Mini Project - Roof Vent

Steven Zadourian

07/12/2020

1 Description

California is known for its seasonal fires that can cause up too billions of dollars of damage to the state. Every year people loose homes, sentimental items, and much more due to these wildfires. One of the major fire vulnerabilities that every household has are roofs that are designed to ventilate the attics of households which have a variety of benefits from longevity to energy savings. However these vents provide easy access for fires ember to come into contact with housing insulation resulting in devastating damage. As an attempt to solve this problem, I've decided to built a prototype that utilizes the benefits of IoT and mechanical design to add an additional level of security to homes during wildfire seasons. To do this, my prototype is an automated vent that is designed to cut attic circulation during an event of an wildfire. Since households contain multiple vents, a ideally a Raspberry Pi will act as a hub, monitoring and controlling multiple vents. For simplicity for this project, my phones hot spot allows my phone to act as the hub and the Blynk App allows for a user interface.

2 Mechanical Details

2.1 Rotor

It is crucial for the rotor to be as light as possible and have minimal surface friction to reduced the load that the motor has to compensate for since the motor will have to overcome inertia, surface friction and the spring mechanism. To do this, a very thin piece of cardboard was wrapped with aluminum foil. The aluminum foils' smooth finish helped minimize surface friction and allowed me to use the rotor as common ground in respect to the stopping pins. This is discussed in more details in the electronic portion.

2.2 Spring Mechanism

The spring mechanism is to mechanically close and secure the vent in cases of power outages which are fairly common during windy days. To pick a spring, I went to the local hardware store and bought a couple reasonable springs that needed to be experimented with.

2.3 Pin and Rail

Multiple pin and rails are designed to keep rotor plate tight against the frame minimizing deflection. This improves the dynamics and would minimize the risk of ashes slipping through gaps.

2.4 Motor Mount

The movement is fairly simple, motor and rotor move together because the motors shaft is mounted on the main frame. The motor is mounted to the rotor with the mount seen in Figure , this mount was made from scrap metal found the garage, followed by epoxied (J-B Weld) and screwed together. Ideally this component would be spot welded together or printed as a whole from a 3D printer. The motor shaft hub was also epoxied to the main frame. Ideally spot welding or fasteners on a metal frame would be a desirable option but due the composition of the foam board, epoxy was chosen.

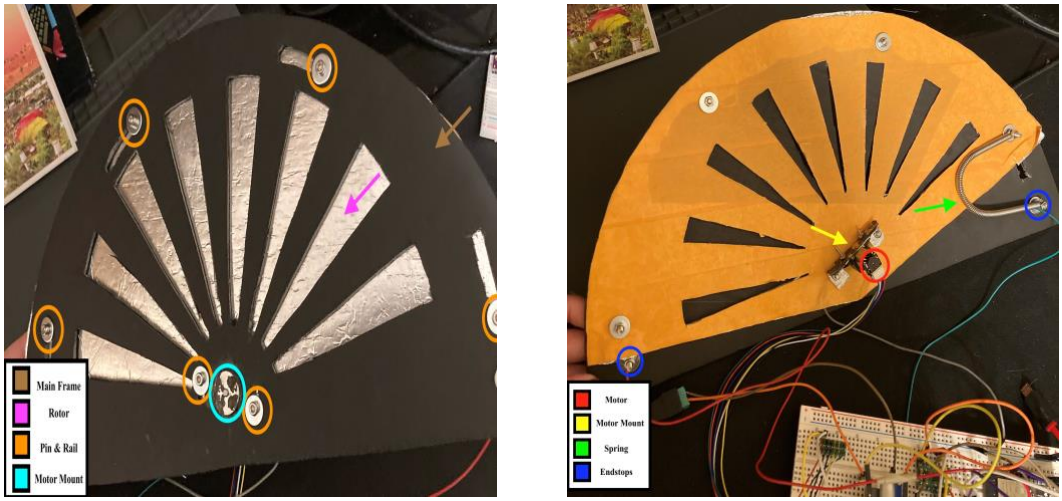

Figure 1: Front and Back

3 Electronic Details

3.1 Circuit

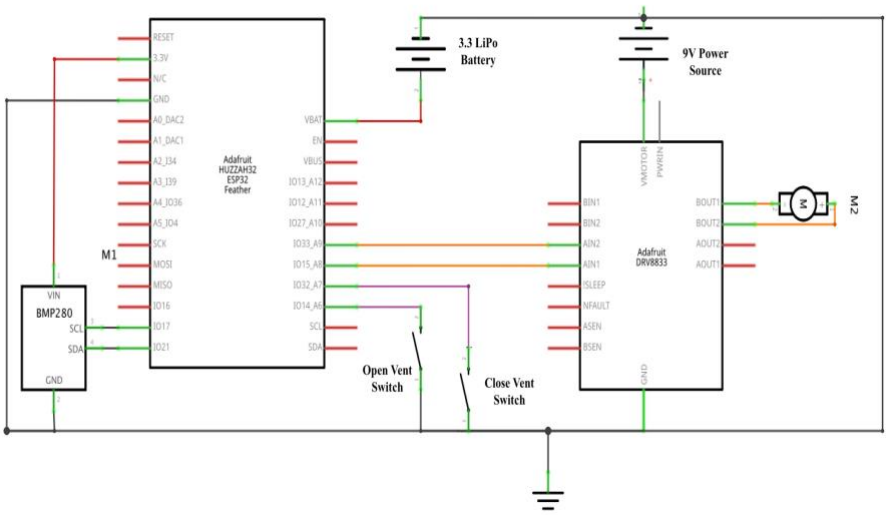

Figure 2: Circuit Made in Fritzing

3.2 Motor

The motor from the microkit is used to reduce cost and is driven is driven by the DRV8833 motor driver which was also included in the microkit. It is important to note that this motor is not rated strong enough for this application. It is rated for 6V and 9V had to be applied to get the rotor to move a desirable manner. If tools and cost were not a problem, using a much stronger geared motor or utilizing a lever mechanism is more desirable than the current implementation.

3.3 Pin Sensors

Both stopping pins are attached to a pull up pins on the Huzzah feather and when either one makes contact with the rotor it, they get sent to ground. These pins are essentially switches that provide feedback to determine whether the rotor is fully opened or closed.

3.4 Environment Sensor

For this sensor I chose the HiLetgo BME280 simply because it was av to run on 3.3V and it was available on prime shipping. To use this sensor I set up an I2C protocol that samples temperature, pressure, humidity and altitude every 1 seconds. After every sample the data is pushed back into Blynks' API allowing the user to monitor historical and live stream the data through the app.

4 Software Details

Most houses have multiple vents, it is desirable for a hub to connect too all the vents but provide a local access point to control them when the internet is down. In this project my phones hot spot acts as the 'hub' and the Blynk app acts as the interface for. In figure , we the following:

1. **LCD:** Tells the user whether the rotor is open or closed based on the pin switches. This is important because it confirms the rotors position after its been set by the switch.
2. **Switch:** Changes the state from open to close
3. **Bar:** The PWM value that is being applied to the motor
4. **Super Graph:** Time series data updated every 10 seconds of current temperature, humidity and hall sensor of ESP32

Figure shows the enlarged live graph, that updates environmental settings every 1 second. The gradual temperature decrease is due to the sensor cooling since I warmed it up with my finger for a few seconds beforehand.

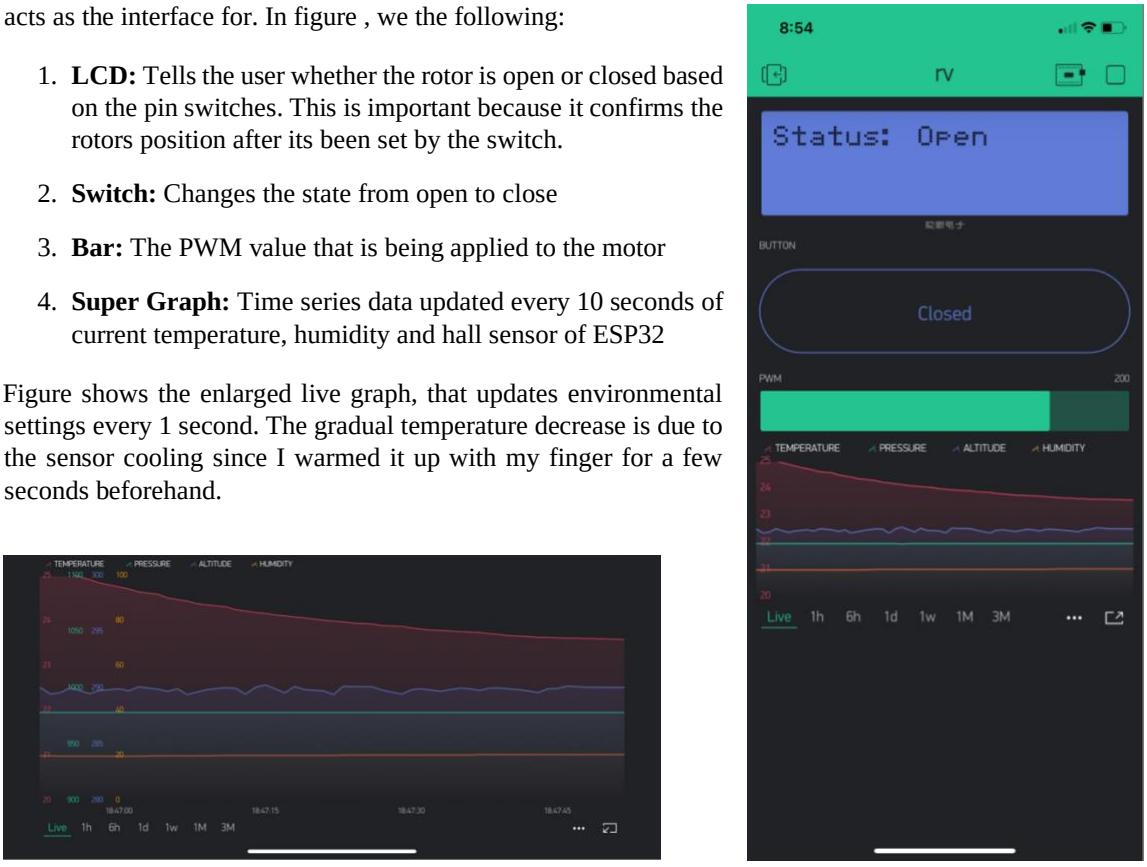

Figure 3: Blynk Application interface. Enlarged smart graph (right). Regular portrait mode (left)

5 Finite State Machine

There are two main stages in this machine, first when the micro controller is initially booted, it waits until a successful connection with my smartphone is established. Once the connection is established, the micro controller is either in open or closed state. Within either states, a 10 second interrupt is used to sample the environmental sensor.

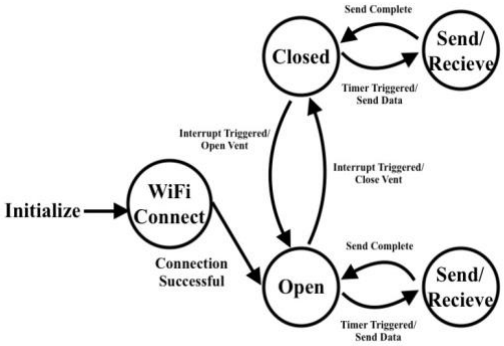

Figure 4: Finite State Machine

6 Challenges

6.1 Frame

There was a handful of challenges that came from making this project at home. Since I dont have access to the maker-space, I had to do everything with the limited tools that I have in my garage. Such cutting the frame and rotor by hand with a X-acto knife. Cutting the vent design right was extremely difficult and the way I approached it was designing on Solidworks and printing it out a sheet of paper to use as a guide. Even with this is was difficult to achieve clean straight lines. This process would have been extremely faster if I could have had access to a laser cutter.

6.2 Spring Mechanism

Finding a spring from a hardware store that had just the right amount of force for the motor to counteract and push the rotor closed was also very difficult. I tested them all and none of them worked, so I picked the one that felt best and extra elongated it to make it weak enough for the motor to counter act.

6.3 Surface Friction

Surface friction coupled with the interior of the motor is very difficult to overcome. Through trial and error I found the aluminum foil covered thin cardboard was the best option given the items that I were available to me.

7 Future Work

7.1 Raspberry Pi

To manage multiple vents ideally a Raspberry Pi will be used as a local hub that uploads data to the internet but is also accessible without the internet. Having a centralized system can due routine check ups to make sure all vents are operating accordingly. Any issues can be reported to the home owners.

7.2 A Bigger Prototype

I would ultimately like to purchase a standalone vent from a local hardware shop. Then design a robust metal prototype with a much stronger motor.

7.3 For Future ME102B Remote Students

Checkout the application Blynk for future students! Really easy way to provide hands on experience with IoT.

Vent Code

Steven Zadourian

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include "Adafruit_BME280.h"

#define BLYNK_PRINT Serial
#define I2C_SDA 21
#define I2C_SCL 17
#define close_switch_pin  14
#define open_switch_pin 32
#define motor_pin_a 15
#define motor_pin_b 33
#define SEALEVELPRESSURE_HPA (1013.25)
#define BME280_ADD 0x76

//Blynk Authorization
char auth[] = "-nsLI0d-0XU0w8IKPUWxHvJWIQTNZhuv";
// My phones hotspot
char ssid[] = "Test"; `
char pass[] = "test12345";

int open_state = 0;
int close_switch = 1;
int open_switch = 0;
int inital_pwm = 200;
int motor_pwm = 200;

Adafruit_BME280 bme(I2C_SDA, I2C_SCL);

// Sampling timer for sensor data
BlynkTimer timer;

// This reads the value of the switch
BLYNK_WRITE(V1) //Button Widget is writing to pin V1
{
  open_state = param.asInt();
}

void setup() {
  // Start Serial
  Serial.begin(115200);
  // Start Blynk
  Blynk.begin(auth, ssid, pass);
  // Setting up I2C
  bool status = bme.begin(BME280_ADD);
  while (!status) {
    status = bme.begin(BME280_ADD);
  }
  // Setup endswitches
  pinMode(close_switch_pin, INPUT_PULLUP);
  pinMode(open_switch_pin, INPUT_PULLUP);
  //Setup PWM
  ledcAttachPin(motor_pin_a, 1);
  ledcAttachPin(motor_pin_b, 2);
  ledcSetup(1, 5000, 8);
  ledcSetup(2, 5000, 8);
```

```
  //Setup Timer
  timer.setInterval(1000L, myTimerEvent);
}

// This is my timer interupt that samples sends the values to the graph
void myTimerEvent()
{
  Blynk.virtualWrite(V6, bme.readTemperature());
  Blynk.virtualWrite(V7,  bme.readPressure() / 100.0);
  Blynk.virtualWrite(V8, bme.readAltitude(SEALEVELPRESSURE_HPA);
                     Blynk.virtualWrite(V9, bme.readHumidity());
}

/* Ideally this code was meant to adjust the motor PWM accordinly to keep the vent
open.
  Although the only way to keep my vent open was to pump 9v into the motor.
*/
void keep_open() {
  if (digitalRead(open_switch_pin) == HIGH) { //Shorting the pin the ground
    Blynk.virtualWrite(V0, "Status: Closed"); // Updates the 'LED' Screen on blynk
    ledcWrite(1, 255);
    ledcWrite(2, 0);
  } else {
    Blynk.virtualWrite(V0, "Status: Open"); // Updates the 'LED' Screen on blynk
    motor_pwm -= 1;
  }
  Blynk.virtualWrite(V2, motor_pwm); //This updates the current PWM portion on the app
}

void loop() {
  Blynk.run(); //Initiates Blynk in general
  timer.run(); // Initiates BlynkTimer
  switch (open_state) {
    case (true):
      keep_open();
      break;
    default:
      motor_pwm = 200;
      ledcWrite(1, 0);
      ledcWrite(2, 0);
      if (digitalRead(close_switch_pin) == LOW) { //Shorting the pin the ground
        Blynk.virtualWrite(V0, "Status: Closed"); //This writes to the LED Screen
      } else {
        Blynk.virtualWrite(V0, "Status: Open"); //This writes to the LED Screen
      }
  }
}
```