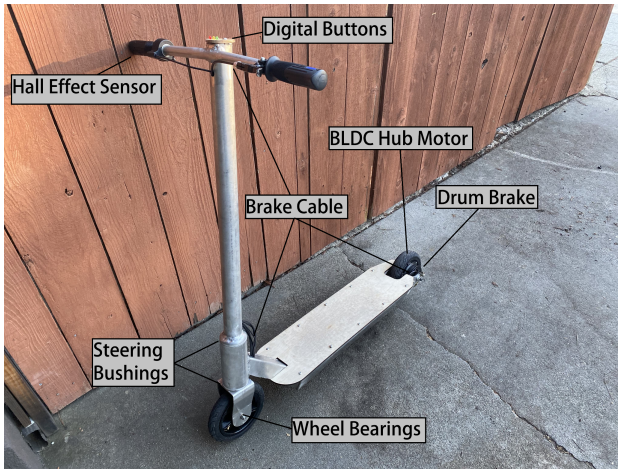
Mark Kallos, Ricardo Alfaro, Jake Glick

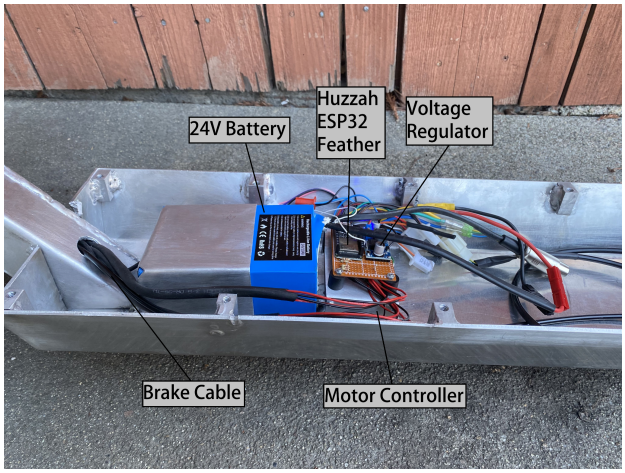
Bear Boost Manual

The Bear Boost scooter was designed to provide a simple, safe, and convenient urban transportation option for people like students.

The scooter was designed to be simple and intuitive so that anyone could ride it. The user ensures that the battery is connected by checking to see if any of the LED lights on the handlebars are lit. If not, the user can connect it under the wooden scooter deck. With the green LED light lit, the scooter is in “standby” mode where throttle input will not activate the motor. The user can press the left button to put the scooter in a drive mode. The yellow LED represents “comfort” mode, while the red LED represents “performance” mode. The right button toggles between these two drive modes. Comfort mode provides a gentle throttle response while performance mode gives a direct throttle response. The user can activate their desired drive mode then stand on the scooter and twist the right throttle handle. The left hand brake lever can be used to slow down. While not being used or while charging, the user can disconnect the battery under the deck.

Before designing and building the scooter, a few success metrics were established. The first being safe and repeatable functionality which was certainly achieved. We wanted the scooter to weigh under 40 pounds and the final design weighs 26 pounds. The range that we desired from the scooter was 5 miles at the least, and the final scooter has a range of 6.7 miles. We also wanted the scooter to be able to reach a top speed of at least 12 miles per hour and the final design reaches around 14 miles per hour.





$$m_r = 115kg, m_s = 14kg, P_m = 350W, \theta_{max} = 10deg. \quad (1)$$

$$\frac{P_m}{m_t * g * \sin\theta_{max}} = v_{min} = \frac{350}{129 * 9.81 * \sin 10} = 1.6m/s. \quad (2)$$

$$P_m = 350W, V_m = 24V, C_b = 10A * hr, v_{max} = 14mph. \quad (3)$$

$$I_m = \frac{P_m}{V_m} = 14.6A. t_{max} = \frac{C_b}{I_m} = 0.68hours. Range = v_{max} * t_{max} = 9.6miles. \quad (4)$$

$$m_r = 115kg, m_s = 14kg, FOS = 1.5, F_{push} = 100N, h = 83cm, d = 12cm. \quad (5)$$

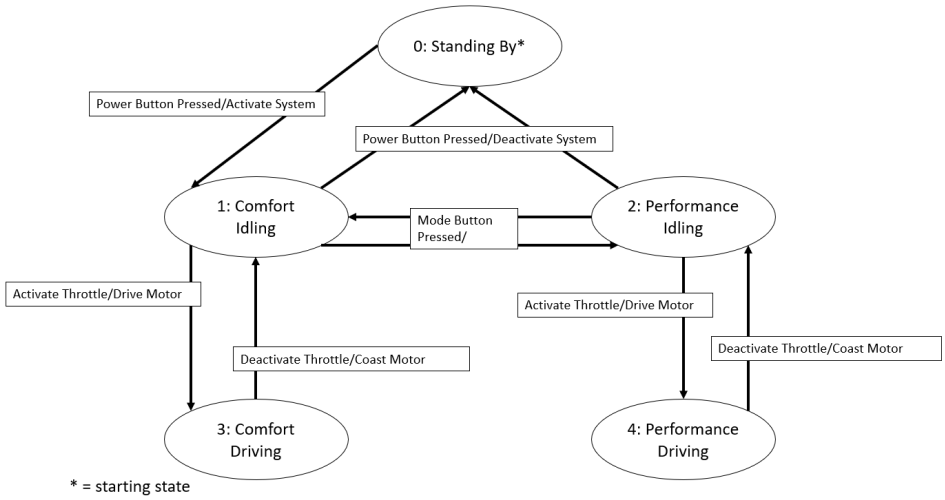
$$F_{b,max} = N_b * FOS = g * (m_r + \frac{m_s}{2}) * FOS = 1.8kN. \quad (6)$$

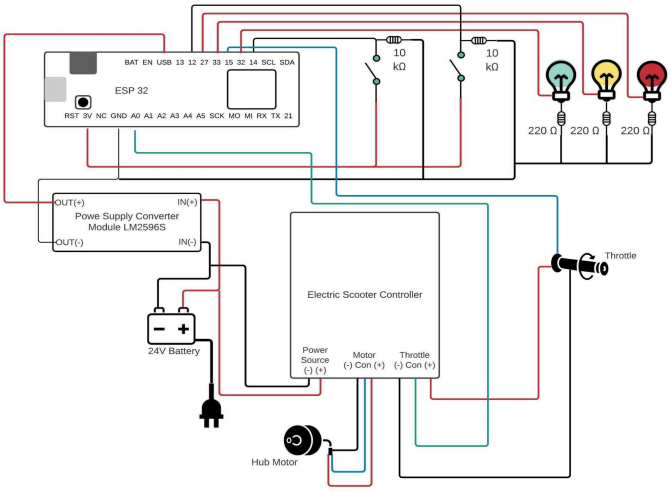
$$F_{b,axial} = (F_r + \frac{F_s}{2}) * FOS = 1.8kN. \quad (7)$$

$$F_{push} + R_1 + R_2 = 0. F_{push} * h - R_2 * d = 0. R_2 = 700N, R_1 = -800N. \quad (8)$$

Using equations 1 and 2 from above, we decided a 350 watt motor would be sufficient to maintain velocity at the max slope we decided of 10 degrees. Equations 3 and 4 were used to determine which size battery to get. A 10 amp-hour battery would theoretically provide enough charge to have a 9.6 mile range which, accounting for losses, would be sufficient for our uses. Equations 5 and 6 were used to determine the max force on the front and rear wheel bearings (in the worst case scenario of the rider standing directly over them) with a factor of safety of 1.5. Equations 7 and 8 were used to choose the bearing mechanism for the steering. With a max axial force of 1.8 kN and a max radial force of 800N, we decided a much cheaper and easier to manufacture option would be to use bushings rather than bearings.

We decided to incorporate a brushless direct current (BLDC) hub motor to minimize drivetrain losses as well as have longer term reliability and quieter operation. However, since BLDC motors don't have the mechanical commutation that brushed motors do, the commutation must be handled by electronics or software. We elected to handle this electronically using the motor controller that was designed for and sold with our motor. Our motor uses three hall effect sensors to determine the position of the rotor and communicates this to the motor controller. The motor controller uses these signals to cycle power to each of the coils which keeps the motor spinning in the same direction. We elected to use this motor controller and simply command it with our ESP32 as the controller was already designed to function with our motor.





After completing this project for ME 102B, our group believes we did some things well and that there were some other things we could improve upon in the future. Splitting up the workload so that each person worked more specifically on the parts of the project they were strong in really helped manage the workload and improve our efficiency. The due dates for the deliverables helped us spread out the work over the semester. We also did a lot of the manufacturing together which allowed us all to collaborate on any design decisions and brainstorm solutions to problems we came across. We could improve upon our planning and communication outside of work hours. Our greatest shortcoming was that we left very little time for manufacturing. A lot of small problems and design decisions that we didn't anticipate needed to be fleshed out toward the tail end of manufacturing which left us very little time to finish the project. We should've anticipated complications and began working on manufacturing much earlier to leave extra time for us to solve these problems and ensure our scooter worked as intended.

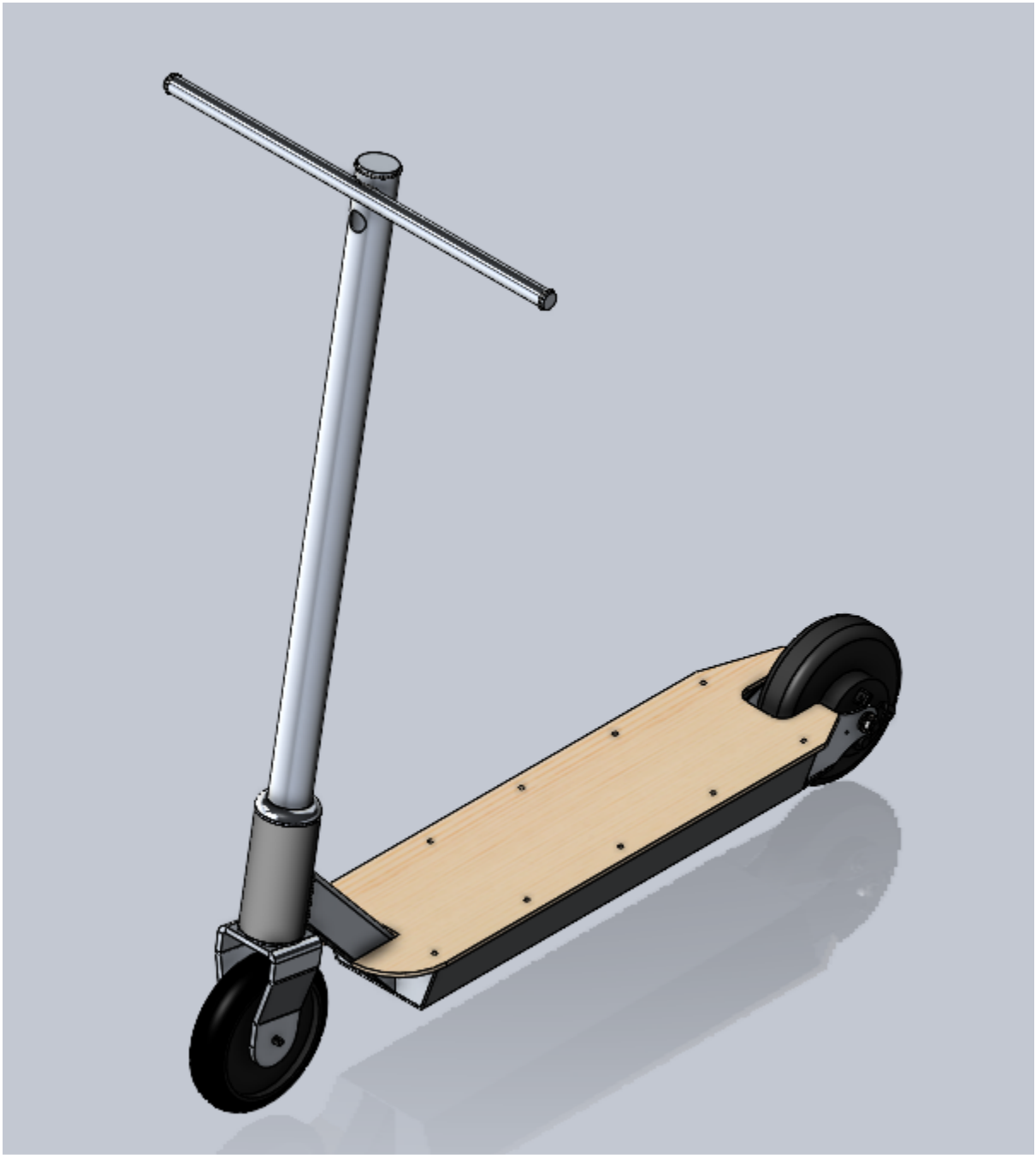
Appendices
A1. BOM

Item	Item Name	Description	Purchase Justification	Serial Number / SKU	Price (ea.)	Quantity
1	Brushless Hub Motor Kit	350W Motor, rear wheel, motor controller, throttle, etc.	Main power transmission components and required driver	B092DD31L9	$ 132.99	1
2	1/8" Aluminum 24" x 24" 5052 Sheet Plate with Vinyl PVC Coating one side	24" x 24" 5052 1/8" .125" aluminum sheet stock	Chassis	162636639358	$ 45.60	1
3	Multipurpose 6061 Aluminum Round Tube	0.065" Wall Thickness, 2" OD, 3'Length	Steering Rod	9056K83	$ 48.07	1
4	Multipurpose 6061 Aluminum Round Tube	1/8" Wall Thickness, 3" OD, 1' Length	Steering Stem	9056K41	$ 29.69	1
5	Zinc Yellow-Chromate Plated Hex Head Screw	Grade 8 Steel, 1/2"-13 Thread Size, 7" Long, Partially Threaded, Packs of 1	Fasteners	91257A738	$ 3.73	1
6	316 Stainless Steel Washer	1/4" Screw Size, 0.281" ID, 0.625" OD	Fasteners	90107A029	$ 7.11	1
7	High Strength Steel Nylon-Insert Locknut	Grade 8, 1/4" - 20 Thread Size	Fasteners	90630A110	$ 4.49	1
8	Button Head Torx Screws	Zinc-Plated Steel, 1/4"-20 Thread Size, 1" Length, Packs of 25	Fasteners	90910A531	$ 7.54	1
9	Steel Belleville Spring Lock Washer, Zinc-Plated	for M8 Screw Size, 8.400mm ID, 18.000mm OD, Packs of 10	Fasteners	90127A128	$ 4.96	1
10	Multipurpose 6061 Aluminum Round Tube	0.035" Wall Thickness, 7/8" OD, 2 Feet Long	Handlebar	9056K72	$ 18.68	1
11	Multipurpose	1/2" Thick x 1" Wide, 1/2	Inserts	8975K11	$ 4.34	1

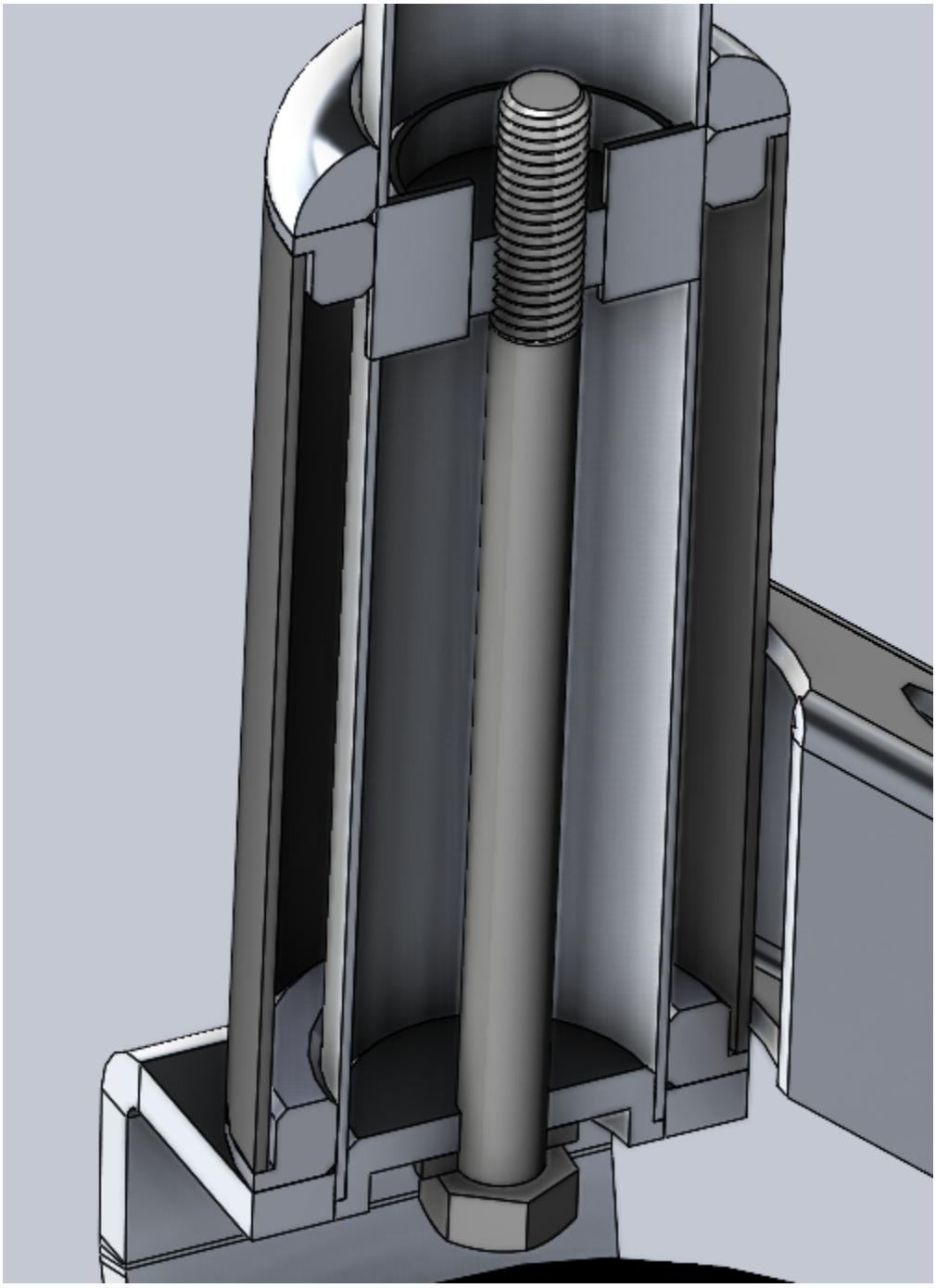
	6061 Aluminum	Feet Long				
12	Padyrytu 8 Inch Electric Scooter Tire 8X1 1/4 Inner Tire 200x45 Pneumatic Tire Whole Wheel-8MM	Front Wheel for scooter	Front Wheel/Bearings	B098XPJFRQ	$ 25.99	1
13	MDS-Filled Easy-to-Machine Cast Nylon Tube	Wear-Resistant, 1/2" Wall Thickness, 3" OD, 2" ID, 1 Foot Long	Bushings	4363N126	$ 45.97	1
14	21700 Lithium Ion Battery	24V 10Ah 20Ah 250W 350W Lithium Ion Pack Ebike Battery for Scooter EBike Motor	Power Source	224646931907	$ 149.99	1
15	Stainless Steel Rod	5/16" x 12"	Front Axle Stock	5203872	$ 4.99	1
16	Brake Cable+Housing	Cable and housing for the brake system	Brake Cable	CAB41903039K	$ 26.45	1
17	Plywood 1/4x24x48"	Plywood sheet	Scooter Deck	N/A	$ 13.32	1
18	10k Ohm Resistor	Resistor	Electronic Component	RSF200JB-73-10K	$ 0.44	2
19	220 Ohm Resistor	Resistor	Electronic Component	F1W122	$ 1.45	1
20	Huzzah32 Feather ESP32	Microcontroller	Microcontroller	3405	$ 19.95	1
21	Green LED	Dashboard light	Electronic Component	WP7113GT	$ 0.39	1
22	Yellow LED	Dashboard light	Electronic Component	WP7113YD5V	$ 0.64	1
23	Red LED	Dashboard light	Electronic Component	WP1503ID	$ 0.44	1
24	Digital Button	Digital Input	Sensor/Inputs	SW-PB1-1BS-A-PK1-A	$ 0.38	2
25	LM2596	Voltage Regulator	Electronic Component	200102-10	$ 14.95	1
26	Brake Handle	Handle	Brake System	HB331	$ 12.49	1

A2. CAD

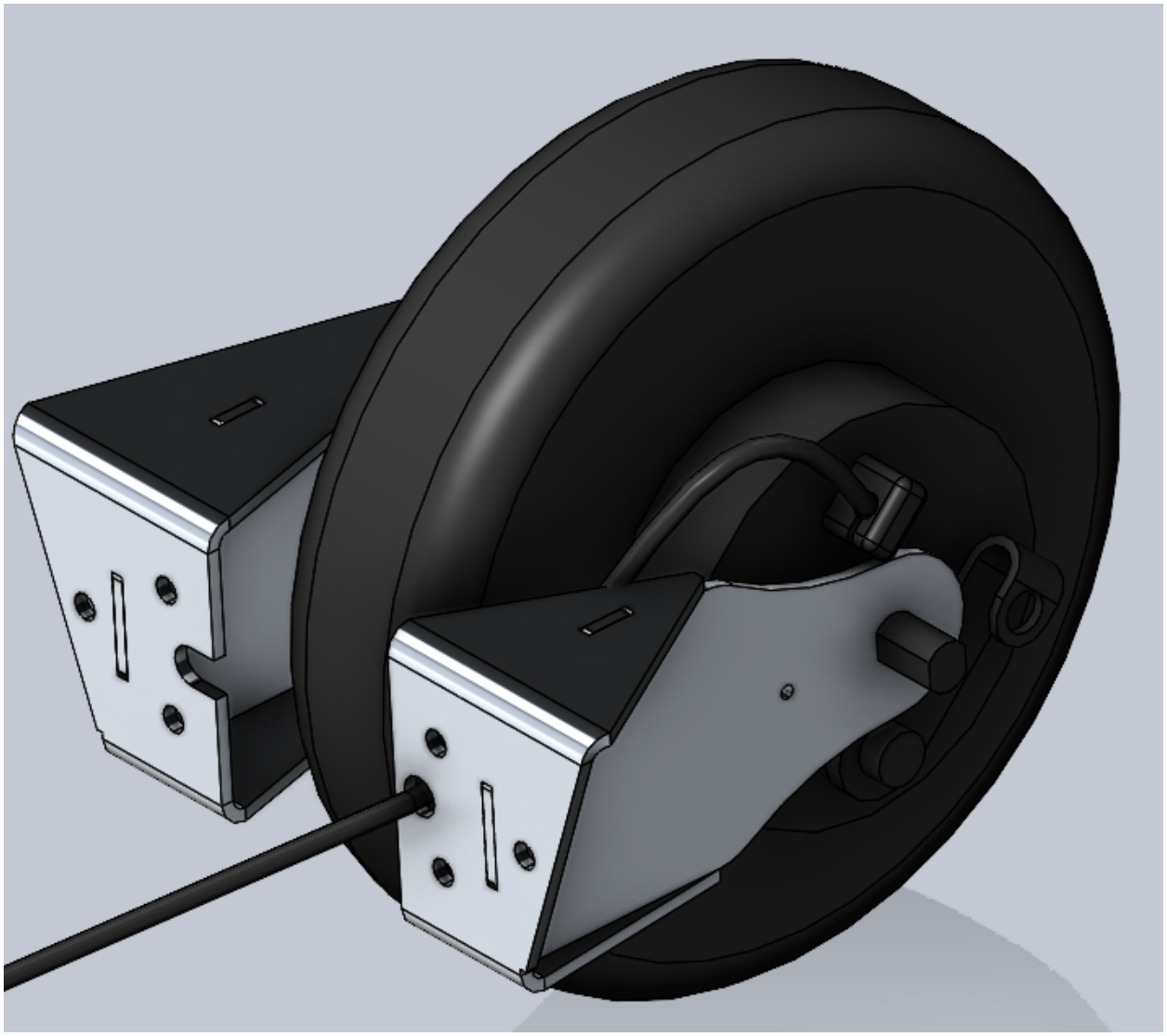
Isometric View



Steering Rod View



Hub Motor/Drum Brake



A3. Code

```
#define motorPin 26
#define throttlePin 15
#define buttonPin 12
#define powerPin 14
#define greenPin 32
#define yellowPin 33
#define redPin 27

const int freq = 5000;
const int motorChannel = 1;
const int resolution = 8;

int state = 0;

const int throttleThresh = 90;
const int increment = 1;
const int minThrottle = 90;
volatile int actThrottle = minThrottle;
volatile int desThrottle = 0;

hw_timer_t * timer0 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

volatile bool buttonPressed = false;
volatile bool powerPressed = false;
volatile bool buttonRecently = false;
volatile bool updateMotor = false;

void IRAM_ATTR isr() {
  buttonPressed = true;
}

void IRAM_ATTR isr1() {
  powerPressed = true;
}

void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  updateMotor = true;
  portEXIT_CRITICAL_ISR(&timerMux0);
}
```

```
void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  buttonRecently = false;
  portEXIT_CRITICAL_ISR(&timerMux1);
  timerAlarmDisable(timer1);
}

void setup() {
  pinMode(throttlePin, INPUT);
  pinMode(buttonPin, INPUT);
  pinMode(powerPin, INPUT);

  attachInterrupt(buttonPin, isr, RISING);
  attachInterrupt(powerPin, isr1, RISING);

  ledcSetup(motorChannel, freq, resolution);
  ledcAttachPin(motorPin, motorChannel);

  Serial.begin(115200);

  timer0 = timerBegin(0, 80, true); // Motor Update timer
  timerAttachInterrupt(timer0, &onTime0, true);
  timerAlarmWrite(timer0, 12000, true);
  timerAlarmEnable(timer0);

  timer1 = timerBegin(1, 80, true); // Debounce timer
  timerAttachInterrupt(timer1, &onTime1, true);
  timerAlarmWrite(timer1, 1000000, true);

  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  digitalWrite(greenPin, HIGH);
  digitalWrite(yellowPin, LOW);
  digitalWrite(redPin, LOW);
}

void loop() {
  setLED();

  switch (state) {

    case 0: // OFF state
```

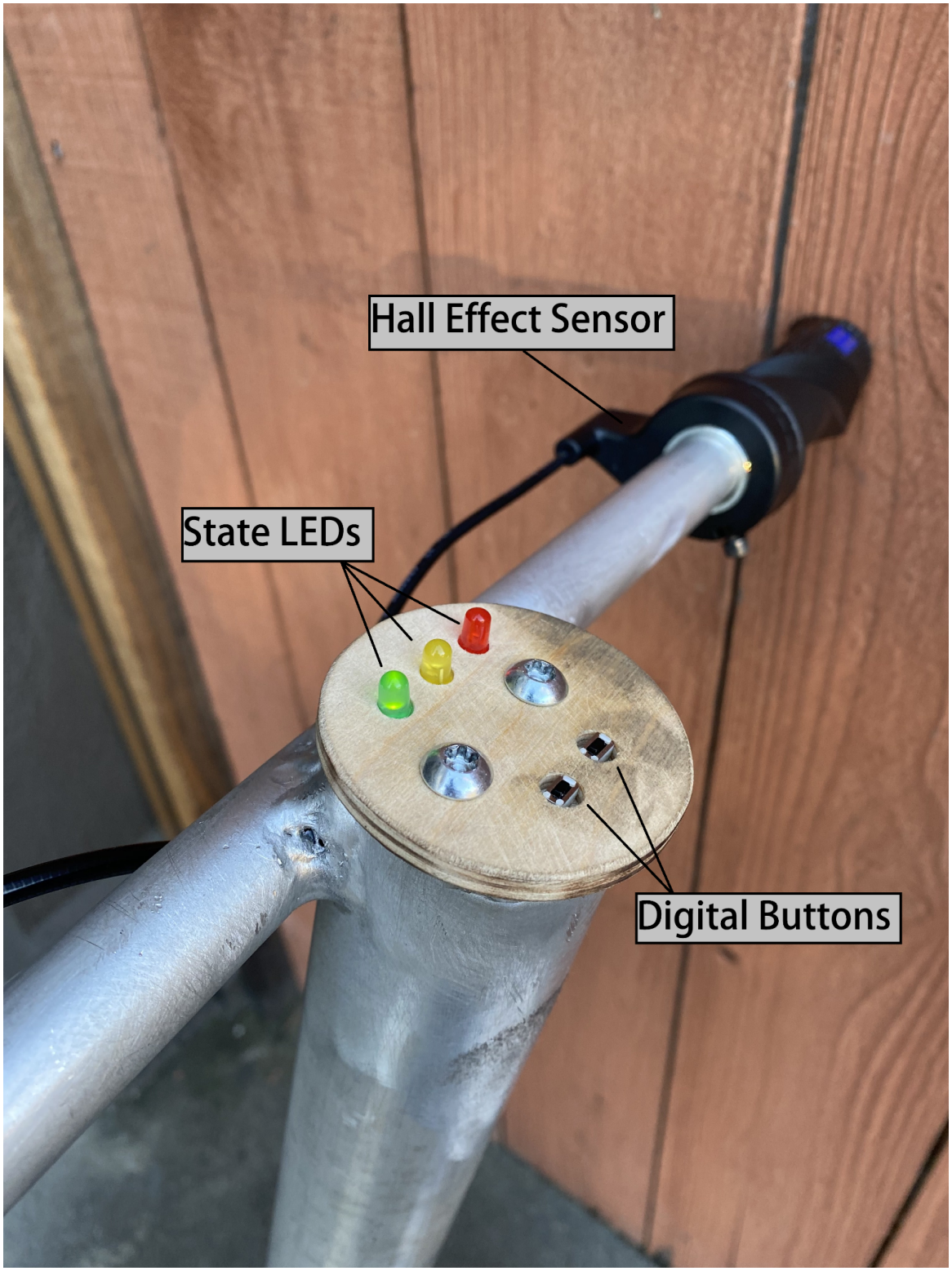
```
  if (powerPressEvent()) {
    activateSystem();
    state = 1;
  }
  break;

case 1: // Comfort Idling state
  desThrottle = map(analogRead(throttlePin), 0, 4095, 0, 255);
  if (buttonPressEvent()) {
    state = 2;
  }
  if (powerPressEvent()) {
    deactivateSystem();
    state = 0;
  }
  if (desThrottle > throttleThresh) {
    startMotor();
    state = 3;
  }
  break;

case 2: // Performance Idling state
  desThrottle = map(analogRead(throttlePin), 0, 4095, 0, 255);
  if (buttonPressEvent()) {
    state = 1;
  }
  if (powerPressEvent()) {
    deactivateSystem();
    state = 0;
  }
  if (desThrottle > throttleThresh) {
    startMotor();
    state = 4;
  }
  break;

case 3: // Comfort Driving state
  desThrottle = map(analogRead(throttlePin), 0, 4095, 0, 255);
  if (desThrottle < throttleThresh) {
    actThrottle = minThrottle;
    coastMotor();
    state = 1;
  }
  if (updateMotor) {
```

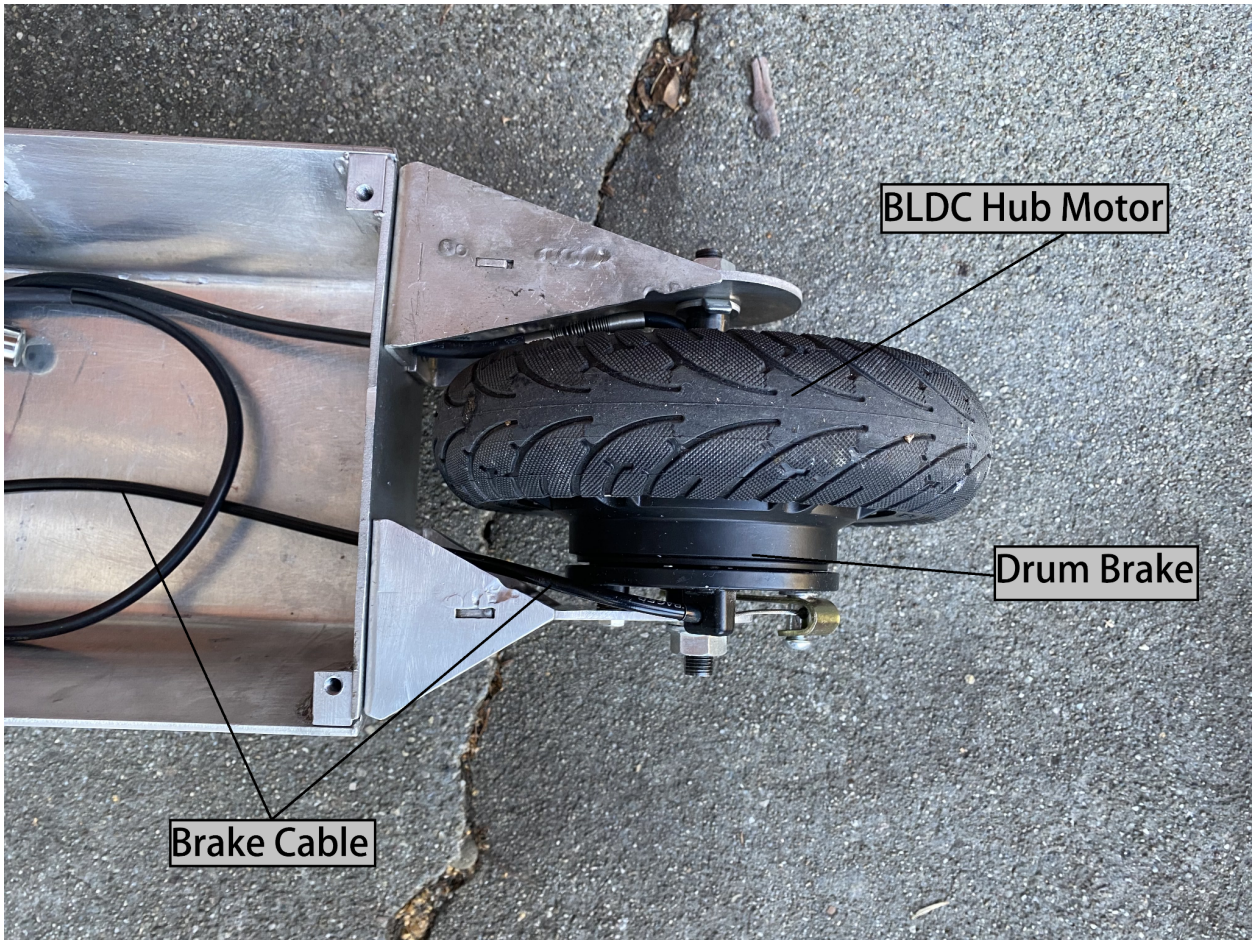
```
        portENTER_CRITICAL(&timerMux0);
        updateMotor = false;
        portEXIT_CRITICAL(&timerMux0);

        comfortMotor();
      }
      break;

    case 4: // Performance Driving state
      desThrottle = map(analogRead(throttlePin), 0, 4095, 0, 255);
      if (desThrottle < throttleThresh) {
        actThrottle = minThrottle;
        coastMotor();
        state = 2;
      }
      if (updateMotor) {
        portENTER_CRITICAL(&timerMux0);
        updateMotor = false;
        portEXIT_CRITICAL(&timerMux0);

        performanceMotor();
      }
      break;

    default: // error state
      Serial.println("STATE ERROR");
      break;
  }
}

bool buttonPressEvent() {
  if (buttonPressed && !buttonRecently) {
    buttonPressed = false;
    buttonRecently = true;
    timerAlarmEnable(timer1);
    return true;
  }
  else {
    buttonPressed = false;
    return false;
  }
}

bool powerPressEvent() {
```

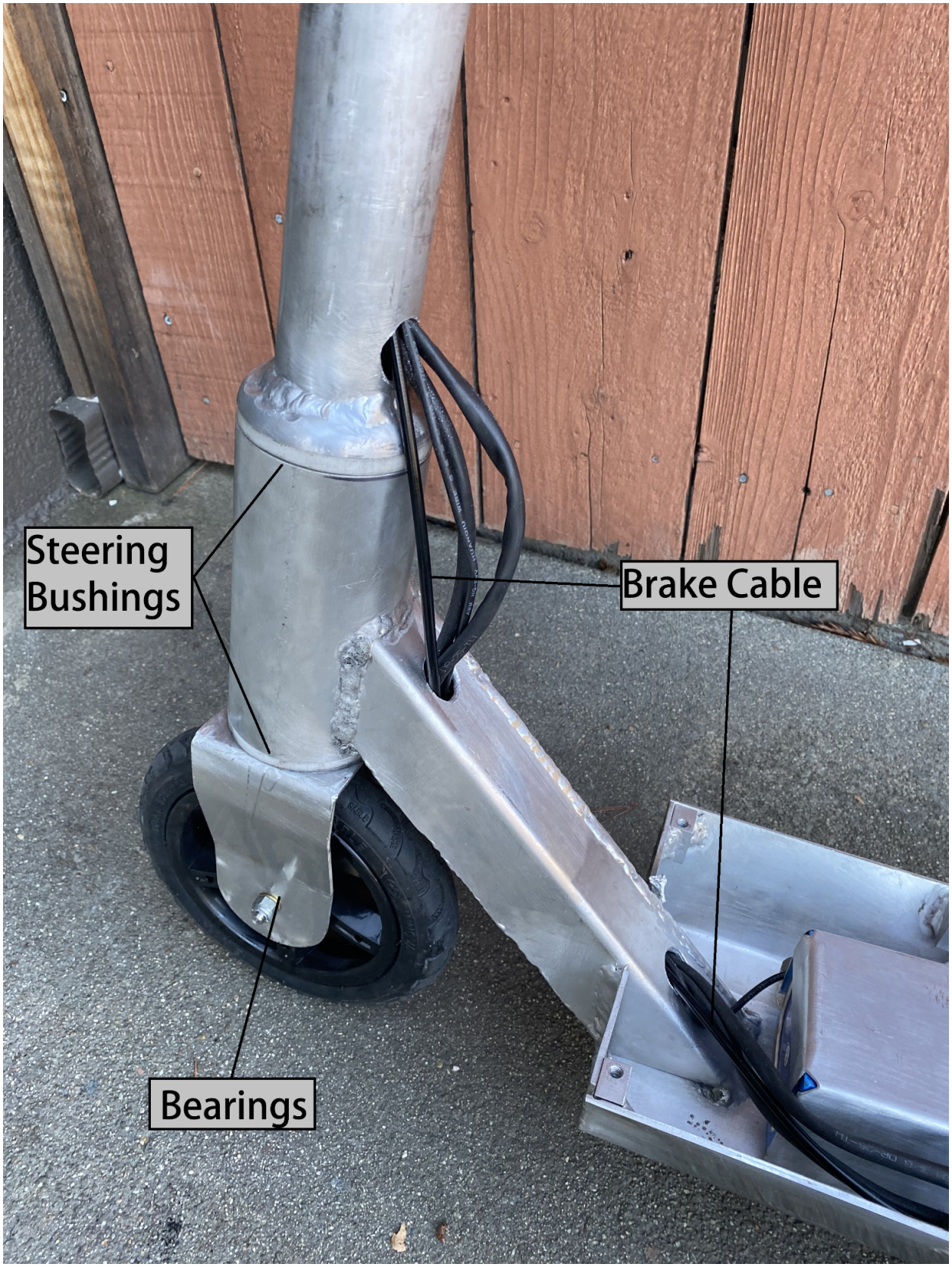
```
  if (powerPressed && !buttonRecently) {
    powerPressed = false;
    buttonRecently = true;
    timerAlarmEnable(timer1);
    return true;
  }
  else {
    powerPressed = false;
    return false;
  }
}

void activateSystem() {
  buttonPressed = false;
  actThrottle = minThrottle;
}

void deactivateSystem() {
  actThrottle = 0;
}

void setLED() {
  if (state == 0) {
    digitalWrite(greenPin, HIGH);
    digitalWrite(yellowPin, LOW);
    digitalWrite(redPin, LOW);
  }
  else if (state == 1 || state == 3) {
    digitalWrite(greenPin, LOW);
    digitalWrite(yellowPin, HIGH);
    digitalWrite(redPin, LOW);
  }
  else if (state == 2 || state == 4) {
    digitalWrite(greenPin, LOW);
    digitalWrite(yellowPin, LOW);
    digitalWrite(redPin, HIGH);
  }
}

void startMotor() {
  actThrottle = minThrottle;
  ledcWrite(motorChannel, actThrottle);
}
```

```
void coastMotor() {
  actThrottle = 0;
  ledcWrite(motorChannel, actThrottle);
}

void comfortMotor() {
  if (actThrottle < desThrottle) {
    actThrottle = actThrottle + increment;
  }
  else {
    actThrottle = desThrottle;
  }
  ledcWrite(motorChannel, actThrottle);
}

void performanceMotor() {
  actThrottle = desThrottle;
  ledcWrite(motorChannel, actThrottle);
}
```

A4. Additional Photos




BLDC Hub Motor
Drum Brake
Brake Cable


Steering Bushings
Brake Cable
Bearings