AUTO-WINDOW

ME106 PROJECT: Spring 2021
By: Team JEE
Joshua Cooney, Ehsan Al-Agtash, Eduardo Molina

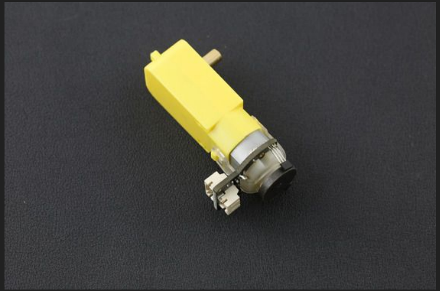
THE PURPOSE

- Making life easier, to open and close the window automatically
  - Manual and automatic modes
  - 3 sensors: time,gas,temperature

- Increased safety
- Less worry throughout the day

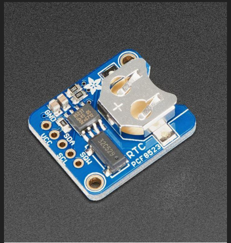
SPECIFICATIONS

- Window dimensions: 23.5 inches x 23.5 inches
- Two motors
- Motor 1: @7v
  - No load current: .17 A
  - No load speed: 160 RPM
  - Gear ratio: 120:1
- Motor 2 @7v
  - No load current: .17 A
  - No load speed: 160 RPM
  - Gear ratio: 120:1
  - Stall torque: .8 kgf-cm
  - Rated torque: .2 kgf-cm

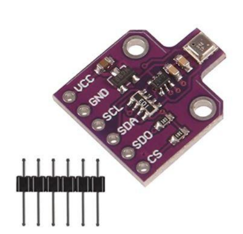


PRIMARY COMPONENTS

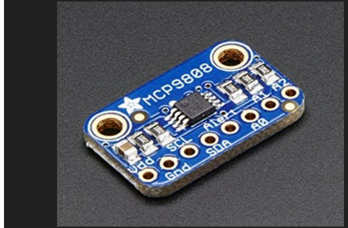
A. Real time clock sensor
B. Alcohol and VOC gas sensor
C. Temperature sensor
D. DC electric 6 volt motor
E. Belt pulley wheel
F. H-Bridge L298N
G. Nrf board
H. Power supply



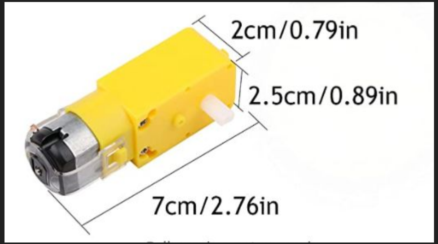
(A)



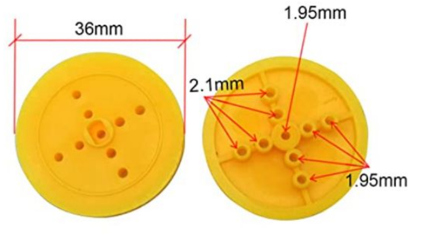
(B)



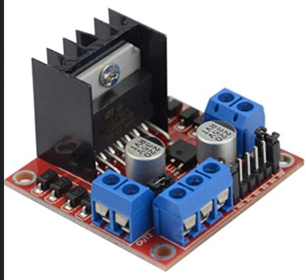
(C)



(D)

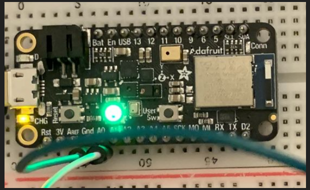


(E)



(F)

(H)


Bat En USB 13 12 11 10 9 6 5 SCL SDA
adafruit
CHG
Rst 3V Aref Gnd A0 A1 A2 A3 A4 A5 SCK MO MI RX TX D2

(G)

Cost Analysis

Goal

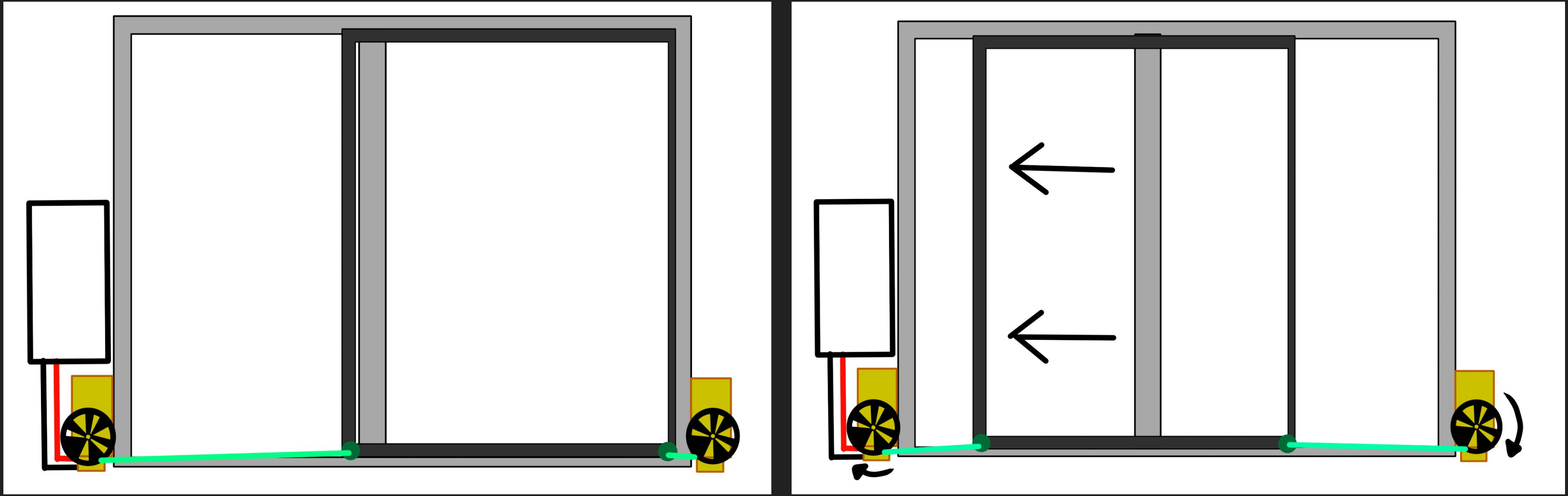
- Keep under $50
- Use most of the parts in kit
- Use one motor to move the window

Outcome

- Total cost of the prototype
  - $9- temp sensor
  - $5 - real time clock
  - $14- voc alcohol and gas sensor
  - $4- solid wire
  - $10- pulley wheel set
  - $7- two motors
- Total= $49

(shipping, mishaps, labor and other cost not included)

THE DESIGN

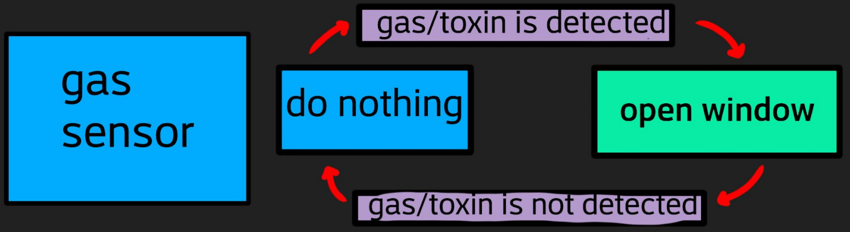


Simple design sketch: motors move clockwise to close and counter clockwise to open

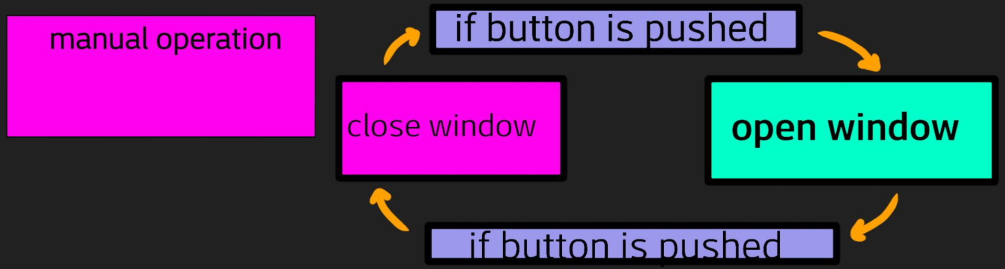
State diagram

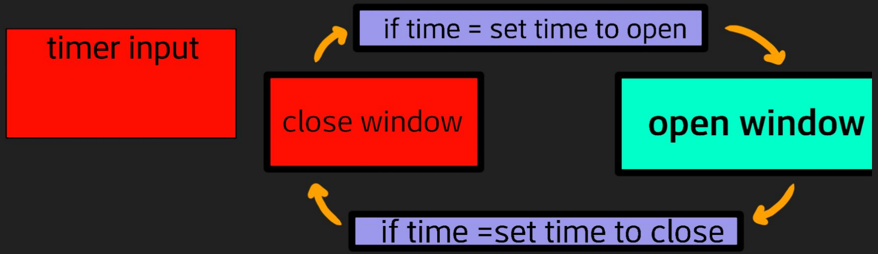
The state diagram with a priority directive A followed by B then C and finally D to open/close the window

(A)

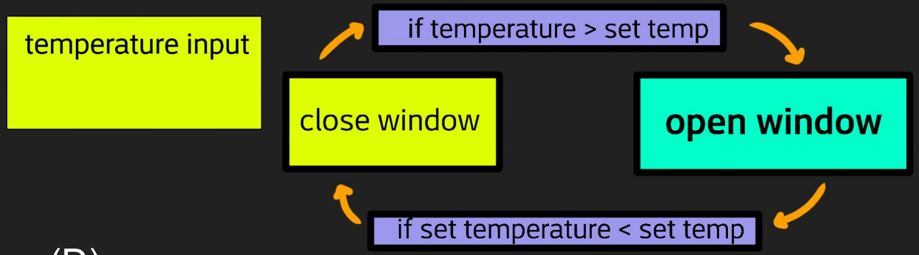


(B)



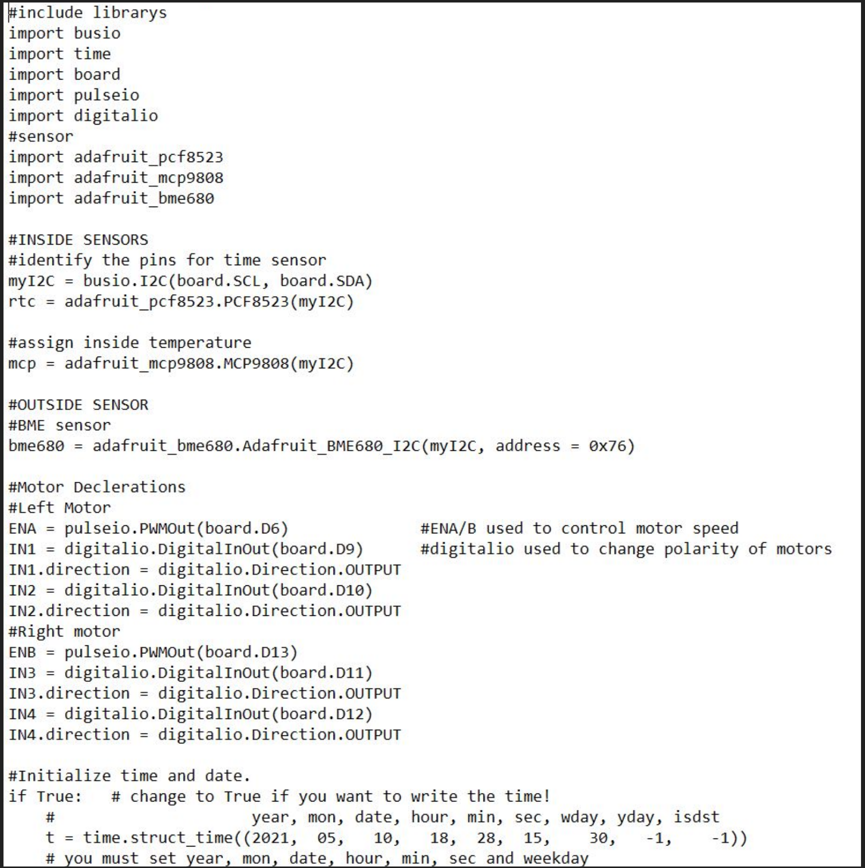


(C)

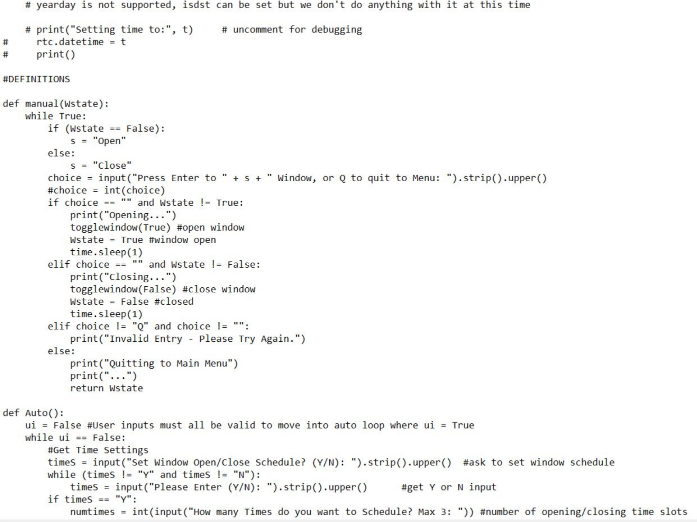


(D)

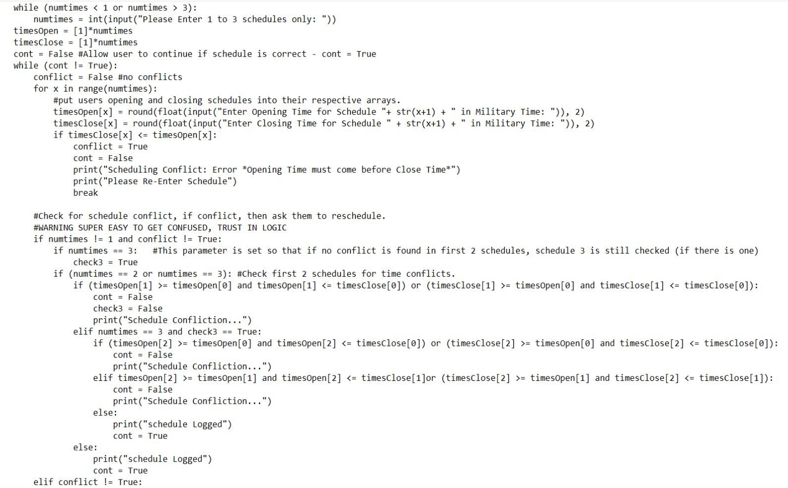
CODE



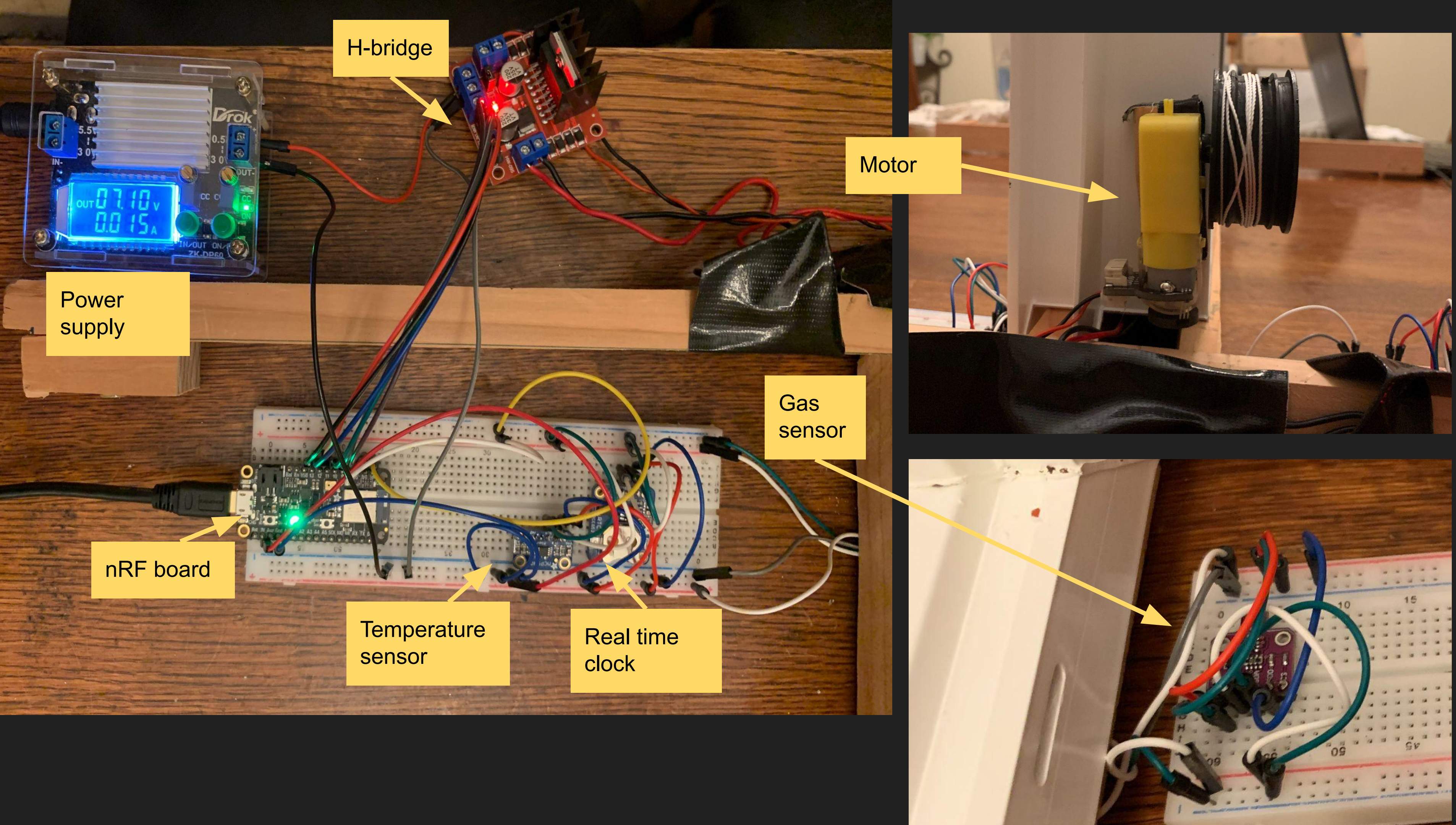
```
#include librarys
import busio
import time
import board
import pulseio
import digitalio
#sensor
import adafruit_pcf8523
import adafruit_mcp9808
import adafruit_bme680

#INSIDE SENSORS
#identify the pins for time sensor
myI2C = busio.I2C(board.SCL, board.SDA)
rtc = adafruit_pcf8523.PCF8523(myI2C)

#assign inside temperature
mcp = adafruit_mcp9808.MCP9808(myI2C)

#OUTSIDE SENSOR
#BME sensor
bme680 = adafruit_bme680.Adafruit_BME680_I2C(myI2C, address = 0x76)

#Motor Declerations
#Left Motor
ENA = pulseio.PWMOut(board.D6)              #ENA/B used to control motor speed
IN1 = digitalio.DigitalInOut(board.D9)      #digitalio used to change polarity of motors
IN1.direction = digitalio.Direction.OUTPUT
IN2 = digitalio.DigitalInOut(board.D10)
IN2.direction = digitalio.Direction.OUTPUT
#Right motor
ENB = pulseio.PWMOut(board.D13)
IN3 = digitalio.DigitalInOut(board.D11)
IN3.direction = digitalio.Direction.OUTPUT
IN4 = digitalio.DigitalInOut(board.D12)
IN4.direction = digitalio.Direction.OUTPUT

#Initialize time and date.
if True:   # change to True if you want to write the time!
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2021,  05,   10,   18,  28,  15,    30,   -1,    -1))
    # you must set year, mon, date, hour, min, sec and weekday
```



```
    # yearday is not supported, isdst can be set but we don't do anything with it at this time

    # print("Setting time to:", t)     # uncomment for debugging
#     rtc.datetime = t
#     print()

#DEFINITIONS

def manual(Wstate):
    while True:
        if (Wstate == False):
            s = "Open"
        else:
            s = "Close"
        choice = input("Press Enter to " + s + " Window, or Q to quit to Menu: ").strip().upper()
        #choice = int(choice)
        if choice == "" and Wstate != True:
            print("Opening...")
            togglewindow(True) #open window
            Wstate = True #window open
            time.sleep(1)
        elif choice == "" and Wstate != False:
            print("Closing...")
            togglewindow(False) #close window
            Wstate = False #closed
            time.sleep(1)
        elif choice != "Q" and choice != "":
            print("Invalid Entry - Please Try Again.")
        else:
            print("Quitting to Main Menu")
            print("...")
            return Wstate

def Auto():
    ui = False #User inputs must all be valid to move into auto loop where ui = True
    while ui == False:
        #Get Time Settings
        timeS = input("Set Window Open/Close Schedule? (Y/N): ").strip().upper()  #ask to set window schedule
        while (timeS != "Y" and timeS != "N"):
            timeS = input("Please Enter (Y/N): ").strip().upper()      #get Y or N input
        if timeS == "Y":
            numtimes = int(input("How many Times do you want to Schedule? Max 3: ")) #number of opening/closing time slots
```



```
while (numtimes < 1 or numtimes > 3):
    numtimes = int(input("Please Enter 1 to 3 schedules only: "))
timesOpen = [1]*numtimes
timesClose = [1]*numtimes
cont = False #Allow user to continue if schedule is correct - cont = True
while (cont != True):
    conflict = False #no conflicts
    for x in range(numtimes):
        #put users opening and closing schedules into their respective arrays.
        timesOpen[x] = round(float(input("Enter Opening Time for Schedule "+ str(x+1) + " in Military Time: ")), 2)
        timesClose[x] = round(float(input("Enter Closing Time for Schedule " + str(x+1) + " in Military Time: ")), 2)
        if timesClose[x] <= timesOpen[x]:
            conflict = True
            cont = False
            print("Scheduling Conflict: Error *Opening Time must come before Close Time*")
            print("Please Re-Enter Schedule")
            break

    #Check for schedule conflict, if conflict, then ask them to reschedule.
    #WARNING SUPER EASY TO GET CONFUSED, TRUST IN LOGIC
    if numtimes != 1 and conflict != True:
        if numtimes == 3:   #This parameter is set so that if no conflict is found in first 2 schedules, schedule 3 is still checked (if there is one)
            check3 = True
        if (numtimes == 2 or numtimes == 3): #Check first 2 schedules for time conflicts.
            if (timesOpen[1] >= timesOpen[0] and timesOpen[1] <= timesClose[0]) or (timesClose[1] >= timesOpen[0] and timesClose[1] <= timesClose[0]):
                cont = False
                check3 = False
                print("Schedule Confliction...")
            elif numtimes == 3 and check3 == True:
                if (timesOpen[2] >= timesOpen[0] and timesOpen[2] <= timesClose[0]) or (timesClose[2] >= timesOpen[0] and timesClose[2] <= timesClose[0]):
                    cont = False
                    print("Schedule Confliction...")
                elif timesOpen[2] >= timesOpen[1] and timesOpen[2] <= timesClose[1]or (timesClose[2] >= timesOpen[1] and timesClose[2] <= timesClose[1]):
                    cont = False
                    print("Schedule Confliction...")
                else:
                    print("schedule Logged")
                    cont = True
            else:
                print("schedule Logged")
                cont = True
    elif conflict != True:
```



H-bridge

Motor

Power supply

Gas sensor

nRF board

Temperature sensor

Real time clock

SCHEMATIC DIAGRAM

A. nRF

B. POWER SUPPLY

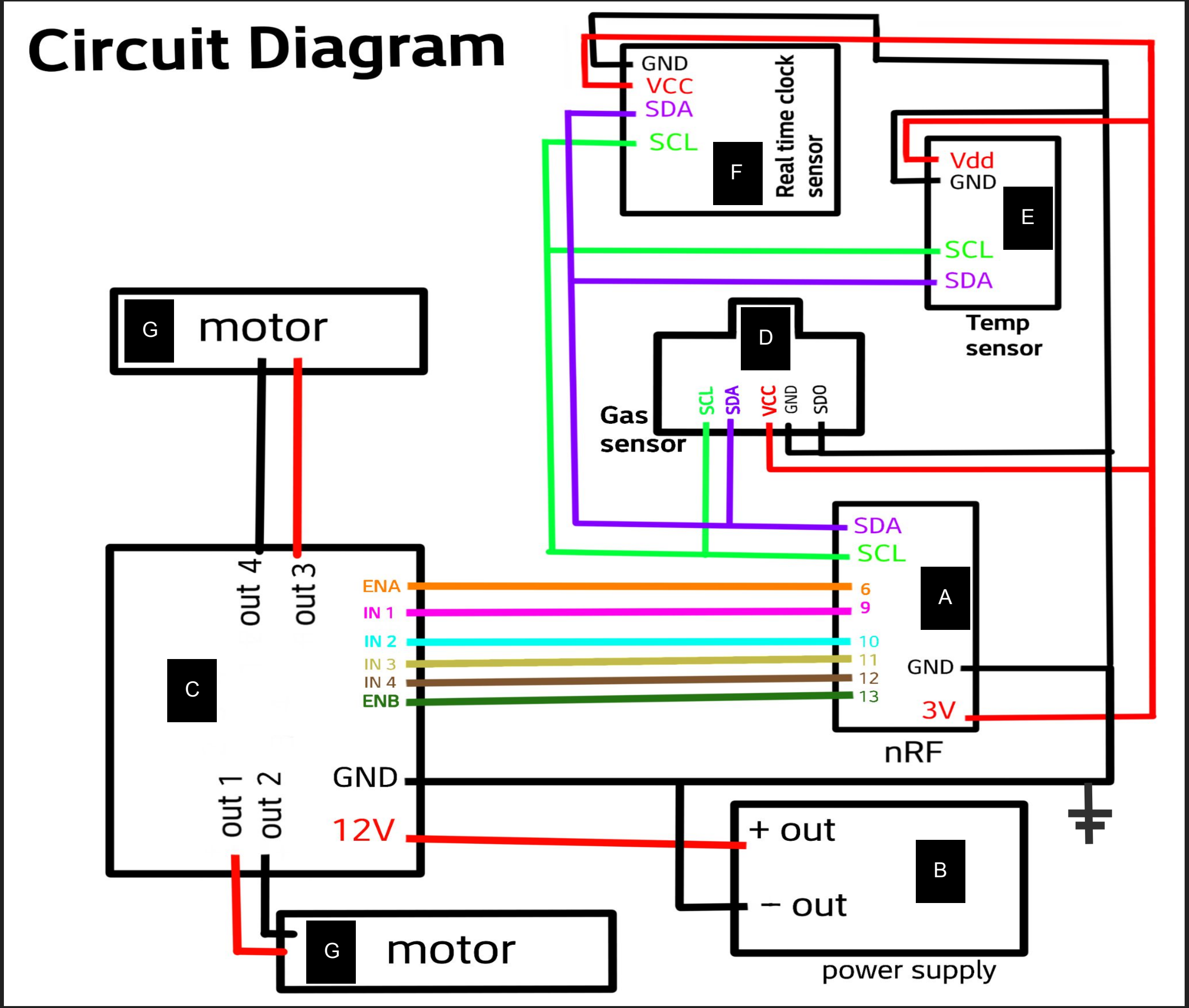
C. H-BRIDGE L298N

D. VOC ALCOHOL AND GAS SENSOR

E. TEMPERATURE SENSOR

F. REAL TIME CLOCK SENSOR

G. MOTORS



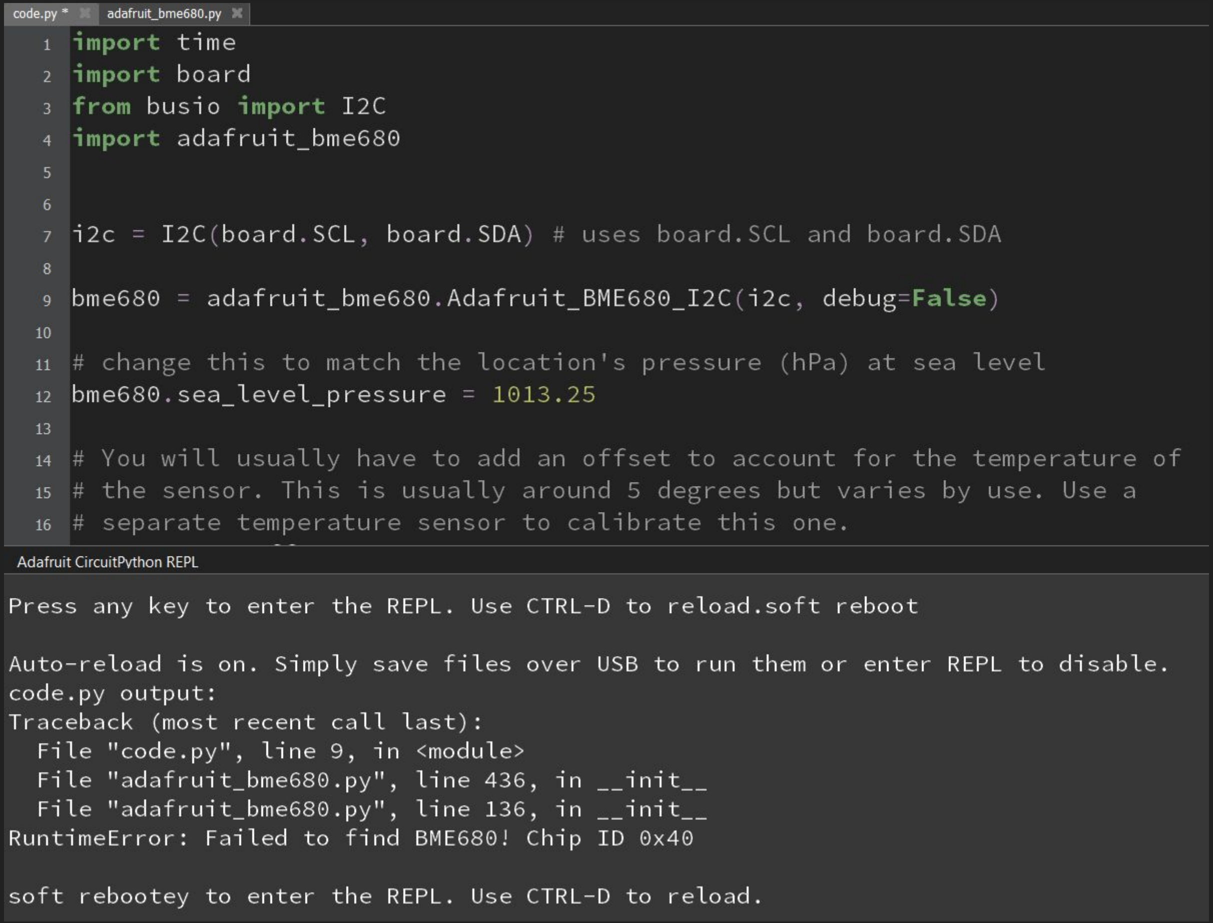
DEMO

CHALLENGES

- Cost
- Issues with having parts delivered
- Working on the project via zoom
- Schedule conflictions
- Soldering all the components
- Coding
- Going through gas sensors

The Gas Sensor

- Several error messages from both gas sensors
- Uses resistors to determine gas value, but was stuck at constant value
- Wrong address id default
  - Changed to ID 0x76

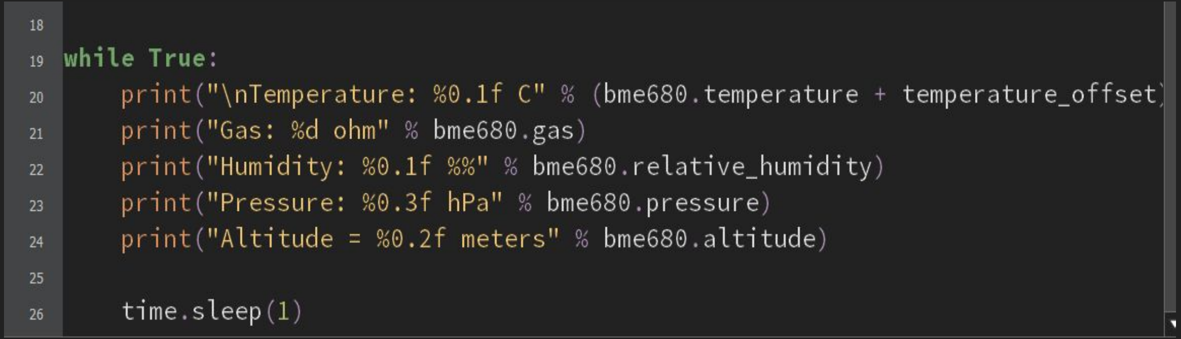


```python
import time
import board
from busio import I2C
import adafruit_bme680


i2c = I2C(board.SCL, board.SDA) # uses board.SCL and board.SDA

bme680 = adafruit_bme680.Adafruit_BME680_I2C(i2c, debug=False)

# change this to match the location's pressure (hPa) at sea level
bme680.sea_level_pressure = 1013.25

# You will usually have to add an offset to account for the temperature of
# the sensor. This is usually around 5 degrees but varies by use. Use a
# separate temperature sensor to calibrate this one.
```

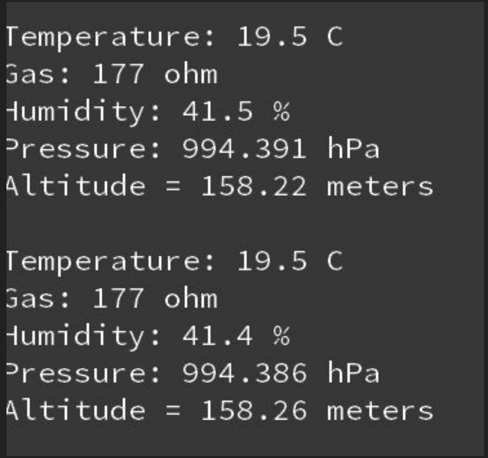
```
Press any key to enter the REPL. Use CTRL-D to reload.soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Traceback (most recent call last):
  File "code.py", line 9, in <module>
  File "adafruit_bme680.py", line 436, in __init__
  File "adafruit_bme680.py", line 136, in __init__
RuntimeError: Failed to find BME680! Chip ID 0x40

soft rebootey to enter the REPL. Use CTRL-D to reload.
```

Gas sensor continued

- Sdo pin to ground was one solution but another problem came up
- Gas output remained constant even introduced to 2 different samples
- Ultimately after tests concluded both were just defective



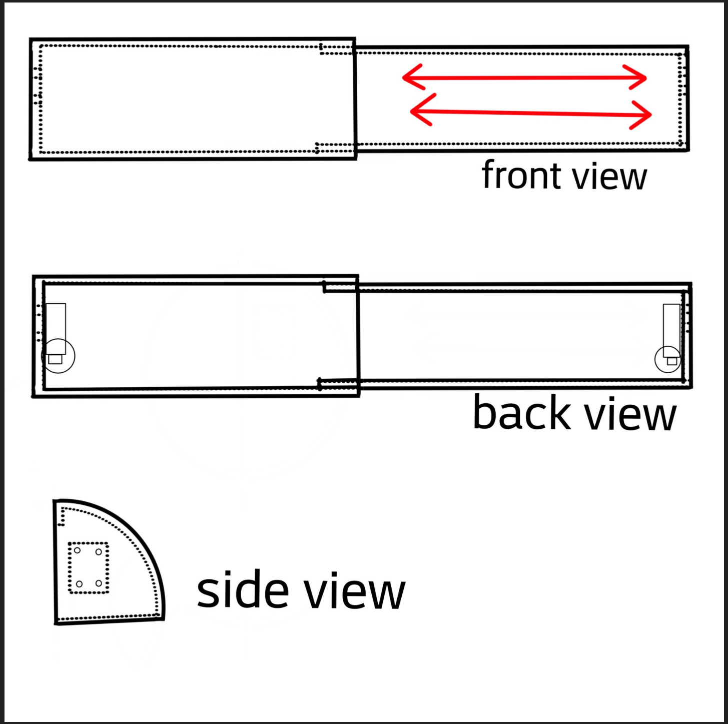
```
while True:
    print("\nTemperature: %0.1f C" % (bme680.temperature + temperature_offset)
    print("Gas: %d ohm" % bme680.gas)
    print("Humidity: %0.1f %%" % bme680.relative_humidity)
    print("Pressure: %0.3f hPa" % bme680.pressure)
    print("Altitude = %0.2f meters" % bme680.altitude)

    time.sleep(1)
```



```
Temperature: 19.5 C
Gas: 177 ohm
Humidity: 41.5 %
Pressure: 994.391 hPa
Altitude = 158.22 meters

Temperature: 19.5 C
Gas: 177 ohm
Humidity: 41.4 %
Pressure: 994.386 hPa
Altitude = 158.26 meters
```

CONCLUSIONS AND RECOMMENDATIONS

- Would create a housing for the device
- Make for easy installation and adjustment for most windows
- Adjust the design to only need one motor
- Include a feature to save set settings for ease of access
- Add a bluetooth function for control
- Manage time better



Housing design for the prototype

What was learned and challenges (eduardo molina)

What i learned was:

- How to manage time much more efficiently
- How to wire the microcontroller to allow the sensors to function
- How to interpret circuit schematics/ diagrams
- How to work collaboratively

Some challenges were:

- Remote meetings over zoom when testing
- Having to work with everyone's different schedules
- Balance- to do work on the project, as well as other courses and family responsibilities

What was learned and challenges (Joshua Cooney)

What I learned was:

- How to coordinate a project and work with my group over distance
- How to take a design and create a physical model
- How to better work with sensors and python to create an intuitive software experience
- How to solder components, and deeper understanding of circuitry

Some challenges were:

- Time and energy management
- Working in a less familiar programming language (Python)
- Distance between group mates and having only one physical model to test with.
- Gas sensors not working properly despite extensive troubleshooting

What was learned and challenges (Ehsan Al-Agtash)

What I learned was:

- Connecting 3 sensor together using the same pins
- Wire management
- How import is it to design and do rough sketches before rushing to do a prototype
- How important testing is before building the prototype
- Reading circuit diagrams, pulling up motor specifications
- Soldering pins to boards

Some challenges were:

- Prioritizing school work and projects
- Time managements
- Working over zoom and rarely meeting up
- Struggling to get the VOC/Gas sensor to work

QUESTIONS?