Smart Vent Project

Shazil Razzaq

SCVAIR

Introduction:

Although central air-conditioning units are constantly becoming more efficient and effective every year, one of the biggest issues with these types of air conditioning systems has not been resolved yet. This is the problem of airflow. Regardless of whether one lives in an older or newer house, the upper floor is always hotter in the summer. This is because the further the air travels through the ducts, the more the airflow weakens. As a result, when it comes time to cool the room, the air barely reaches the target room. What started as a summer idea shaped up to become the best solution available for this problem, the smart vent. The SCVAIR smart vent is an affordable way to make the users home comfortable on any floor at a significantly cheaper price compared to purchasing single room air conditioning units.

What Problems Does it Solve?

Central air-conditioning units are well known for their issues when it comes to air flow to upper floors regardless of their size. Many people not knowing this spend thousands of dollars in new HVAC only realizing after that there is barely any difference. Or, some spend thousands on single room air conditioning, which costs more initially and more to run. The smart air vent solves the problem of air flow by boosting the idle air from the ducts towards the rooms where it is needed, and blocks air going to unnecessary rooms. Along with this, there are additional features such as being able to control the vent with a phone or PC, and debris protection which helps cut costs for duct cleaning. As a result of these features, not only does it make it more comfortable for the user, but also is better for the environment.

How Does it Work:

The smart vent consists of both software and hardware components. When it comes to hardware, it is comprised of 4 major components. A 90mm 12V fan, low rpm motor, 3D printed part and temperature sensor. The fan is attached to the vent which will be used to boost idle air into the room. The temperature sensor module is attached inside the enclosing for the vent and provides the temperature data to the software and tells the vent to open and close and tells the fan to turn on and off. The low rpm module will be used alongside the 3D printed component which is used to move the flaps. The

3D printed part is a precisely measured and well designed L shape part that is key to opening and closing the flaps which I made with Autodesk Inventor. In terms of software, the vent is connected within the IoT (Internet of Things) through the use of Cayenne. Cayenne is a tool which allows for anyone to access their vent controls and lets them set up their own schedules. These elements cohesively come together to provide a great user experience and will both efficiently and effectively solve a common household problem.

The Cost:

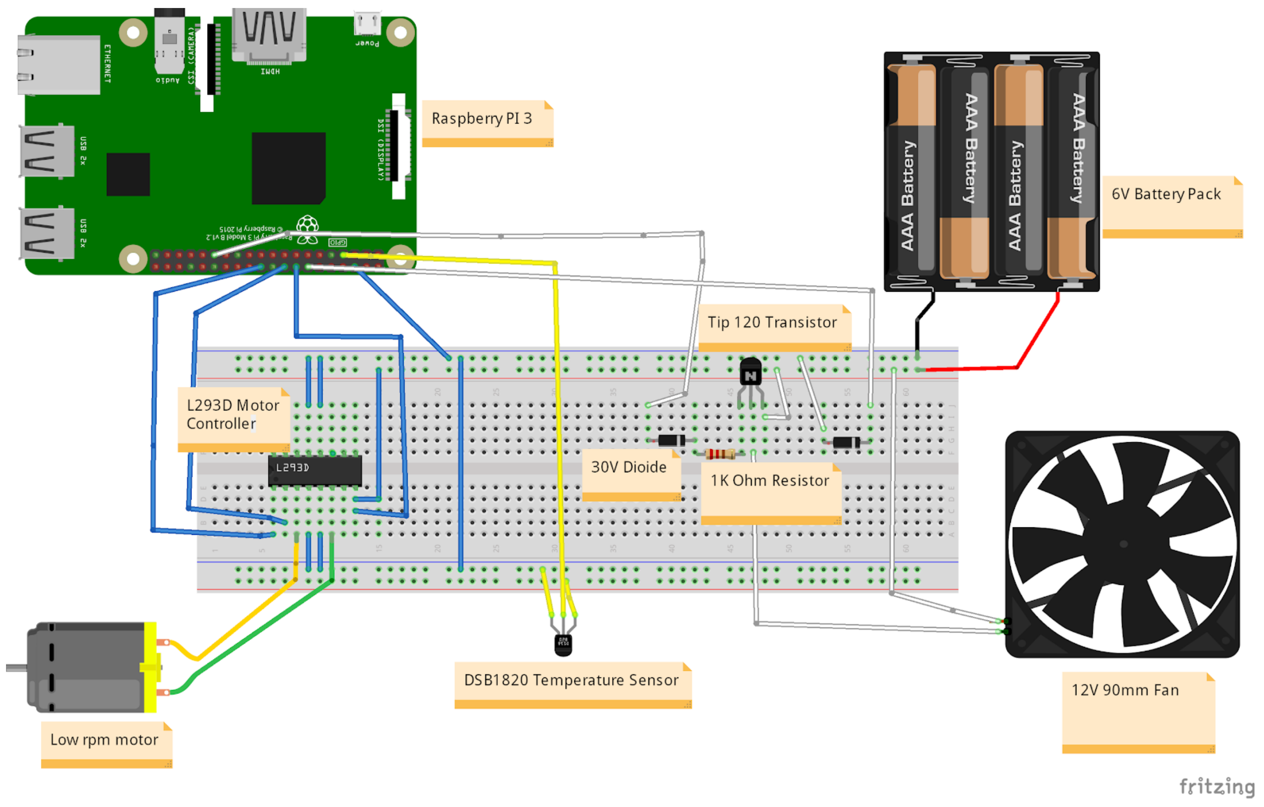
Part	**Cost**	**Supplier**
12V Fan	$10	Amazon
Low RPM Motor	$12	Banggood
Temperature Sensor	$8	eBay
3D Printed Flap Mover	$5	Library
2 30V Transistors	$2	Banggood
Glue Gun	$0	Me
1K Ω Resistor	$0	School
Glue Sticks	$3	Dollar Store
Construction Paper	$2	Dollar Store
Raspberry Pi 3	$0	Me

Total Cost Incl Taxes: $35.40

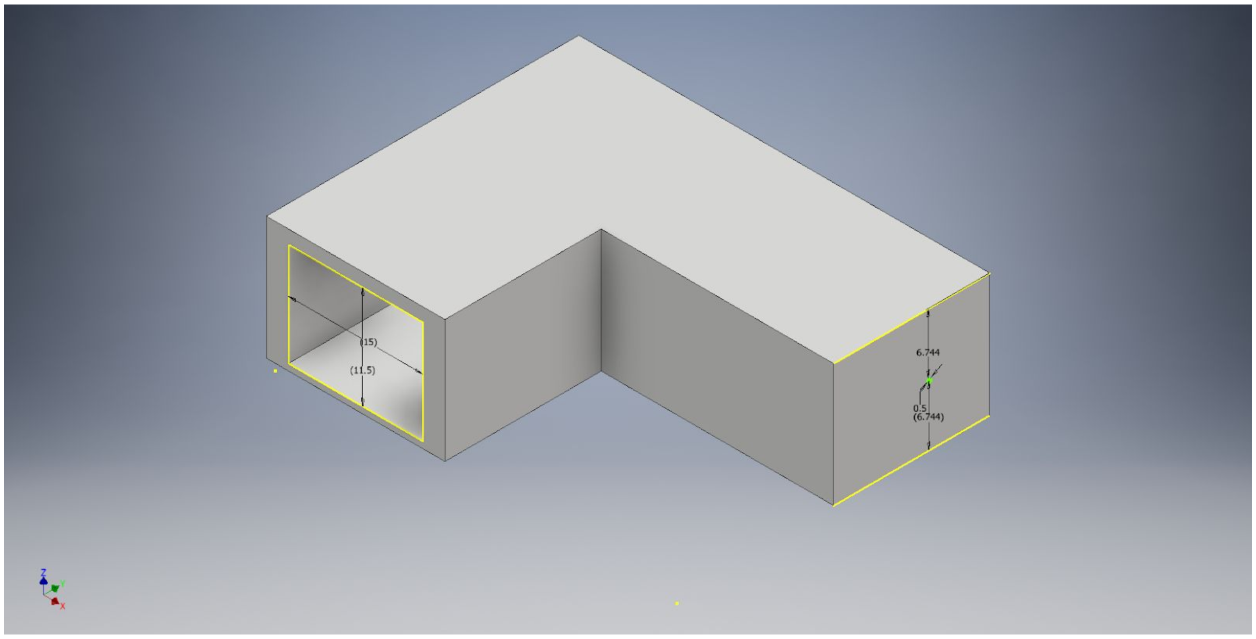
The Hardware Build:

The hardware used here is paramount to making sure all functions of the smart vent work as efficiently and effectively as possible.

A key component that was necessary in making this a success was Fritz. Fritz allowed me to troubleshoot any problems in a timely manner. Here is the labelled Fritz circuit schematic diagram:



An extremely crucial part is the 3D printed L-shape flap controller that I designed from the ground up using Autodesk Inventor. It was measured precisely to provide complete opening/closing functions of the vent. Here is a picture of the part:



Why Raspberry Pi?

The Raspberry Pi was used over an Arduino due to its ability to connect within the IoT without a WiFi chip. It was also used because of it being able to do multiple things really well at once. This means that when given more time and resources, refinements and new features can be added without a problem.

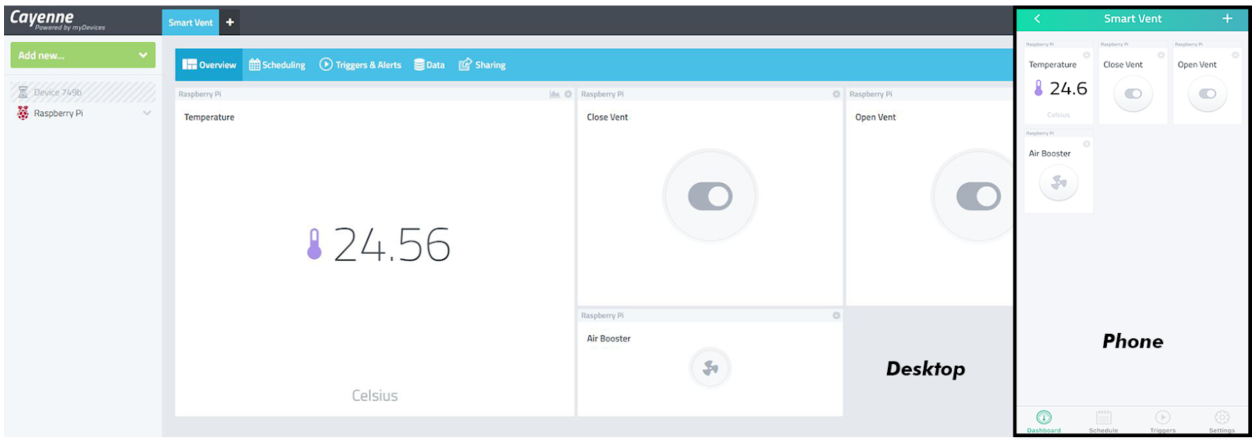
Why a Low RPM Motor?

A low RPM motor was used because of it being able to have a high amount of torque. This is more important that speed because moving the vent requires a high amount of torque. Speed would only be a bonus if the distance between the open and close flaps was significantly larger.

The Software:

In addition to the hardware components of the smart vent, the right software was required to create a seamless experience for the user.

In order to connect the smart vent to the IoT, Cayenne was used. It allows the vent to be controlled from anywhere around the world with your phone and PC. It allows the user to control the flaps, monitor the temperature, and turn the fan on/off. Additionally, it has a scheduling mode which allows the user to setup a time and date of when they want that vent open. This can come in handy in places such as home gyms. It also includes a notification system, notifying the user of the current state of the vent. A screenshot of Cayenne running on a desktop and phone is shown below.



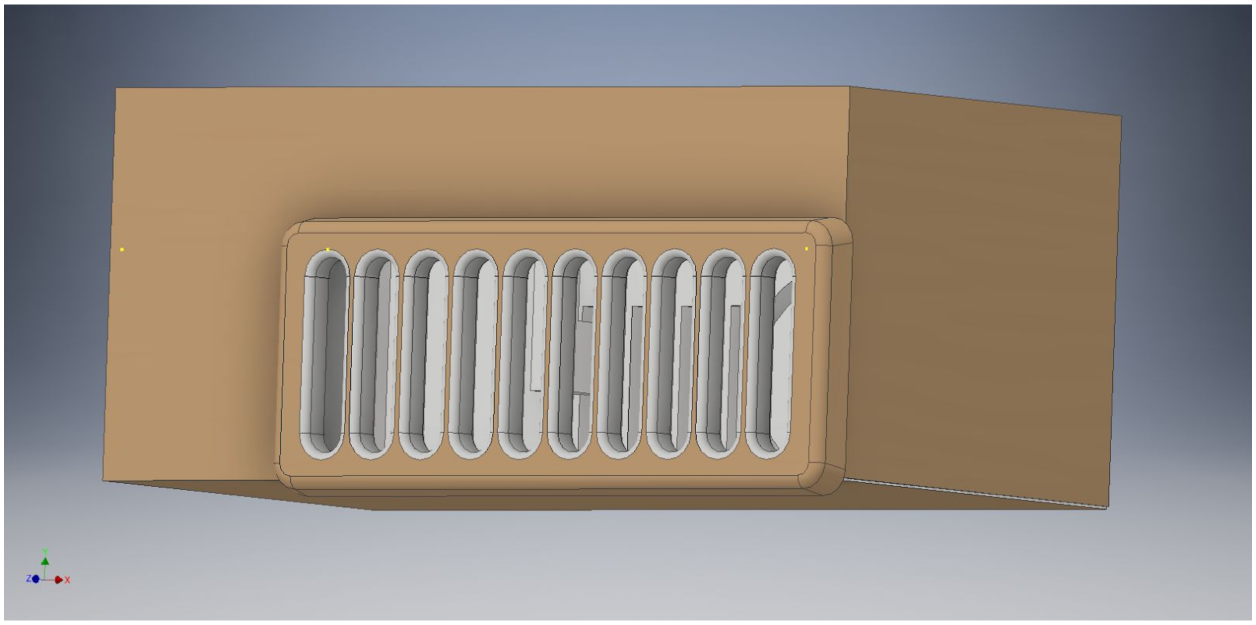
In order to enable autonomous mode for the vent, I created a program in Python. This program allows for the user to just drop their vent in the floor register and forget about it. It works by calibrating the temperature in the room and accordingly will open/close the vents, and turn the fan on/off. The code is shown below with comments to help you understand.

```
import RPi.GPIO as GPIO
from time import sleep
import os

#This sets it to GPIO mode
GPIO.setmode(GPIO.BCM)

#GPIO Pins where hardware is connected
MotorA = 23
MotorB = 24
MotorE = 25
```

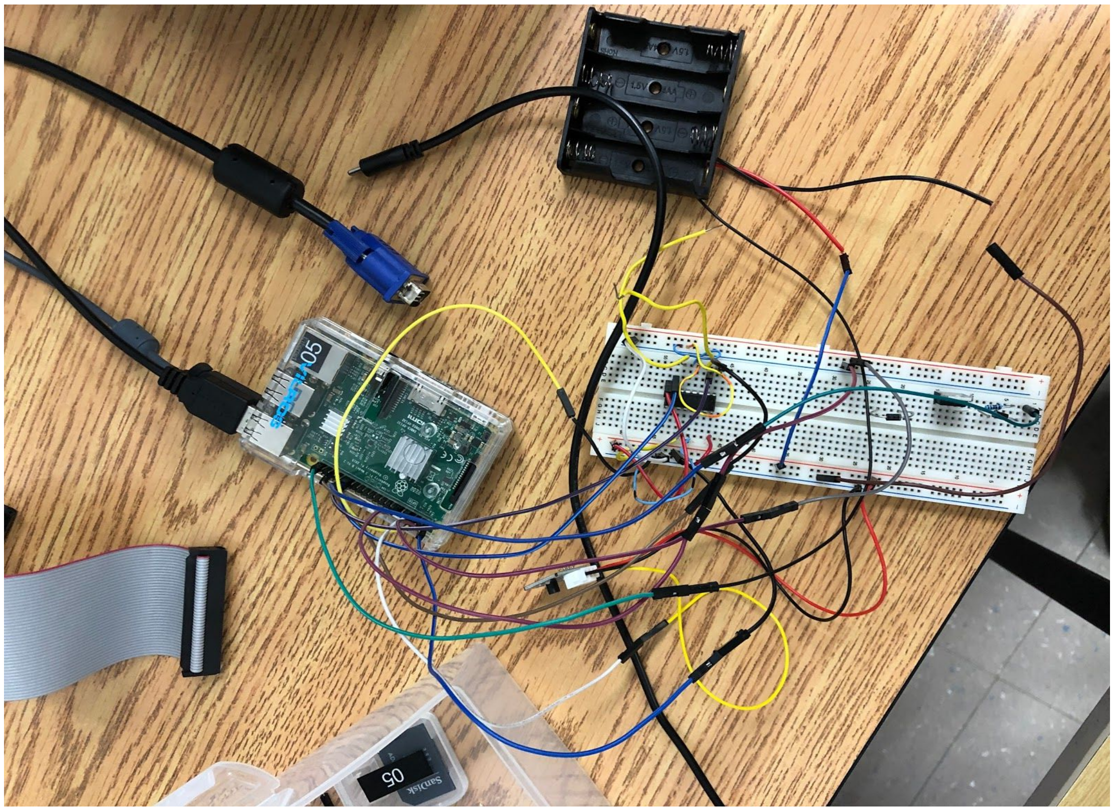
```
Fan = 5

#Setup devices
GPIO.setup(MotorA,GPIO.OUT)
GPIO.setup(MotorB,GPIO.OUT)
GPIO.setup(MotorE,GPIO.OUT)
GPIO.setup(Fan, GPIO.OUT)

#Temperature Sensor from Sunfounder
GPIO.ds18b20 = '28-0000093e7173'

def setup():
    global ds18b20
    for i in os.listdir('/sys/bus/w1/devices'):
        if i != 'w1_bus_master1':
            ds18b20 = i

#Reads text file for temperature readings
def read():
#   global ds18b20
    location = '/sys/bus/w1/devices/' + ds18b20 + '/w1_slave'
    tfile = open(location)
    text = tfile.read()
    tfile.close()
    secondline = text.split("\n")[1]
    temperaturedata = secondline.split(" ")[9]
    temperature = float(temperaturedata[2:])
    temperature = temperature / 1000
    return temperature

#Vent autonomy
def loop():
    #Calibration
    prevTemp = read()

    while True:
        x = read()
        #A/C is on
        if (x<prevTemp+1):
            #Fan on
            GPIO.output(Fan,True)

            #Motor Controls
            GPIO.output(MotorA,GPIO.HIGH)
            GPIO.output(MotorB,GPIO.LOW)
            GPIO.output(MotorE,GPIO.HIGH)

        #A/C is off
        Else:
            #Fan off
            GPIO.output(Fan,False)

            #Motor controls
            GPIO.output(Motor1A,GPIO.LOW)
            GPIO.output(Motor1B, GPIO.HIGH)
            GPIO.output(Motor1E, GPIO.HIGH)

def destroy():
    pass

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        destroy()

GPIO.cleanup()
```

How it All Came Together - Progress Pictures

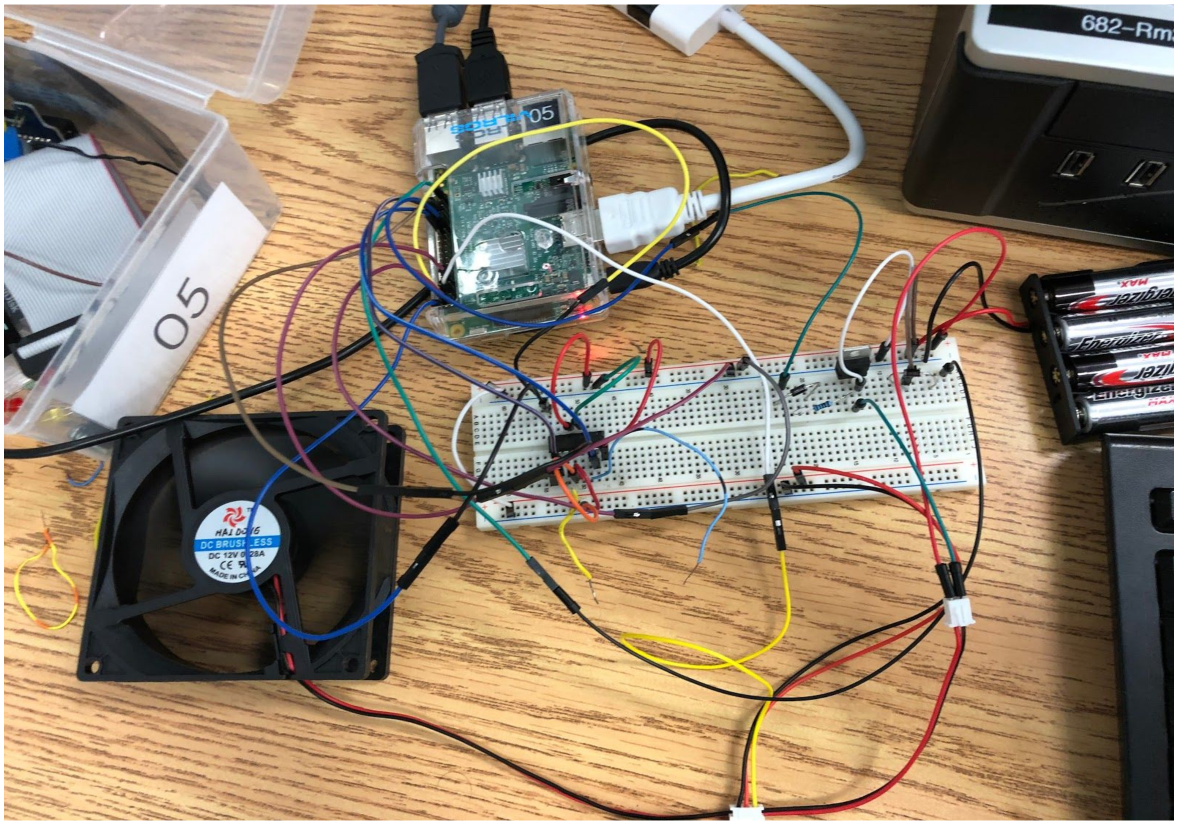
I used Autodesk Inventor to help me set an overall goal of what the smart vent should look like. Here is a picture.



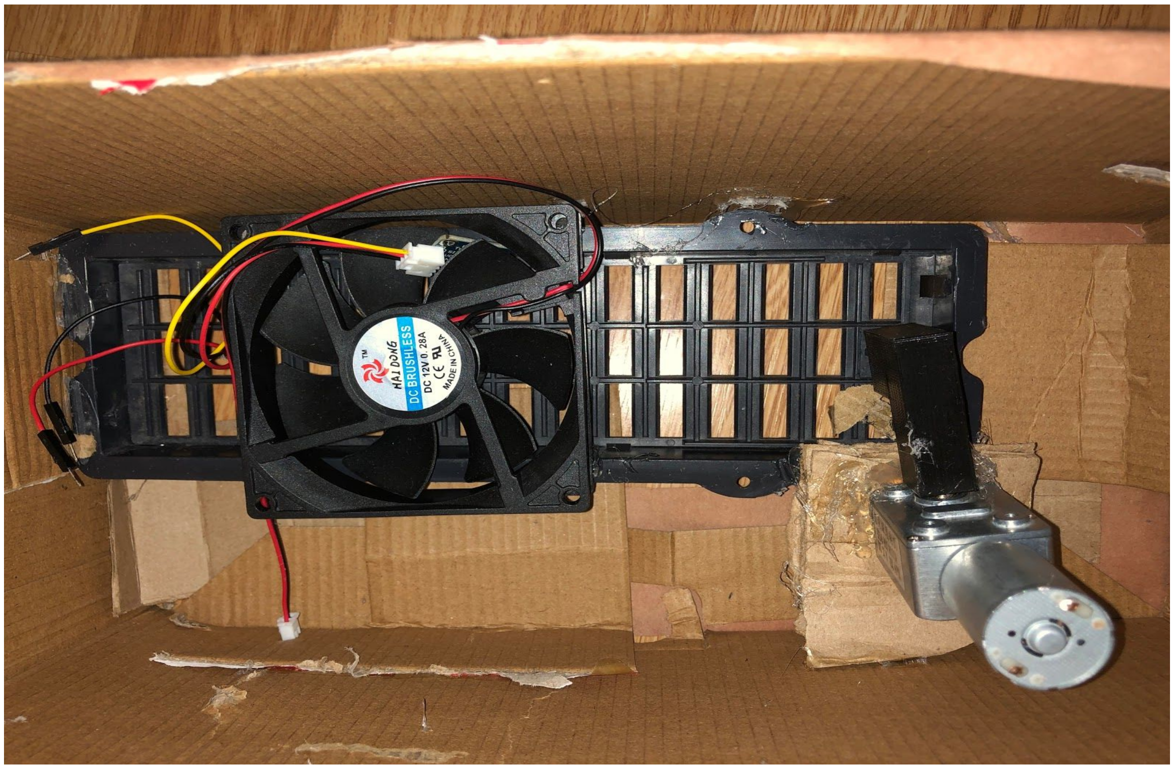
This is a picture of the motor and temperature sensor components being implemented.



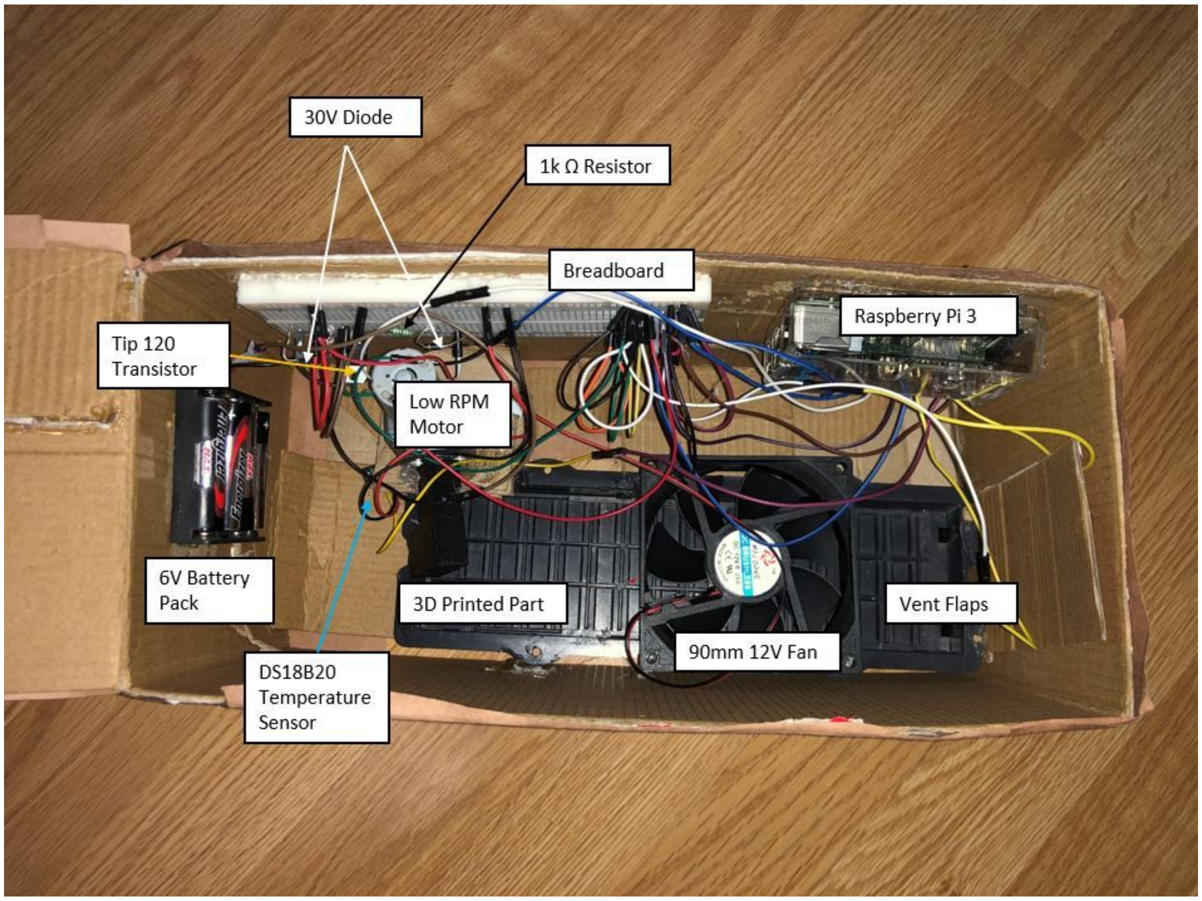
This is a picture of the fan and motor components being implemented.



This is a picture of the enclosure part of the smart vent being worked on.



This is a labelled picture of the final build of the smart vent.



What problems did you have and how did you fix them?

Despite this being a seamless implementation for the user, there were some hurdles along the way. For example, the 3D printed part was time consuming because of how precise it had to be. In an effort to save time and resources, I worked on the Autodesk Inventor drawing until the dimensions were perfect for the vent so it can completely open/close. Another problem was the amount of components required given the space. This made it very hard to implement the features into a smaller design. To fix this, I first connected the bulkier components such as the fan, vent, and motor before handling with the circuitry. To make it easy to access, I purposely placed the breadboard, Raspberry Pi, and battery pack towards the front for easy access. In fact, I also created a flap on the top of the box that allowed me to troubleshoot even with the vent cover on top.

What Would You Do Differently Next Time?

Next time, I would work on a better solution for cable management; one that allows for tampering but is still kept nicely in place without being very messy. This is because of the difficulty involved with wiring many components in a small space.

Future Improvements

Given more time and resources, I would work on creating a more compact circuit by working with companies such as Shape Products. Also, I would implement more fans in the vent to boost the air flow even more. Additionally, I would improve the IoT connection by implementing Google Home, Siri, and Amazon Alexa support. To improve the vent's accuracy and speed of opening and closing, I would implement Nest thermostat support in order to tell the vent when to turn the fan on/off and open/close the flaps. I would also work on supporting different floor register sizes in order to fit a larger market. To add, I would increase the marketing of the product to showcase it to more people.

Conclusion:

Overall, the SCVAIR vent is an exceptional way to improve the user's home environment by making it more comfortable for all floors. It features a smart flap, air booster, IoT connection, and automation. These features come together to provide a great experience and benefit for the user.

GoFundMe: https://www.gofundme.com/smart-vent/

Credits:

https://www.hackster.io/stuman2000/control-a-12-volt-fan-with-a-rasberry-pi-and-a-transistor-53c06c

https://www.w3schools.com/howto/howto_js_slideshow.asp

https://webgradients.com/

https://www.sunfounder.com/learn/lesson-17-ds18b20-temperature-sensor-sensor-kit-v1-0-for-pi.html