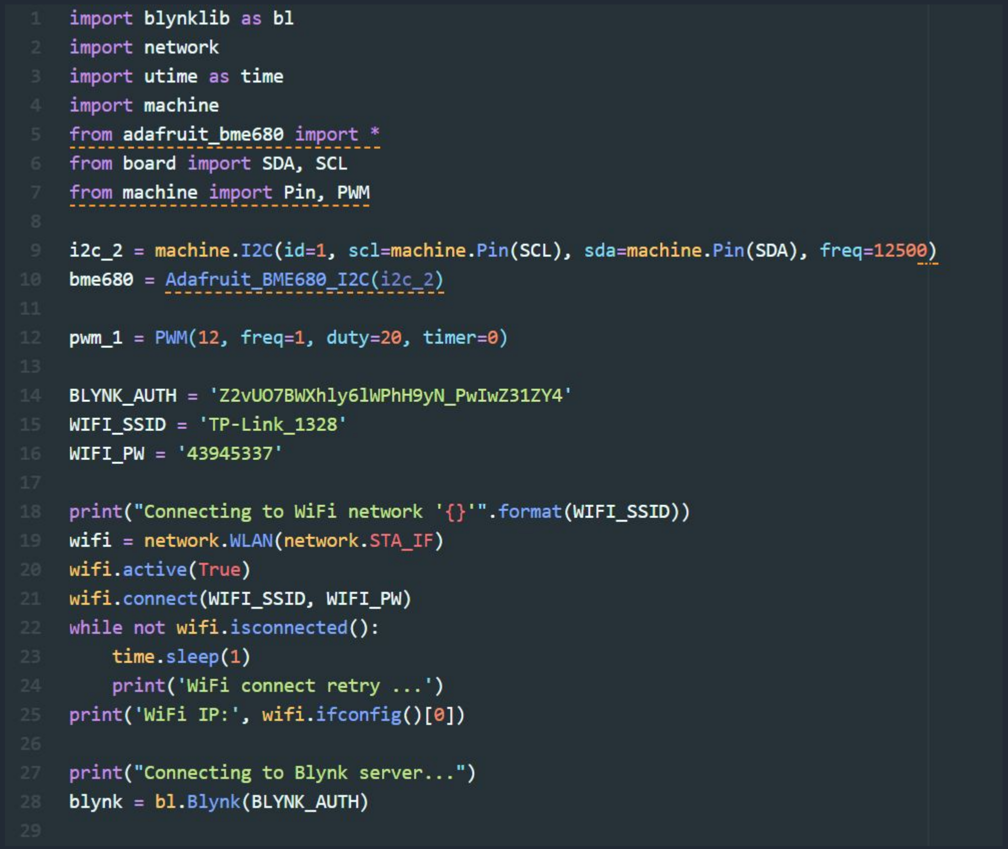
- Board 3V to BME680 VIN
- Board GND to BME680 GND
- Board SCL to BME680 SCK
- Board SDA to BME680 SDI

Sensing



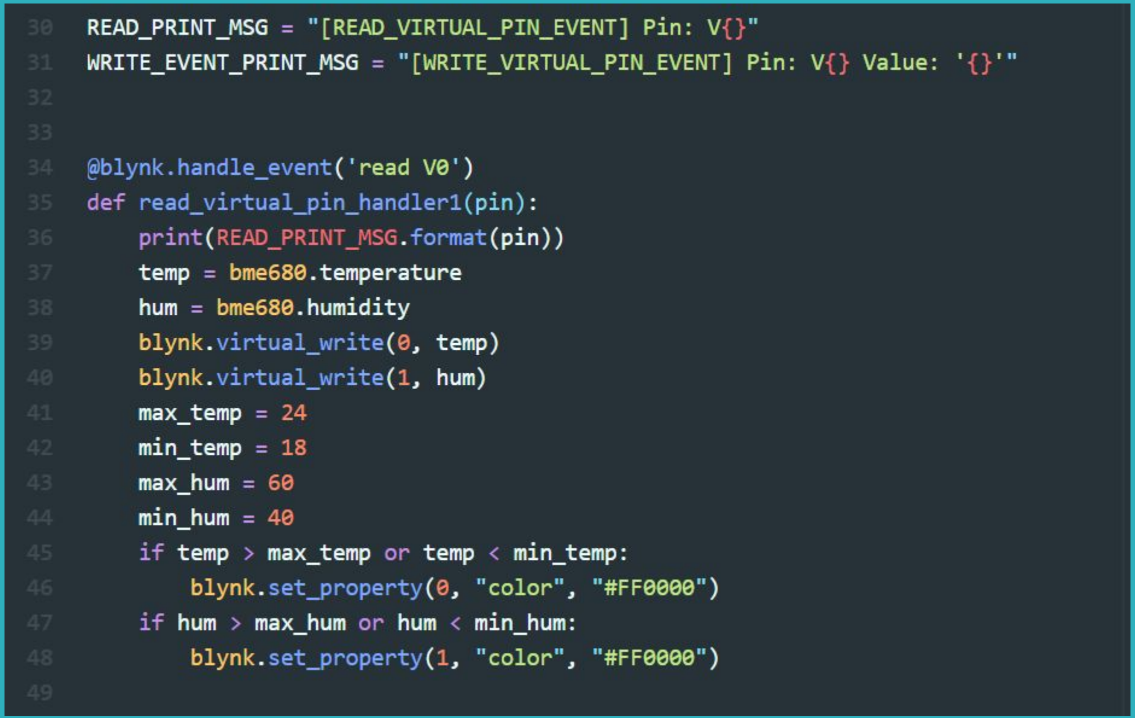

BME680
VIN
GND
SDO
CS
3Vo
SCK
SDI
SCL
SDA
adafruit! Feather M0 basic proto
Reset

Sensing



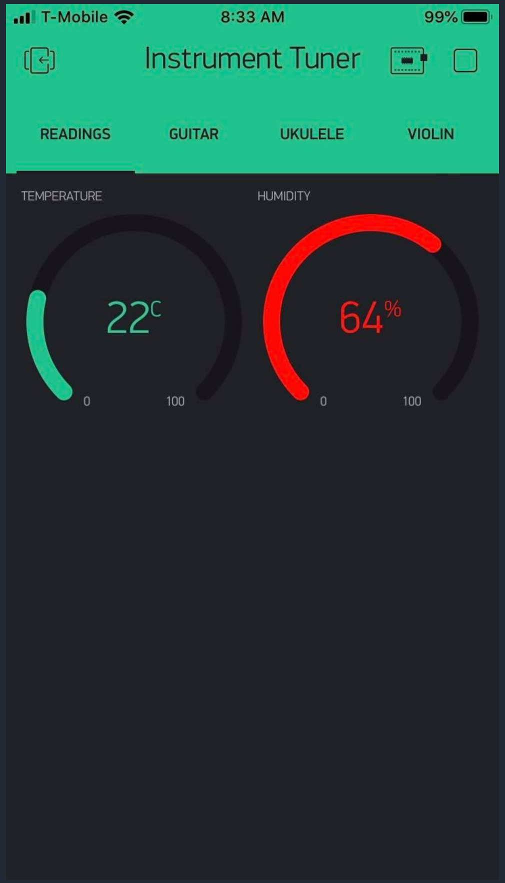
```python
import blynklib as bl
import network
import utime as time
import machine
from adafruit_bme680 import *
from board import SDA, SCL
from machine import Pin, PWM

i2c_2 = machine.I2C(id=1, scl=machine.Pin(SCL), sda=machine.Pin(SDA), freq=12500)
bme680 = Adafruit_BME680_I2C(i2c_2)

pwm_1 = PWM(12, freq=1, duty=20, timer=0)

BLYNK_AUTH = 'Z2vUO7BWXhly6lWPhH9yN_PwIwZ31ZY4'
WIFI_SSID = 'TP-Link_1328'
WIFI_PW = '43945337'

print("Connecting to WiFi network '{}'".format(WIFI_SSID))
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PW)
while not wifi.isconnected():
    time.sleep(1)
    print('WiFi connect retry ...')
print('WiFi IP:', wifi.ifconfig()[0])

print("Connecting to Blynk server...")
blynk = bl.Blynk(BLYNK_AUTH)

```

Code



```python
READ_PRINT_MSG = "[READ_VIRTUAL_PIN_EVENT] Pin: V{}"
WRITE_EVENT_PRINT_MSG = "[WRITE_VIRTUAL_PIN_EVENT] Pin: V{} Value: '{}'"


@blynk.handle_event('read V0')
def read_virtual_pin_handler1(pin):
    print(READ_PRINT_MSG.format(pin))
    temp = bme680.temperature
    hum = bme680.humidity
    blynk.virtual_write(0, temp)
    blynk.virtual_write(1, hum)
    max_temp = 24
    min_temp = 18
    max_hum = 60
    min_hum = 40
    if temp > max_temp or temp < min_temp:
        blynk.set_property(0, "color", "#FF0000")
    if hum > max_hum or hum < min_hum:
        blynk.set_property(1, "color", "#FF0000")

```

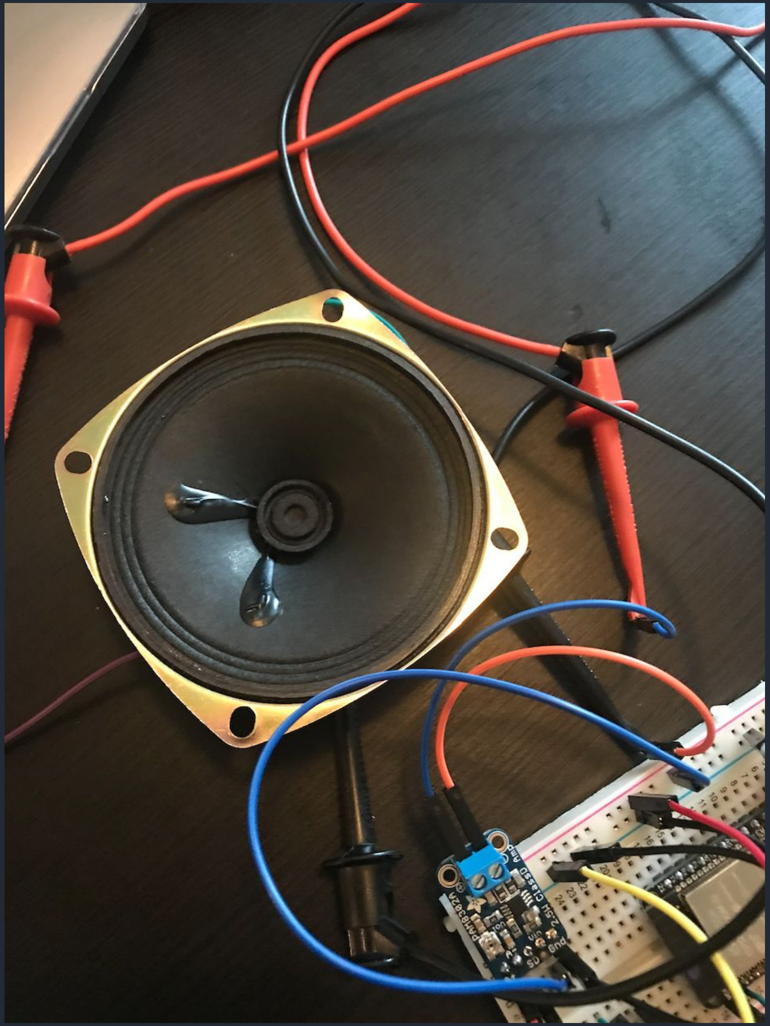
Sensing



Display

- Real Time Display of Sensor data
- Temperature on a scale of 0 to 100 C
- Humidity from 0-100%
- Display turns red when ideal conditions are not met for tuning and storing.

- IDEAL HUMIDITY: 40-60%
- IDEAL TEMPERATURE: 18-24 deg C



Sound

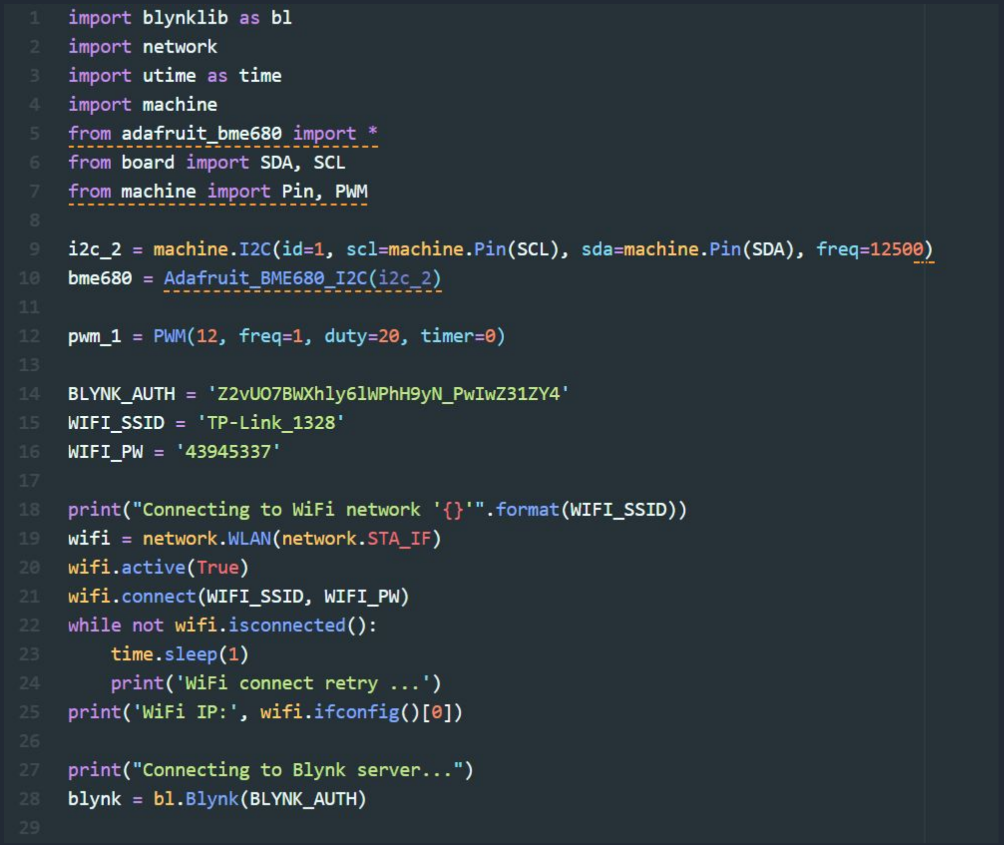
Set Up (amplifier to board):

- Amplifier Vin to Board 3V
- Amplifier GND to Board GND

Set Up (speaker to amplifier):

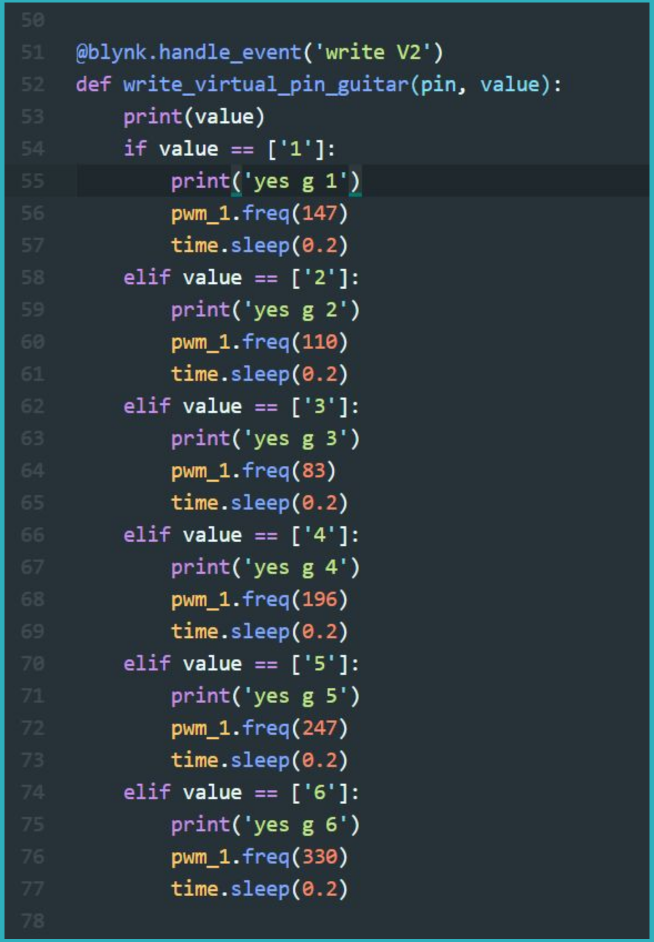
- Speaker (+) to Amplifier (+)
- Speaker (-) to Amplifier (-)
- Board SCL to BME680 SCK
- Board SDA to BME680 SDI

Sound



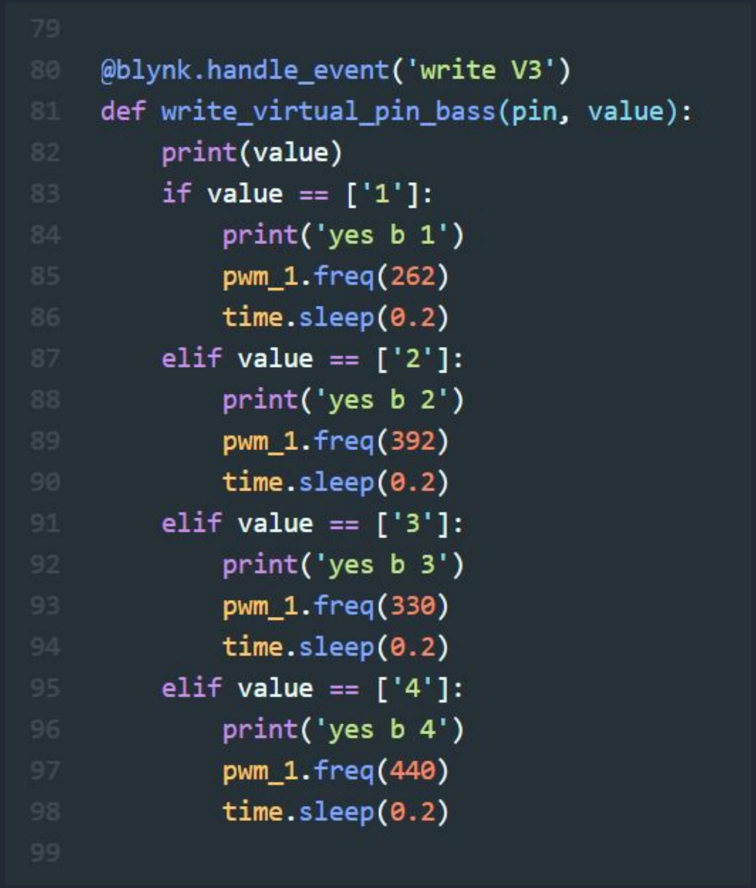
```
import blynklib as bl
import network
import utime as time
import machine
from adafruit_bme680 import *
from board import SDA, SCL
from machine import Pin, PWM

i2c_2 = machine.I2C(id=1, scl=machine.Pin(SCL), sda=machine.Pin(SDA), freq=12500)
bme680 = Adafruit_BME680_I2C(i2c_2)

pwm_1 = PWM(12, freq=1, duty=20, timer=0)

BLYNK_AUTH = 'Z2vUO7BWXhly6lWPhH9yN_PwIwZ31ZY4'
WIFI_SSID = 'TP-Link_1328'
WIFI_PW = '43945337'

print("Connecting to WiFi network '{}'".format(WIFI_SSID))
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PW)
while not wifi.isconnected():
    time.sleep(1)
    print('WiFi connect retry ...')
print('WiFi IP:', wifi.ifconfig()[0])

print("Connecting to Blynk server...")
blynk = bl.Blynk(BLYNK_AUTH)

```

Code

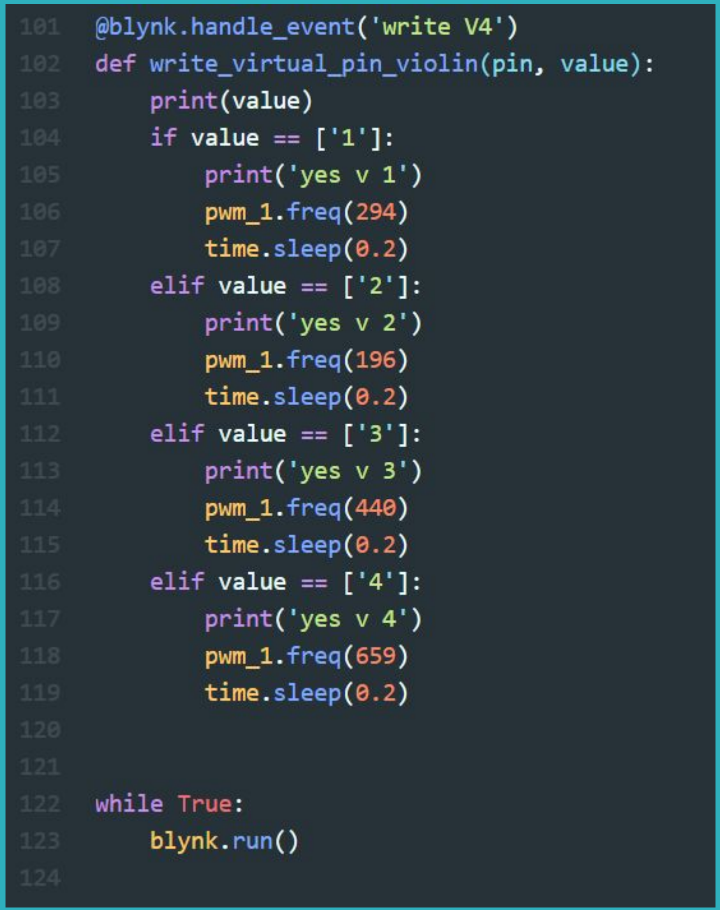


```

@blynk.handle_event('write V2')
def write_virtual_pin_guitar(pin, value):
    print(value)
    if value == ['1']:
        print('yes g 1')
        pwm_1.freq(147)
        time.sleep(0.2)
    elif value == ['2']:
        print('yes g 2')
        pwm_1.freq(110)
        time.sleep(0.2)
    elif value == ['3']:
        print('yes g 3')
        pwm_1.freq(83)
        time.sleep(0.2)
    elif value == ['4']:
        print('yes g 4')
        pwm_1.freq(196)
        time.sleep(0.2)
    elif value == ['5']:
        print('yes g 5')
        pwm_1.freq(247)
        time.sleep(0.2)
    elif value == ['6']:
        print('yes g 6')
        pwm_1.freq(330)
        time.sleep(0.2)

```

Sound Code

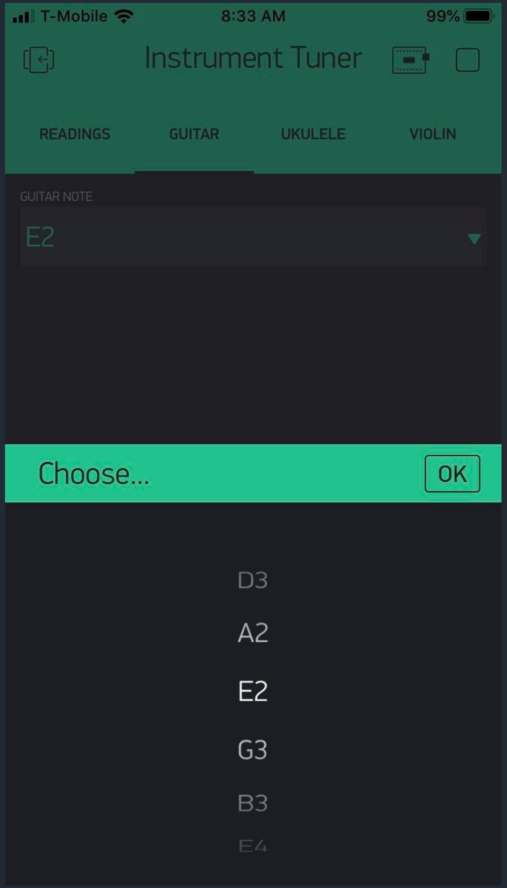


```python
@blynk.handle_event('write V3')
def write_virtual_pin_bass(pin, value):
    print(value)
    if value == ['1']:
        print('yes b 1')
        pwm_1.freq(262)
        time.sleep(0.2)
    elif value == ['2']:
        print('yes b 2')
        pwm_1.freq(392)
        time.sleep(0.2)
    elif value == ['3']:
        print('yes b 3')
        pwm_1.freq(330)
        time.sleep(0.2)
    elif value == ['4']:
        print('yes b 4')
        pwm_1.freq(440)
        time.sleep(0.2)
```



```python
@blynk.handle_event('write V4')
def write_virtual_pin_violin(pin, value):
    print(value)
    if value == ['1']:
        print('yes v 1')
        pwm_1.freq(294)
        time.sleep(0.2)
    elif value == ['2']:
        print('yes v 2')
        pwm_1.freq(196)
        time.sleep(0.2)
    elif value == ['3']:
        print('yes v 3')
        pwm_1.freq(440)
        time.sleep(0.2)
    elif value == ['4']:
        print('yes v 4')
        pwm_1.freq(659)
        time.sleep(0.2)


while True:
    blynk.run()
```

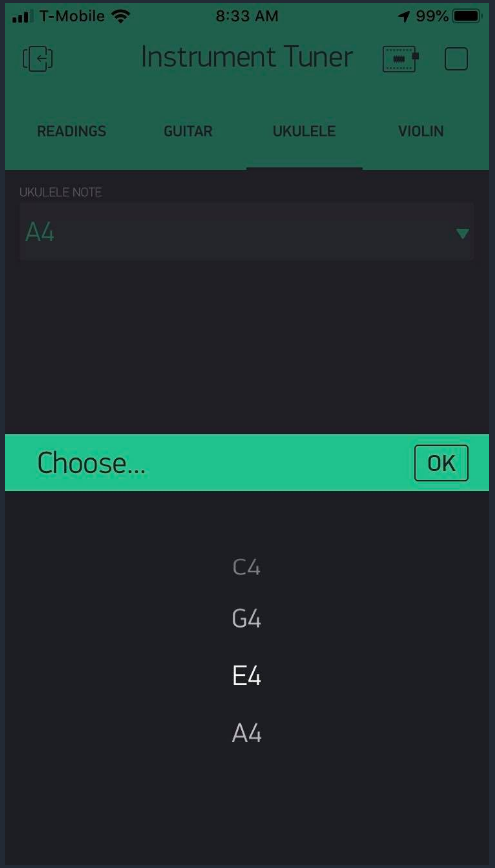
Sound Display



GUITAR NOTE
E2

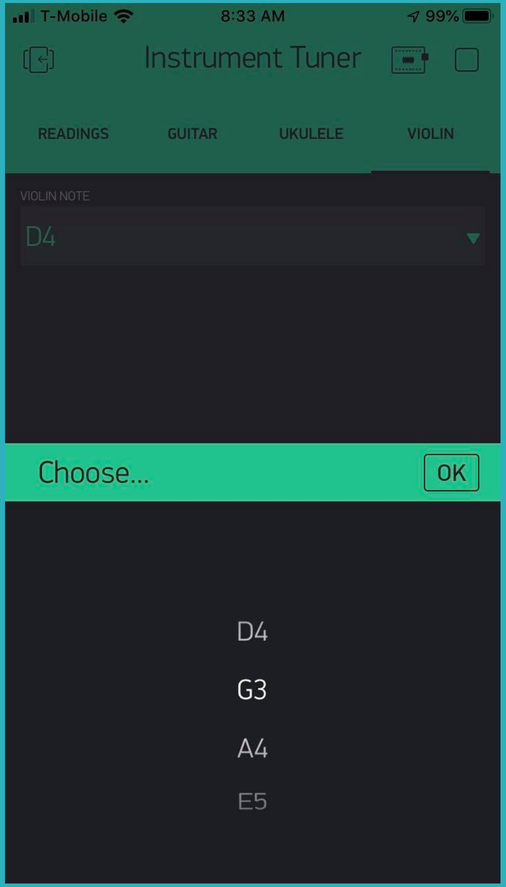
- Different Tabs for different instruments
  - Guitar
  - Ukulele
  - Violin
- Selection of notes for Tuning from drop down menu that plays automatically allowing users to accurately tune strings

Sound



UKULELE NOTE
A4

Display



VIOLIN NOTE
D4

Future considerations

- Add greater library of musical instruments
- Allow for push notifications when unideal conditions
- Allow for tuning to different scales
- Make packaging compact