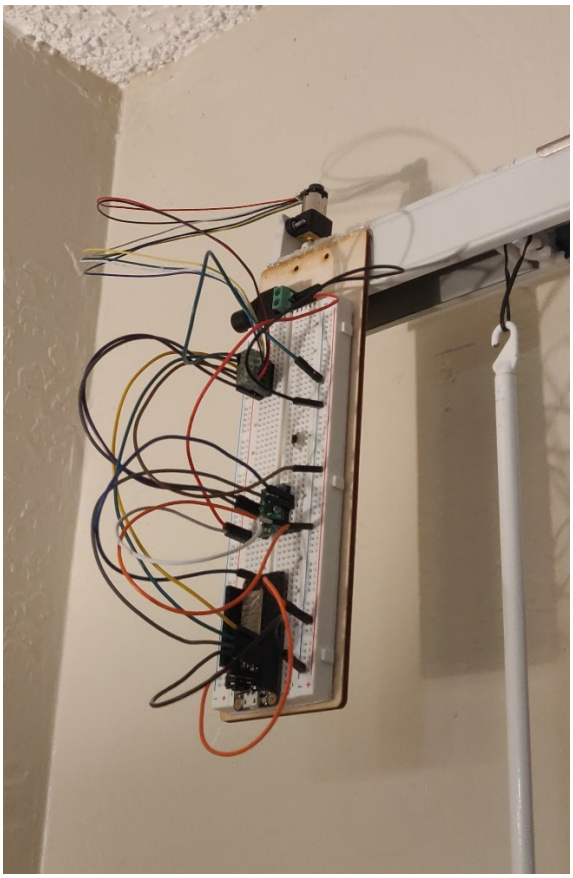
ME102B – Fall 2020 – Final Project

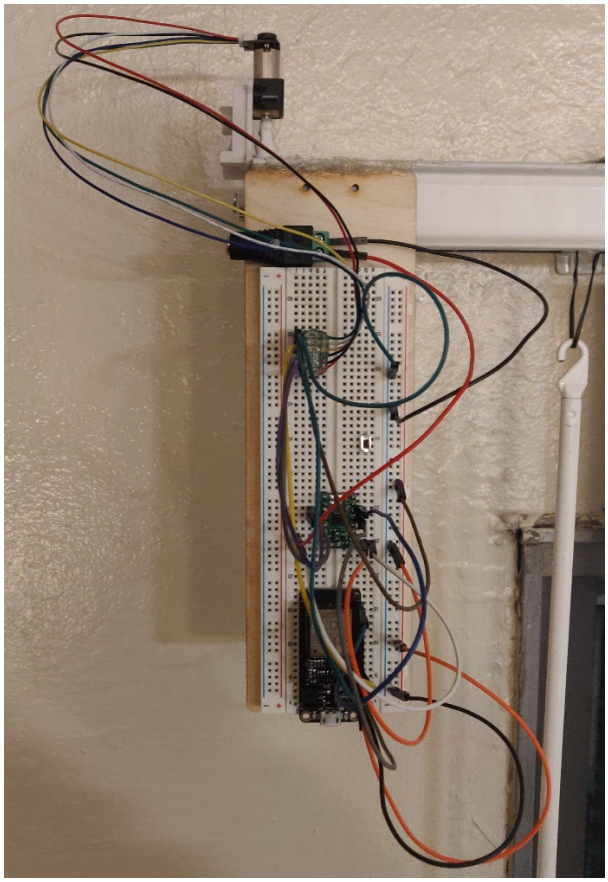
Bluetooth Blinds

By Dimitri Stetsenko

Project Description:

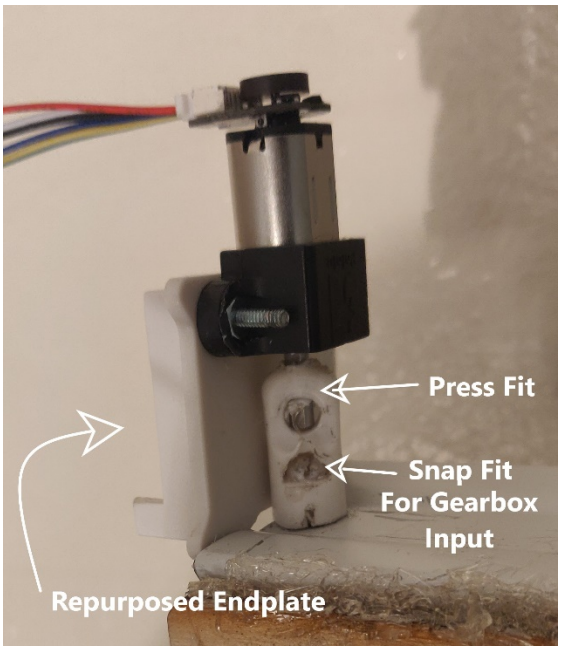
The objective here is to automate something at home that both my roommates and I can get use of and have fun with. In my apartment, we have a large balcony window which is great for letting light in during the day but also makes us feel a little exposed at night, so we find ourselves opening and closing the blinds at least twice a day. I saw this as a great opportunity to implement the motor control strategies covered this semester, while also exploring other wireless features in the ESP32. In the end I opted to use the Bluetooth module so that we could all control the blinds from our phones relatively easily, however the simple circuit and code could be easily modified to work on Wi-Fi or even with a capacitive touch sensor somewhere else in the room. This obviously isn't a permanent solution however with the addition of some simple 3D printed housings and a smaller more dedicated motor controller and Bluetooth module the system could clean up nicely and feature a very small footprint on the actual blinds.

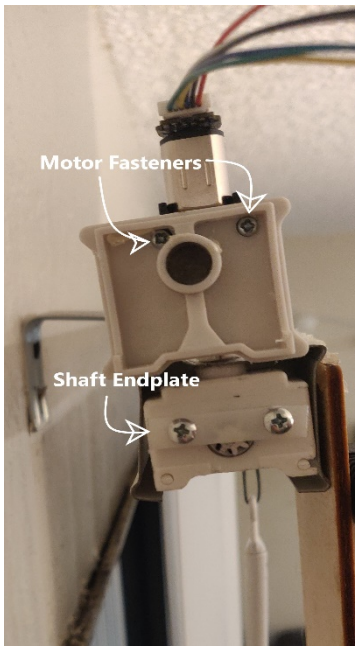




Electromechanical Details:

In order to interface the motor with the blinds, the existing worm and wheel gearbox that previously served as the interfaced between the user and the blinds was used to drive the rotation of the blind panels. To accomplish this the motor's output shaft was press fitted into the input of this gearbox. This assembly was then attached rigidly through a hole in the frame rail with the motor mechanically fastened to a repurposed base plate. For the time being, the circuitry was hot glued in place on the side of the frame rail.

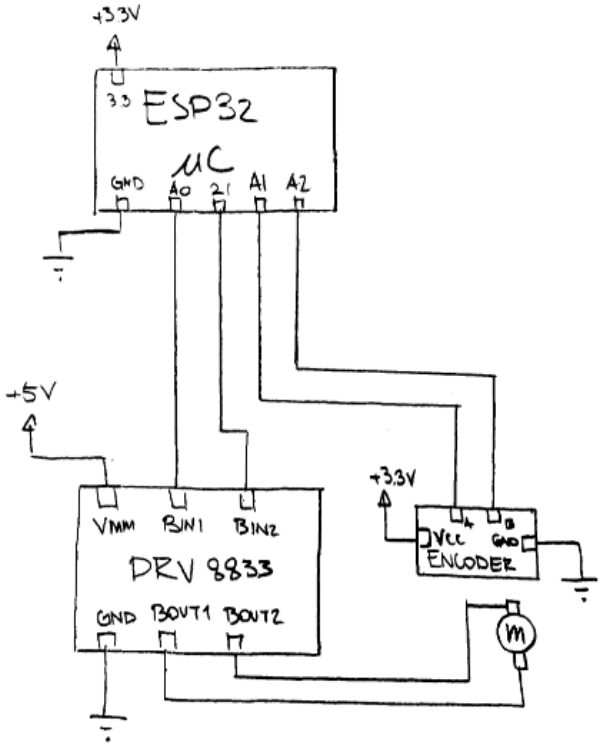




From the images, we can see an existing snap fit repurposed for double duty: holding the motor to the gearbox and keeping the gearbox from falling out of the hole in the frame rail. Additionally, since the end plate was repurposed as a motor mount, a new piece of plastic was used to prevent the splined shaft from working its way out of the side of the blinds.

Circuit:

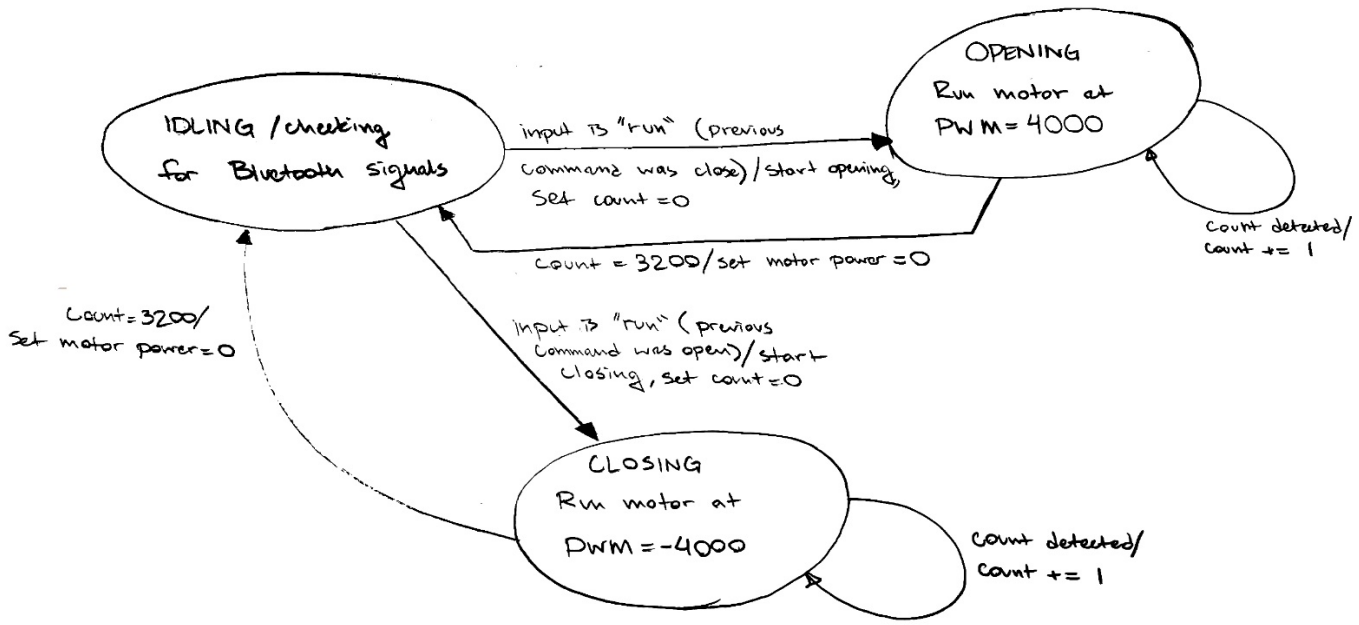
The circuit used is relatively simple and relies only on the DRV8833 motor driver and ESP32 microcontroller:



Aside from this final design, the circuitry went through several iterations that tested different mechanical and digital features. At first a MOSFET was used to run the motor and test if the mechanical side of the project could run and then a switch was used to rest the direction of rotation between ECU flashes. This was done to prevent damage to the motor as it would stall and short out if it tried to close already closed blinds or vice versa.

Finite State Machine:

Below is a Finite State Machine that commands the logic circuit.



The logic only had three states: Idling which here means simply checking for any new Bluetooth communications, opening, where the motor spins in the opening direction until the counter reaches the stopping value of 3200 counts, and closing, which is the inverse of opening. A key feature that was incorporated into the code, and is shown here as the guard conditions, is that while the machine is idling the next state can not be the same as the previous state. This prevents damage to the components if the machine tries to open an already opened blind, for example.

See the full code listing below. Note that in order to more elegantly implement this guard condition the command variable, which dictates whether the blinds will open or close, is multiplied by negative one every time a Bluetooth run command is received. This is then translated through the switch case structure and the microcontroller either opens or closes.

References:

randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/

Arduino Code:

```
// Load libraries
#include "BluetoothSerial.h"

// Check if Bluetooth configs are enabled
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

// Bluetooth Serial object
BluetoothSerial SerialBT;

// Handle received and sent messages
String message = "";
char incomingChar;

int state = 0;
int command = 1;
volatile long count;  //counter for encoder

void setup() {
  Serial.begin(115200);
  // Bluetooth device name
  SerialBT.begin("Automatic Blinds");
  Serial.println("The device started, now you can pair it with bluetooth!");

  ledcAttachPin(A0, 1); // GPIO pin A0, PWM channel 1
  ledcSetup(1, 5000, 12); //PWM channel 1, 5kHz, 12-bits
  ledcAttachPin(21, 2); // GPIO pin 21, PWM channel 2
  ledcSetup(2, 5000, 12); //PWM channel 2, 5kHz, 12-bits

  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  attachInterrupt(digitalPinToInterrupt(A1), encode, CHANGE); // sets interrut to react to changes
  attachInterrupt(digitalPinToInterrupt(A2), encode, CHANGE); // sets interrut to react to changes

}

void loop() {

  // Read received messages (LED control command)
  if (SerialBT.available()){
    char incomingChar = SerialBT.read();
    if (incomingChar != '\n'){
      message += String(incomingChar);
    }
    else{
      message = "";
    }
    Serial.write(incomingChar);
  }

  // Check received message and control output accordingly
  if (message =="run"){
    state = 1;
    command = -1*command;
  }


 switch (state){ //state derived from command over BT
    case 1: // forward bin1 = PWM, bin2 = low
//        Serial.print("State: ");
//        Serial.print(state);
//        Serial.print(" & CMD: ");
//        Serial.print(command);
```

```
//

        switch (command){
//            Serial.print("Entering Command: ");
//            Serial.println(command);
            count = 0;
            case 1: // close it
                  SerialBT.println("Closing");
                  while (count < 3200) {
                  ledcWrite(1, 4000); //BIN 1 - PWM --increase speed forward
                  ledcWrite(2, 0); // BIN 2 - LOW
                  state = 0;
                  }
                SerialBT.println("Done Closing");
                break;


            case -1: //open it
                  SerialBT.println("Opening");
                  while (count < 3200) {
                  ledcWrite(1, 0); //BIN 1 - LOW
                  ledcWrite(2, 4000); //BIN 2 - PWM
                  state = 0;
                  }
                SerialBT.println("Done Opening");
                break;
        }
       break;

    case 0: //stop
      //Serial.println("state 0");
        ledcWrite(1, 0); //BIN 1 - LOW
        ledcWrite(2, 0); // BIN 2 - LOW
        count = 0;
      break;
  }
}


///////count ISR/////////
void encode(){
    count = count + 1;    //increase count
}
//////////////////////////
```