ME102B Final Project: Cooling Fan for Fried Food

By: Ethan Chung

Product Description:

During quarantine, my mother bought an air fryer to experiment with new cooking methods she saw on YouTube. After successfully frying her first plate of chicken, my mom has been frying things nonstop and my family in turn always ends up waiting patiently in the kitchen for my mom's latest fried creation to cool. I saw this excruciating wait time as a cute opportunity to build my own little cooling device using the material provided in the Mechanical Engineering department's microkit and the techniques I learned in class. The system I designed is essentially a fan that will only turn on after pressing a button and once on, the speed can be adjusted in real-time using a potentiometer. The fan blades themselves are made of cardboard so it is light enough for the small brushless DC motor to rotate at a fast enough speed needed to generate the cooling breeze.

Figure 1: The cooling fan made out of cardboard is powered by a 5V power supply provided by a power cable that is plugged under the desk. On the left is the device without the housing and easier access to the breadboard for testing purposes. On the right is the device with the top housing for the circuit.

Electromechanical details:

Building the Fan: As mentioned above, the fan blades are made with cardboard. These cardboard blades are identical in size and as depicted in figure 2 are folded at a concave angle (in order to be able to generate a breeze when they are spun). These blades were then attached and secured to the wooden circular marker attached to the motor shaft hub using brass fasteners, 4-40 screws, and tape.

Building the Standing Platform: The standing platform is needed to elevate the breadboard and BDC motor at a level high enough where the fan does not interfere with the surface the device is sitting on. As depicted in figure 1 on the left, the stand itself is made of cardboard and duct tape while the top of the platform is made of the plywood provided by the ME department microkit. The breadboard is attached to the plywood using its in-built adhesive. The BDC motor is secured to the platform using two 2-56 screws and the motor mounting bracket as shown in figure 2 below.

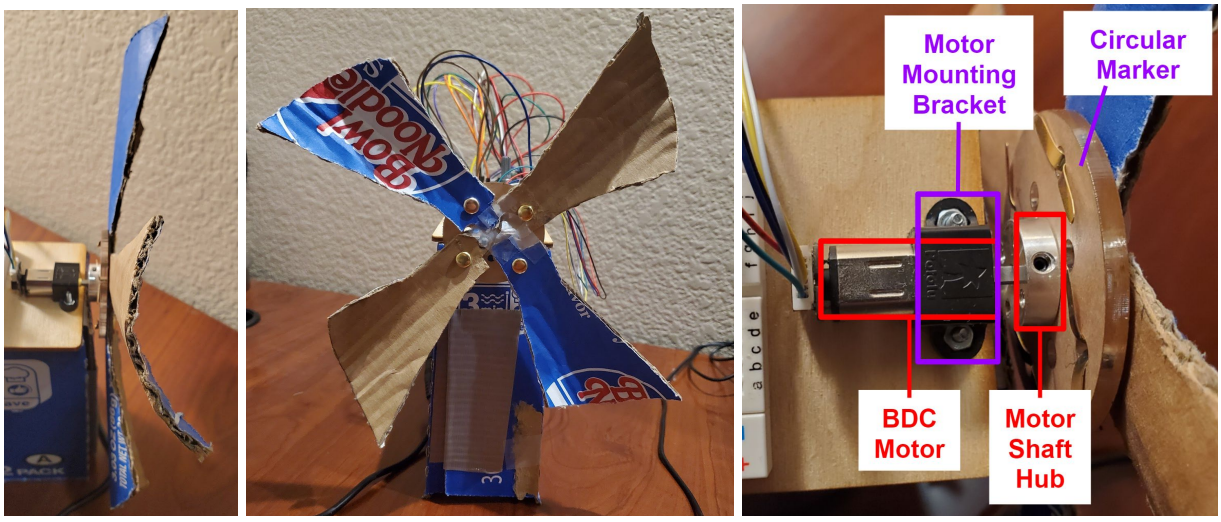

Figure 2: The fan's blades concave inward to more easily generate a larger breeze and make cooling more efficient (left and center). The BDC motor is connected to an encoder/connector and is fastened to the platform using a motor mounting bracket. The motor is then connected to a motor shaft hub and circular marker which in turn attach the fan blades to the rotary motor (right).

Building the Top Housing Box: Once calibration testing was completed, the wires on top of the device needed to be safely housed. Therefore an additional box made of cardboard was placed and secured on top of the plywood platform to contain the wires. Some notable features of the box can be seen in figure 3 and include holes in the front to make space for the fan's motor, holes in the back for the USB cable that powers the microcontroller and the external power supply, and a latch on the top for easy access to turn on the motor and adjust the speed of the fan with the push-button switch and potentiometer respectively.

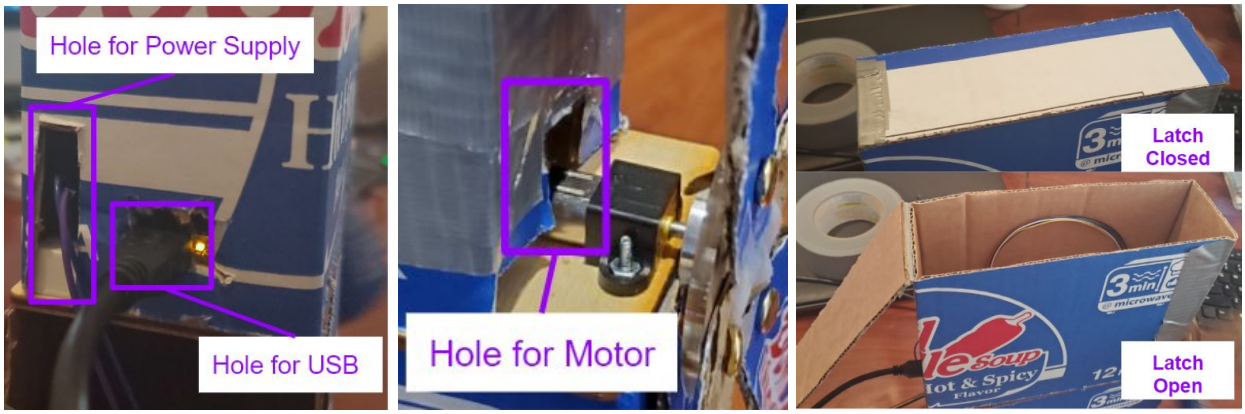

Figure 3: Features of the top housing from the back (left), front (center), and top (right) which are oriented around the breadboard in a way that allows for easy access to the MicroUSB port of the microcontroller, the power supply, BDC motor, and push-button and potentiometer respectively.

Circuit:

This machine's components are centered around its circuit. The four biggest components of this project are the following: the brushless DC motor fan, the potentiometer, the push-button switch, and the microcontroller, all of which could be found on the microkit resource hub (https://microkit.berkeley.edu/).

1. Brushless DC Motor: The motor is a "Micro Metal Gearmotor HP 6V with Extended Motor Shaft" from Pololu and has an encoder attached at the back. The encoder's connector is used to power the motor and also allows for the motor to change to the direction of its rotation and for the microcontroller to record the motor's RPM when used in conjunction with the H-bridge. This motor is powered by an external 5V power supply.
2. Potentiometer: The linear potentiometer (PDB181-E420K-102B) is a passive electronic component that functions as a variable resistor by rotating the knob to adjust the proportion of resistance on each side of the voltage divider. The potentiometer used can be adjusted between a range of 1kΩ to effectively zero resistance and was mapped to a corresponding 8-bit resolution.
3. Pushbutton switch: The other passive electronic component of the circuit, this pushbutton is used to switch between the two states of the device ("On" and "Idle"). By pressing the button and closing the signal, it changes the conditions needed to update a variable in the switch interrupt in my Arduino sketch which dictates whether the potentiometer's resistance will be mapped to the motor or not. The pushbutton requires a pull-up resistor of 10kΩ as it needs to be at around 9/10 of the impedance of the microcontroller
4. Microcontroller: The HUZZAH32 Feather microcontroller is Adafruit's lightweight ESP32-based microcontroller board which is used to run the Arduino sketch I wrote for the project. The potentiometer is connected to one of the analog input pins of the microcontroller, while the other pins (besides ground) are associated with GPIO outputs for the push button and motor connector/encoder. It is powered by the 3V USB port which I connected to my computer.

In addition to these components, I also used the attached encoder's ability to keep track of the motor's RPM in order to properly calibrate it with the mapping of the potentiometer. This is why connections from the encoder to the H-bridge and microcontroller pins are reflected in the circuit diagram in figure 4.

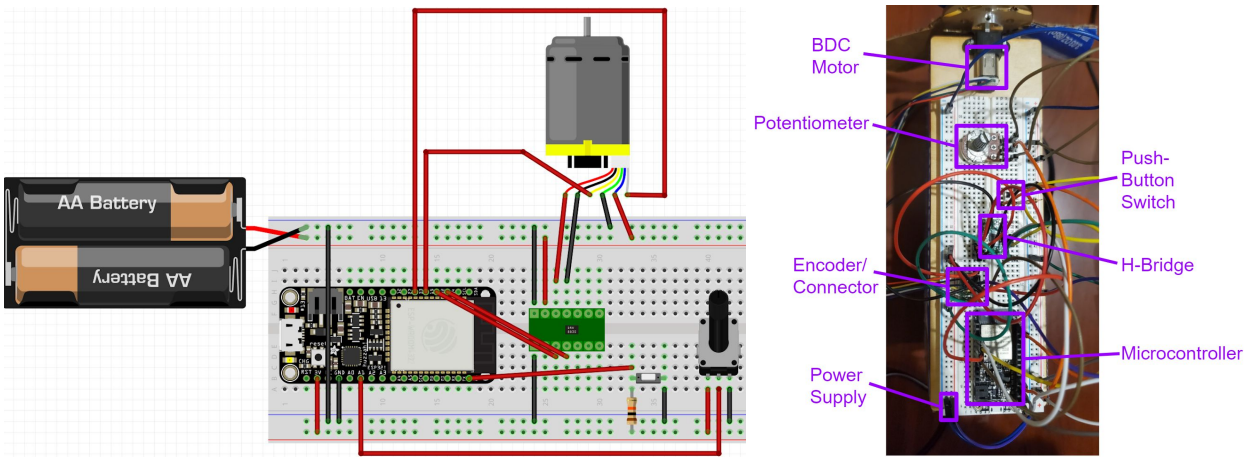

Figure 4: The circuit diagram of the machine alongside a labeled picture of the circuit on the breadboard.

Finite State Machine

The behavior of my machine is fairly straightforward, since the fan is either "ON" or "IDLE," depending on whether the button is pressed. The outcome from the state machine can be observed in the video whereby pushing the button the machine switches from the "idle" state to the "on" state and vice versa. The video and the Finite State Diagram below also demonstrates how the speed can be controlled by turning the potentiometer (counterclockwise for a faster speed, clockwise for slower).

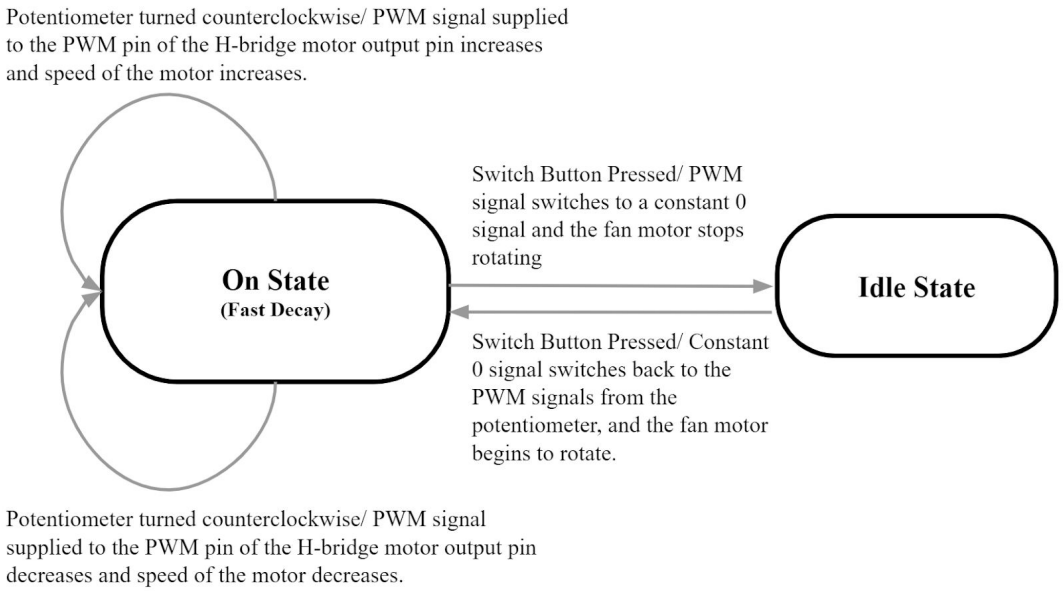

Figure 5: The Finite State Diagram (FSD) for the cooling fan system.

For the complete Arduino IDE code, see Appendix I.

A few things to note regarding the code: I use the millis() function to help implement debounce within the switch interrupt to help prevent the machine from behaving unpredictably if the button is pressed too often in quick succession. Dead space was also accounted for while mapping the potentiometer's resistance values to the fan motor speed and was calibrated with the help of the encoder recording the RPM (the code that calculated the RPM is not included in the following sketch as it is not needed for the device to function properly).

Appendix 1: Arduino Code

```
// Code for Final Project
// Select the output pin for the button
#define Button_pin 21
// Select the input pin for the potentiometer
#define Analog_pin A1
// Select the motor driver pins
#define MD_pin1 14
// Setting a variable to record the analog value from the potentiometer.
int sensorValue = 0;
int motor_speed = 0;
// Volatile Booleans that controls the interrupt
volatile bool interruptstate = false;
// Saves the latest passage of time
unsigned long most_recent = 0;
// Variable to set up for millis() function
unsigned long time_passing;

void IRAM_ATTR switchinput() {
  if (time_passing - most_recent > 150){
  interruptstate = !interruptstate;}
  {
  most_recent = time_passing;
  }
}

void setup(){
  Serial.begin(9600);
  // Sets the switch button press as an input
  pinMode(Button_pin,INPUT);
  // Incorporating debounce with button presses
  attachInterrupt(digitalPinToInterrupt(Button_pin), switchinput, RISING);
  // Sets up the Motor_Driver channel 1, 6 kHz frequency, and 8-Bit resolution
  ledcSetup(1, 6000, 8);
  // Assigns the LED pin to GPIO Channel 1
  ledcAttachPin(MD_pin1, 1);
}
```

```
void loop() {
  // Setting up the millis function
  time_passing = millis();
  // read the value from the sensor:
  sensorValue = analogRead(Analog_pin);
  // Mapping the potentiometer analog reading to output motor speed based on the 8-bit resolution
  motor_speed = map(sensorValue, 0, 4095, 50, 255); //Dead Space accounted for
  ledcWrite(1, motor_speed);
  // Turning off the motor on whether the button is being pressed
  if(interruptstate == true){
    ledcWrite(1, motor_speed);
    ledcWrite(2, 0);
  }
  else{
    ledcWrite(2, 0);
    ledcWrite(1, 0);
  }
}
```