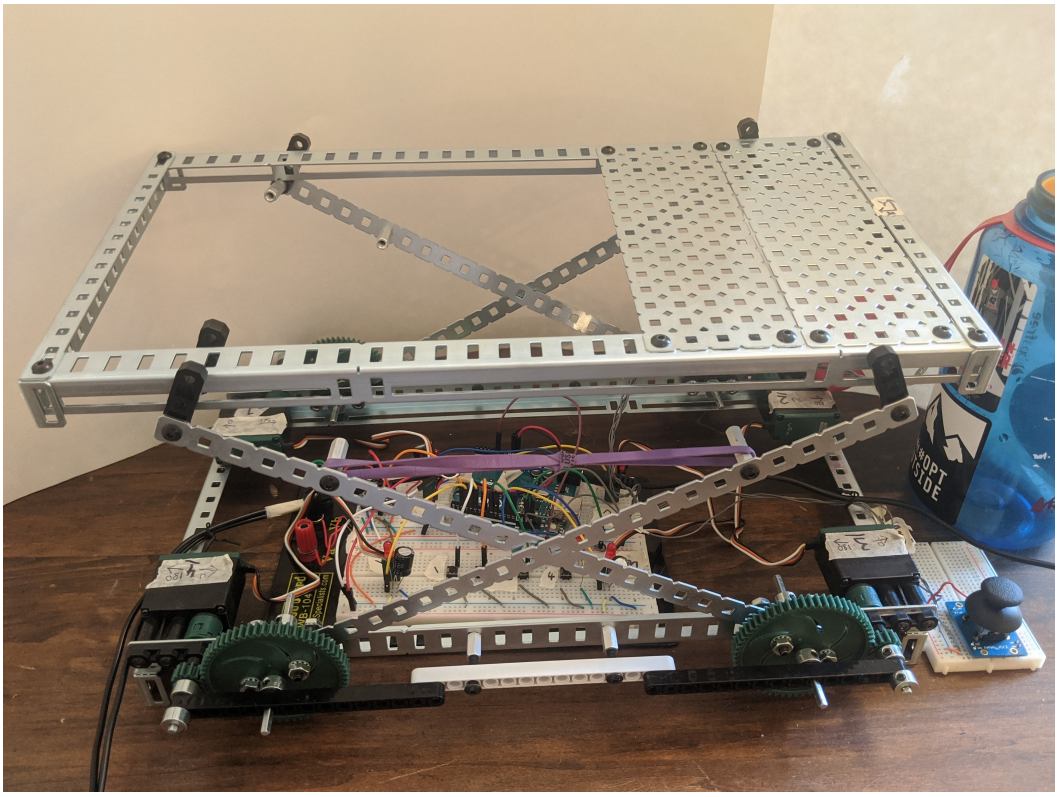


Motorized Standing Desk

BE 121

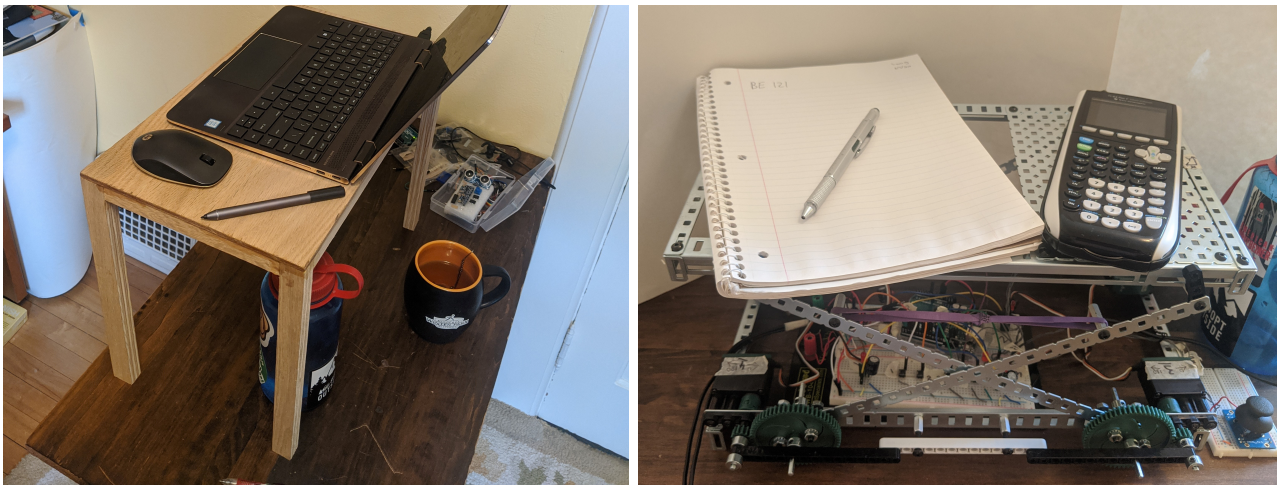
University of California, Los Angeles

Simon Ng

Contents

1 Project Motivation

The health and productivity benefits of a standing desk are well studied [1, 2]. My current standing desk setup is a small wooden platform (standing desk) which sits atop an existing, standard desk (table). I threw my standing desk together in high school in about 30 minutes from a bit of plywood, some scraps of wood, and copious amounts of wood glue. It's functional, but when I want to switch between standing and sitting, I have to remove my things from my standing desk, move it off the table, then put my things back down on the table. It would be much more convenient to have a standing desk able to raise and lower on command, letting me lazily leave everything on top. My solution is a motorized standing desk.



(a) (b)

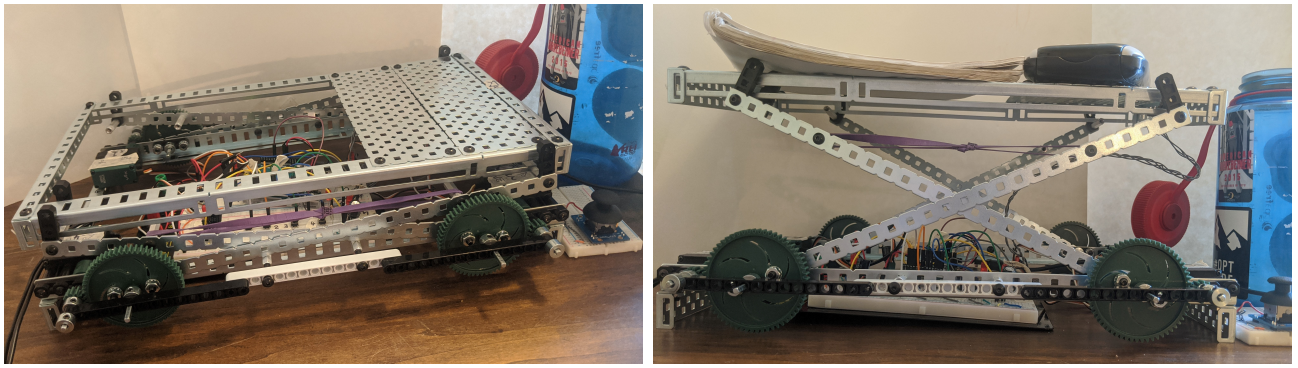
Figure 1: (a) Previous wooden standing desk vs (b) new motorized standing desk

2 Design Overview

My Arduino-based standing desk has 2 pieces, the base and the platform. The base sits on the table with most of the electronics and motors, while the platform is the surface that is raised and lowered. I employed a scissor lift design in which 4 motors attached to the base rotate metal bars (arms) upward. The platform is able to slide on the arms, allowing the contact point between the arms and platform to change as required by the changing arm angle. The height is controlled by a joystick to the side of the desk. Individual motor inconsistencies in lifting the desk are corrected by an accelerometer on the desk platform which is used to automatically adjust the motors until the platform is flat. An optional thermistor can be included as an interrupt. This could be used to detect shorted wires as they grow hot with an LED alert, or could be integrated with a fan on the platform for improved air flow around a hot laptop.

2.1 Solved Mechanical Challenges

Achieving adequate motor torque proved quite difficult. I explored powering the motors with the original VEX Microcontroller to ensure an adequate power supply. This would have required programming the VEX Microcontroller with a $50 proprietary programming module and using the



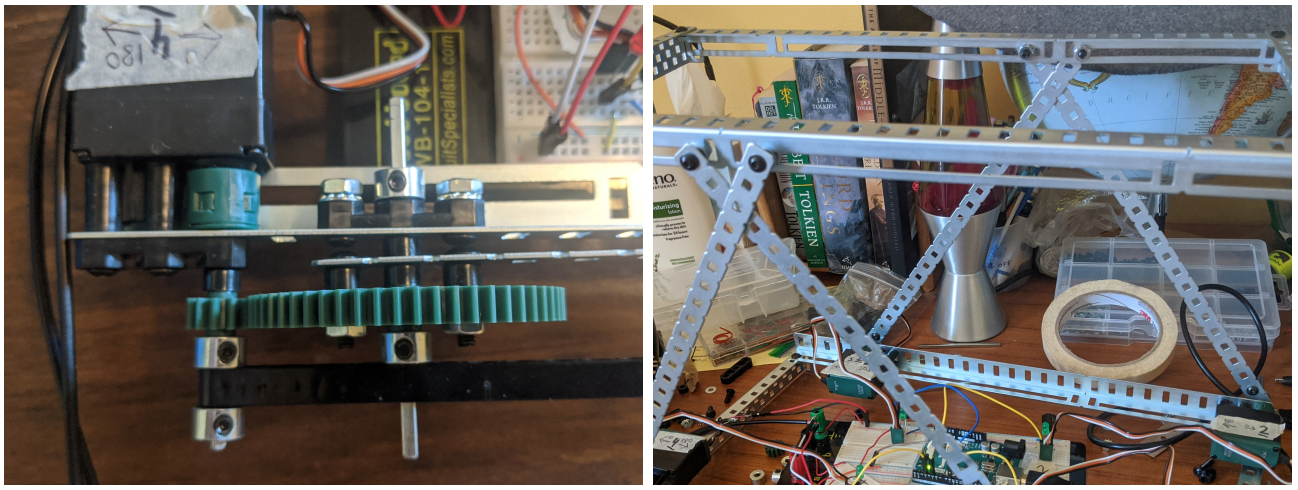
(a) (b)

Figure 2: (a) Desk lowered and (b) desk raised

TX/RX ports to communicate with the Arduino. I abandoned this effort in favor of gearing each arm to provide 5x torque as seen in Figure 3(a). My power challenges were also aided by adding 470 μF capacitors to each motor power supply and providing 2 different 5V, 1A power supplies between the 4 motors.

However, after gearing my motors and arms, the axles for both the small and large gear more often twisted out of place rather than smoothly meshing to lift the load. I added plastic 3-hole bearing flats, which provided a thicker, more stable socket for the axles. I also used Lego pieces to provide another connection between the two axles, preventing them from twisting out of line too much (see Figure 3(a)). I fixed these Lego pieces across both motors on a side and also attached them to the base to prevent the Lego pieces from bending themselves.

I also struggled to achieve a smoothly shifting contact point between each arm and the platform as the arms rotate towards straight upwards. My original design used grooves in the side of the platform by loosely screwing each arm into a groove and allowing it to slide freely (see



(a) (b)

Figure 3: (a) 5x gearing mechanism and (b) unsuccessful asymmetric sliding mechanism attempt

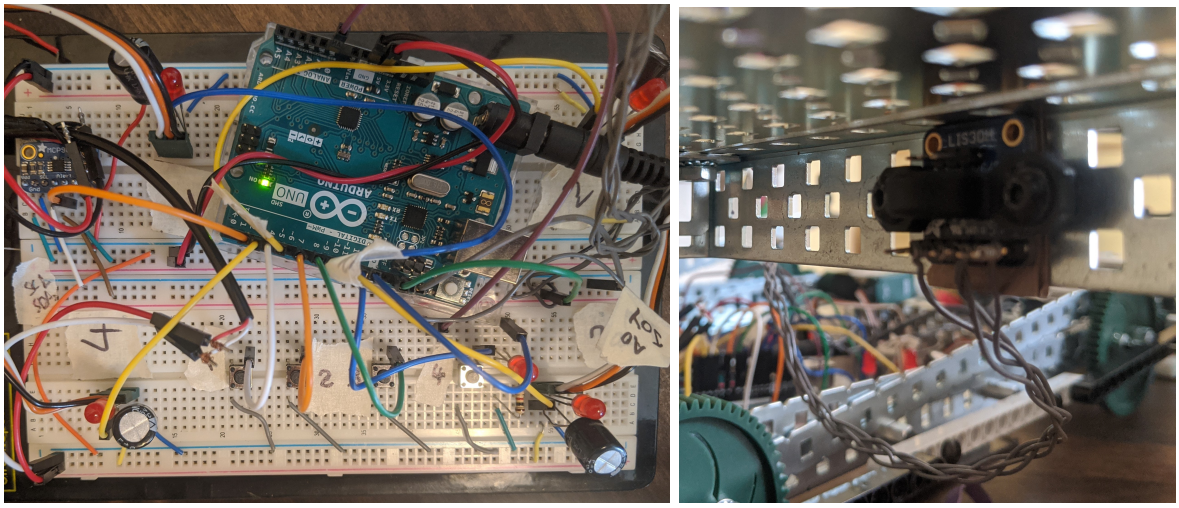
Figure 3(b)). However, the grooves are asymmetric, which made the platform twist and get stuck as it raised. My solution was to use each arm as a rail underneath the entire platform, allowing the platform to slide freely sitting on top of the arms (see Figure 2). I added guide tabs sticking up to help hold the platform in position, but there is still a greater risk for the platform to slide off or otherwise become uneven if the motors do not perform evenly.

When lowering the platform, arms sometimes landed on top of the 60T gear, resulting in a tilted and not fully lowered platform. My solution was to add extra spacers between each metal bar and its attached 60T gear as seen in Figure 3(a), giving ample space for the arms to fully collapse.

3 Electrical

The motors run at 4.4 - 9.1 V and 20 - 1500 mA [3]. I used two separate 5V, 1A power supplies consisting of stripped USB cables attached to wall outlet AC-DC converters rather than powering from the Arduino, as the Arduino's maximum current output is too low at 200 mA [4]. I wrapped the stripped 5V and GND wires around male breadboard wires for an easy breadboard connection.

To mount the accelerometer underneath the platform, I screwed a plastic piece across the back of it and inserted a piece of cardboard between its top face and the metal chassis to ensure no unwanted electrical connections were formed and to cushion the PCB components (see Figure 4(b)). I wrapped stripped wire around each needed pin, braided these together, and guided them to the main breadboard below. A similar approach was used to attach the joystick on a smaller breadboard to the side of the desk. To easily use both the accelerometer and thermistor, I used I2C with the Arduino's SDA and SCL pins and used a spare breadboard bus strip for both sensors' SCL and SDA connections. I connected the thermistor A0 address pin to 5V to give it a unique address from the accelerometer. Once the code is loaded onto the Arduino, there are no computer inputs or necessary printouts, so the Arduino can be powered by the 9V barrel jack without any computer connection.



(a) (b)

Figure 4: (a) Primary breadboard and (b) accelerometer mounted underneath the platform

3.1 Solved Electrical Challenges

Achieving consistent control of the motors proved highly challenging. Among the numerous considerations were the possibility of too high a load, too little available current, and other unexpected issues. My eventual realizations were the need to connect the 3 GNDs (the 2 power supplies and the Arduino) for a common reference voltage and to include a large capacitor (470 μF) between the 5V and GND of each motor to accommodate servo current spikes (see Figure 5). While there should be no situation in which the motor GND is at higher voltage than its 5V line, I added a diode to hopefully drain current back to the 5V line before the capacitor is damaged.

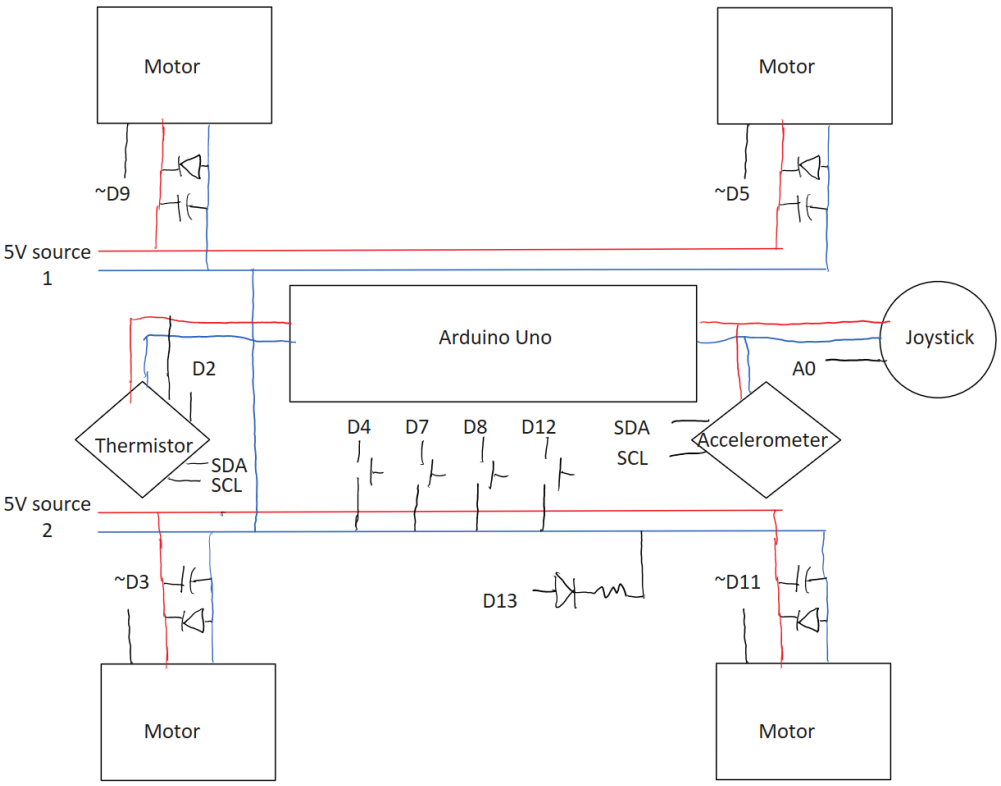


Figure 5: Electronics schematic

4 Code

I created a library called `Desk` to hold most of my functions and created a `Desk` class to provide clarity when library functions are being used. The important functions of my `Desk` library are `moveArm`, which controls individual motors, `movePlatform`, which moves all 4 arms at once, `levelPlatform`, which moves individual arms based on accelerometer readings to level the platform, and `getContDeltaHeight`, which reads the joystick to adjust the platform height. To more easily reference my code, see my GitHub folder or the code at the end of this document.

Within the main script, the thermistor is connected as an interrupt to pin 2, which turns on an LED when the temperature is too hot and turns it off when the temperature is acceptable

again. The `readButtons` function was for my own troubleshooting and is not a main feature of the desk, so it was not included in the library.

4.1 Solved Code Challenges

One challenge I encountered was using the servo library within my library. My solution was to create the servo objects and their pins as private variables in my library header and to fill the pin values when the desk object is created in my main script. The servo pins must be attached in the main setup loop, so I created an additional function, `initialize`, to do this when called in setup.

Another difficulty was my mismatched motors. I had 1 true servo motor with limited rotation and 3 continuous rotation servo (CRS) motors, all in the same VEX-compatible outer casing. I attached all of the motors the same way, but within `moveArm`, I had to control the true servo differently than the CRS motors. The CRS motors take a speed and run until stopped, while the true servo takes a position, moves there, then holds. The solution was to empirically find how long the CRS motors had to move at a given speed to match some position change in the true servo.

5 Materials

The majority of the hardware as well as the motors came from an old VEX V.5 kit. The remaining electronics came from BE121 except for the capacitors and some extra lengths of wire for the accelerometer. Some other miscellaneous parts came from around my house.

Part	**Number**	**Source**	**Cost**
Arduino Uno	1	Arduino Parts Pal	-
LIS3DH Accelerometer	1	BE121	-
MCP9808 Thermistor	1	BE121	-
Parallax 2-Axis Joystick	1	BE121	-
USB cable	3	Spare parts	-
5V, 1A AC power adapter	2	Spare parts	-
Arduino barrel jack	1	BE121	-
Breadboard	3	Spare parts/BE121	-
470 μF capacitor	4	Industrial Electronics	$5
Red LED	5	Arduino Parts Pal	-
Male-male wires	-	Arduino Parts Pal	-
Wire	-	Science and Surplus	$1
Push button	4	Arduino Parts Pal	-
1x25 steel bar	4	Vex V.5	-
2x1x16 Steel chassis rails	4	Vex V.5	-
3x1x15 Steel chassis rails	4	Vex V.5	-
5x15 steel plate	2	Vex V.5	-
VEX servo motor	1	Vex V.5	-
VEX continuous rotation servo	3	Vex V.5	-
Motor clutch	4	Vex V.5	-
Square clutch post	4	Vex V.5	-

Part *cont.*	**Number**	**Source**	**Cost**
#6-32 1/2" screw	8	Vex V.5	-
12T plastic gear	4	Vex V.5	-
60T plastic gear	4	Vex V.5	-
2" square metal shaft	4	Vex V.5	-
3" square metal shaft	4	Vex V.5	-
Bearing flat	13	Vex V.5	-
15-hole Lego rail	6	Lego toys	-
Lego hole-connector	8	Lego toys	-
Plastic spacer 4.6mm	12	Vex V.5	-
Plastic spacer 8mm	8	Vex V.5	-
Shaft collar	16	Vex V.5	-
1/2" #8-32 standoff	4	Vex V.5	-
1" #8-32 standoff	8	Vex V.5	-
#8-32 3/4" screw	12	Vex V.5	-
#8-32 1/2" screw	23	Vex V.5	-
#8-32 3/8" screw	4	Vex V.5	-
#8-32 1/4" screw	12	Vex V.5	-
Keps nuts	34	Vex V.5	-
Pillow block bearing	1	Vex V.5	-
Metal washers	16	Vex V.5	-
Plastic washers	4	Vex V.5	-
Rubber bands	-	Spare parts	-

6 Remaining Challenges

Despite all of the challenges I was able to overcome, some barriers remain to true functionality. The chief problem is that the desk remains unable to raise the platform evenly by itself. The motors seem to have different functional torques despite being set up in the same way. This is understandable for the 1 true servo compared with the 3 CRS motors, however even the CRS motors do not perform evenly. The side with 2 CRS motors is particularly weak. This could be a power issue, as there are 2 motors per power supply. Ideally, each motor would individually receive 1.5 A at 5 V.

My `levelPlatform` function is meant to resolve slight discrepancies in the motors, however the motors still have difficulty correcting. Another issue faced with `levelPlatform` is that it is set to either raise the low arms up or to lower the high arms down. If set to go up, this often results in the platform remaining uneven but being raised very high up. If set to go down, the platform is unable to raise itself since the motors able to raise their arms during movePlatform go back down to the starting position during `levelPlatform` to relevel the platform. A rubber band around the two weak arms helps to pull them up and towards each other, but only after the angle between the band and arms becomes large enough. When at their lowest position, the initial movement of the weak arms still requires my help.

Finally, the arms are too short to raise the platform as high as desired, and the lowest position

provides a good screen height for my laptop but an awkward typing position.

References

1. Contardo Ayala, A. M. *et al.* The impact of height-adjustable desks and prompts to break-up classroom sitting on adolescents' energy expenditure, adiposity markers and perceived musculoskeletal discomfort. *PLOS ONE* **13** (ed Zeeb, H.) e0203938. ISSN: 1932-6203. https://dx.plos.org/10.1371/journal.pone.0203938 (Sept. 2018).
2. Finch, L. E., Tomiyama, A. J. & Ward, A. Taking a stand: The effects of standing desks on task performance and engagement. *International Journal of Environmental Research and Public Health* **14.** ISSN: 16604601 (Aug. 2017).
3. *VEX 3-Wire Servo - Motors and Gears - VEX V5/EDR* https://store.robotmesh.com/vex-robotics/motors-and-gears/vex-servo (2020).
4. Mikey. *Servo with external power* Oct. 2010. https://captain-slow.dk/2010/10/19/servo-with-external-power/ (2020).

7 Appendix: Full Code

The first page of code below is the main script (`.ino`). The second is the library header file (`.h`). The following 4 pages are the library source file (`.cpp`)

```
/*
  deskMain.ino - Main script for controlling motorized standing desk.
  Created by Simon Ng, May 29, 2020. Last edit June 5, 2020
*/
#include "Desk.h"

Desk myDesk(9, 5, 11, 3, 0); // create a desk object called "myDesk"

String dir;
byte PIN_btn1 = 4;
byte PIN_btn2 = 7;
byte PIN_btn3 = 8;
byte PIN_btn4 = 12;
byte PIN_ThermAlert = 2;
byte PIN_ThermLED = 13;

volatile boolean state = LOW;

void setup() {
  Serial.begin(9600);
  myDesk.initialize();
  pinMode(PIN_btn1, INPUT_PULLUP); // button servo 1
  pinMode(PIN_btn2, INPUT_PULLUP); // button CRS 2
  pinMode(PIN_btn3, INPUT_PULLUP); // button CRS3
  pinMode(PIN_btn4, INPUT_PULLUP); // button CRS4

  myDesk.configureThermistor();
  pinMode(PIN_ThermAlert, INPUT_PULLUP);
  pinMode(PIN_ThermLED, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(PIN_ThermAlert), ThermAlert, CHANGE); // when voltage changes
}

void loop() {
  delay(500); // this is necessary to keep motors from small steps each loop
  int dHeight = myDesk.getContDeltaHeight(); // get change in desk height
  if (dHeight <= 0) dir = "Down"; // get direction
  else dir = "Up";
  myDesk.movePlatform(dir, dHeight);
  delay(500);
  readButtons();
  //myDesk.printThermistorDiagnostics();
  //myDesk.printAccel();
  myDesk.levelPlatform(); // adjust motors until accelerometer reads level
  // myDesk.Troubleshoot();
}

void ThermAlert() { // temperature interrupt
  digitalWrite(PIN_ThermLED, state);
  state = !state;
}

void readButtons() { // manual lift buttons
  if (!digitalRead(PIN_btn1)) {
    myDesk.moveArm(1, "up", 50, true);
  }
  else if (!digitalRead(PIN_btn2)) {
    myDesk.moveArm(2, "up", 50, true);
  }
  else if (!digitalRead(PIN_btn3)) {
    myDesk.moveArm(3, "up", 50, true);
  }
  else if (!digitalRead(PIN_btn4)) {
    myDesk.moveArm(4, "up", 50, true);
  }
}
```

```
/*
  Desk.h - Library for controlling motorized standing desk.
  Created by Simon Ng, May 29, 2020. Last edit June 5, 2020
*/

#ifndef Desk_h // protects against bugs if user #includes this library twice
#define Desk_h // protects against bugs if user #includes this library twice

#include "Arduino.h"
#include <Servo.h>
#include<Wire.h>

class Desk
{
  public: // need to access these function externally from main script
    Desk(byte PIN_servo1, byte PIN_servo2, byte PIN_servo3, byte PIN_servo4, byte PIN_joystick);
    void initialize();
    void moveArm(int servo, String dir, int distance, boolean useDistance);
    void movePlatform(String dir, int distance);
    void stopPlatform();
    void levelPlatform();
    void cofigureAccelerometer();
    void getAccel(float * ptrToArray);
    void printAccel();
    int getDeltaHeight();
    int getContDeltaHeight();
    void configureThermistor();
    void printThermistorDiagnostics();
    void printByte(byte myByte);
    float getAccelIn(char accelDir);
    void movePlatformGradual(String dir, int velocity);
    void Troubleshoot();

  private: // create servo objects and accelerometer variables as internal, private variables
    byte _PIN_servo1;
    byte _PIN_servo2;
    byte _PIN_servo3;
    byte _PIN_servo4;
    byte _PIN_joystick;
    Servo servo1;  // create servo object to control a servo
    Servo servo2;  // create servo object to control a servo
    Servo servo3;  // create servo object to control a servo
    Servo servo4;  // create servo object to control a servo
    float _accel[3];
    int _joyPosOld = 0;
};

#endif // protects against bugs if user #includes this library twice
```

```cpp
/*
  Desk.cpp - Library for controlling motorized standing desk.
  Created by Simon Ng, May 29, 2020. Last edit June 5, 2020
*/

#include "Desk.h"

/*==================== OBJECT SETUP ====================*/
Desk::Desk(byte PIN_servo1, byte PIN_servo2, byte PIN_servo3, byte PIN_servo4, byte PIN_joystick) { // should be 9, 5, 11, 3, 0
  _PIN_servo1 = PIN_servo1; // PWM pin
  _PIN_servo2 = PIN_servo2; // PWM pin
  _PIN_servo3 = PIN_servo3; // PWM pin
  _PIN_servo4 = PIN_servo4; // PWM pin
  _PIN_joystick = PIN_joystick; // analog pin
}

void Desk::initialize() {
  servo1.attach(_PIN_servo1);
  servo2.attach(_PIN_servo2);
  servo3.attach(_PIN_servo3);
  servo4.attach(_PIN_servo4);

  cofigureAccelerometer(); // set up accelerometer
}

/*==================== SERVOS ====================*/
void Desk::moveArm(int servo, String dir, int distance, boolean useDistance) { // move an individual motor arm
  int dirValue = 90; // if input direction is incorrect, don't move CRS
  int forwardSpeed = 160; // for CRS (90 - 180)
  int reverseSpeed = 20; // for CRS (0 - 90)

  if (servo == 1) { // actual servo motor
    if (dir == "UP" || dir == "Up" || dir == "up") {
      dirValue = servo1.read() + distance / 20;
      Serial.println(servo1.read());
    }
    else if (dir == "DOWN" || dir == "Down" || dir == "down") {
      dirValue = servo1.read() - distance / 20;
      Serial.println(servo1.read());
    }
    else { // invalid entry, hold position
      dirValue = servo1.read(); // holds position
      Serial.println(servo1.read());
    }
    servo1.write(dirValue);
  }
  else if (servo == 2) { // continuous rotation servo
    if (dir == "UP" || dir == "Up" || dir == "up") {
      dirValue = forwardSpeed; // can be edited to adjust for motor inconsistencies
    }
    else if (dir == "DOWN" || dir == "Down" || dir == "down") {
      dirValue = reverseSpeed; // can be edited to adjust for motor inconsistencies
    }
    servo2.write(dirValue);
  }
  else if (servo == 3) { // continuous rotation servo
    if (dir == "UP" || dir == "Up" || dir == "up") {
      dirValue = reverseSpeed; // can be edited to adjust for motor inconsistencies
    }
    else if (dir == "DOWN" || dir == "Down" || dir == "down") {
      dirValue = forwardSpeed; // can be edited to adjust for motor inconsistencies
    }
    servo3.write(dirValue);
  }
  else if (servo == 4) { // continuous rotation servo
    if (dir == "UP" || dir == "Up" || dir == "up") {
      dirValue = forwardSpeed; // can be edited to adjust for motor inconsistencies
    }
    else if (dir == "DOWN" || dir == "Down" || dir == "down") {
      dirValue = reverseSpeed; // can be edited to adjust for motor inconsistencies
    }
    servo4.write(dirValue);
  }
  if (useDistance) { // can set moveArm to execute individual arm move or just turn motor on to move all at once for movePlatform
    delay(abs(distance));
  }
}

void Desk::movePlatform(String dir, int distance) { // move entire platform at once
  int t_spin = abs(distance);
  moveArm(1, dir, abs(distance), false);
  moveArm(2, dir, abs(distance), false);
  moveArm(3, dir, abs(distance), false);
  moveArm(4, dir, abs(distance), false);
  delay(t_spin);
  stopPlatform();
}
```

```
void Desk::movePlatformGradual(String dir, int velocity) { // doesn't work very well. based on moving each arm until accelerometer reads okay
  int cycles = round(velocity / 10);
  for (int i = 0; i < cycles ; i++) {
    moveArm(1, dir, 100, false); // needs tuning
    while (getAccelIn('X') > 0.1) {
      moveArm(2, dir, 0, false);
    }
    while (getAccelIn('Y') < -0.1) {
      moveArm(3, dir, 0, false);
      moveArm(4, dir, 0, false);
    }
  }
}

void Desk::stopPlatform() { // halt movement of all arms
  servo1.write(servo1.read()); // this holds its position
  servo2.write(90); // holds by torque
  servo3.write(90); // holds by torque
  servo4.write(90); // holds by torque
}

void Desk::levelPlatform() { // attempt to level platform based on accelerometer reading
  int tiltFactor = 600; // how large movements are to level platform
  float threshold = 0.07; // how flat the desk should be
  getAccel(_accel);
  float xAccel = _accel[2]; // Z according to accelerometer chip
  float yAccel = _accel[0]; // X according to accelerometer chip
  String mode = "down"; // should leveling mechanism be to raise low arms or to lower high arms?

  if (mode == "up") {
    if (xAccel > threshold) {
      moveArm(2, "up", abs(xAccel) * tiltFactor, true);
      moveArm(3, "up", abs(xAccel) * tiltFactor, true);
    }
    else if (xAccel < -threshold) {
      moveArm(1, "up", abs(xAccel) * tiltFactor, true);
      moveArm(4, "up", abs(xAccel) * tiltFactor, true);
    }
    if (yAccel > threshold) {
      moveArm(1, "up", abs(yAccel) * tiltFactor, true);
      moveArm(2, "up", abs(yAccel) * tiltFactor, true);
    }
    else if (yAccel < -threshold) {
      moveArm(3, "up", abs(yAccel) * tiltFactor, true);
      moveArm(4, "up", abs(yAccel) * tiltFactor, true);
    }
  }
  else if (mode == "down") {
    if (xAccel < -threshold) {
      moveArm(2, "down", abs(xAccel) * tiltFactor, true);
      moveArm(3, "down", abs(xAccel) * tiltFactor, true);
    }
    else if (xAccel > threshold) {
      moveArm(1, "down", abs(xAccel) * tiltFactor, true);
      moveArm(4, "down", abs(xAccel) * tiltFactor, true);
    }
    if (yAccel < -threshold) {
      moveArm(1, "down", abs(yAccel) * tiltFactor, true);
      moveArm(2, "down", abs(yAccel) * tiltFactor, true);
    }
    else if (yAccel > threshold) {
      moveArm(3, "down", abs(yAccel) * tiltFactor, true);
      moveArm(4, "down", abs(yAccel) * tiltFactor, true);
    }
  }
}

/*==================== ACCELEROMETER ====================*/
void Desk::cofigureAccelerometer() {
  Wire.begin(); // join i2c bus (address optional for master)

  // CTRL_REG1
  Wire.beginTransmission(0b0011000);  // this is our 7bit device address
  Wire.write(byte(0b0100000)); // set pointer to go to control register 1
  Wire.write(byte(0b01110111)); // set to High Res / Normal / Low-power mode (400 Hz)
  Wire.endTransmission();     // stop transmitting

  // CTRL_REG4
  Wire.beginTransmission(0b0011000);  // this is our 7bit device address
  Wire.write(byte(0b0100011)); // set pointer to go to control register 1
  Wire.write(byte(0b10001000)); // set to High Res / Normal / Low-power mode (400 Hz)
  Wire.endTransmission();     // stop transmitting
}

void Desk::getAccel(float * ptrToArray) { // passing pointer to a float
  byte XL;
```

```
  byte XH;
  byte YL;
  byte YH;
  byte ZL;
  byte ZH;
  Wire.beginTransmission(0b0011000);  // this is our 7bit device address
  Wire.write(0b10101000); // set to read desired address
  Wire.endTransmission();     // stop transmitting
  Wire.requestFrom(0b0011000, 6);
  if (Wire.available()) { // is there data to be read? If more than 0 bytes, read it
    XL = Wire.read(); // this is being read
    XH = Wire.read(); // this is being read
    YL = Wire.read(); // this is being read
    YH = Wire.read(); // this is being read
    ZL = Wire.read(); // this is being read
    ZH = Wire.read(); // this is being read
  }
  *ptrToArray = ((XH << 8) + XL) / pow(2, 15) * 2.0;
  ptrToArray++;
  *ptrToArray = ((YH << 8) + YL) / pow(2, 15) * 2.0;
  ptrToArray++;
  *ptrToArray = ((ZH << 8) + ZL) / pow(2, 15) * 2.0;
}

void Desk::printAccel() { // should take in array
  getAccel(_accel); // passes pointer to beginning of array
  float X = _accel[2]; // Z according to accelerometer chip
  float Y = _accel[0]; // X according to accelerometer chip
  float Z = _accel[1]; // Y according to accelerometer chip

  Serial.print("X ");
  Serial.println(X, 6);
  Serial.print("Y ");
  Serial.println(Y, 6);
  Serial.print("Z ");
  Serial.println(Z, 6);
  Serial.println();
}

float Desk::getAccelIn(char accelDir) { // return just the direction of acceleration specified
  byte XL;
  byte XH;
  byte YL;
  byte YH;
  byte ZL;
  byte ZH;
  Wire.beginTransmission(0b0011000);  // this is our 7bit device address w/ SD0 to GND
  Wire.write(0b10101000); // set to read desired address
  Wire.endTransmission();     // stop transmitting
  Wire.requestFrom(0b0011000, 6);
  if (Wire.available()) { // is there data to be read? If more than 0 bytes, read it
    XL = Wire.read(); // this is being read
    XH = Wire.read(); // this is being read
    YL = Wire.read(); // this is being read
    YH = Wire.read(); // this is being read
    ZL = Wire.read(); // this is being read
    ZH = Wire.read(); // this is being read
  }
  if (accelDir == 'X' || accelDir == 'x')      return (((ZH << 8) + ZL) / pow(2, 15) * 2.0); // Z according to accelerometer chip
  else if (accelDir == 'Y' || accelDir == 'y') return (((XH << 8) + XL) / pow(2, 15) * 2.0); // X according to accelerometer chip
  else if (accelDir == 'Z' || accelDir == 'z') return (((YH << 8) + YL) / pow(2, 15) * 2.0); // Y according to accelerometer chip
  return 0;
}


/*==================== JOYSTICK ====================*/
int Desk::getDeltaHeight() {
  int joyOffset = 10; // controls how much joystick must move before activating platform move
  int joyPosNew = analogRead(_PIN_joystick) - 518;
  int joyPosTemp;
  if (abs(joyPosNew) > abs(_joyPosOld) + joyOffset) { // joystick has increased since last time
    joyPosTemp = _joyPosOld; // store old position before updating
    _joyPosOld = joyPosNew; // updated old position
    //Serial.print("Joystick pos: "); Serial.println(_joyPosOld);
    return (0.5 * (joyPosNew - joyPosTemp)); // amount to move, including correct sign
  }
  else if (abs(joyPosNew) < 30) { // joystick at center
    _joyPosOld = 0; // don't reset reference position until joystick is at center
  }
  return 0; // otherwise return no height change
}

int Desk::getContDeltaHeight() { // take current joystick position for move command
  int sensitivity = 5; // low number is non-responsive, high number is very responsive
  int joyPos = analogRead(_PIN_joystick) - 518;
  if (abs(joyPos) < 30) { // joystick at center
    return 0; // don't move anything
```

```
  }
  return joyPos / sensitivity;
}

/*==================== THERMISTOR ====================*/
void Desk::configureThermistor() {
  int t_limit = 30; // alert temperature

  // Activate Alarm
  Wire.beginTransmission(0b0011001);  // this is our 7bit device address 0011000 default mcp9808 . Write address byte
  Wire.write(byte(0x01)); // set pointer to go to configuration register
  Wire.write(byte(0b00000000)); // leave high byte defaults
  Wire.write(byte(0b00001000)); // turn on Alert, leave rest of bits as defaults (nothing locked and in comparator mode)
  Wire.endTransmission();     // stop transmitting

  // Set resolution
  Wire.beginTransmission(0b0011001);  // this is our 7bit device address 0011000 default mcp9808 . Write address byte
  Wire.write(byte(0b00001000)); // set pointer to go to temperature register
  Wire.write(byte(0b00000000)); // change resolution to +-0.5C
  Wire.endTransmission();     // stop transmitting

  // Set T_upper
  int intT_up = round(t_limit * 4);
  byte highT_up = intT_up >> 6;
  byte lowT_up = intT_up << 2;
  Wire.beginTransmission(0b0011001);  // this is our 7bit device address 0011000 default mcp9808 . Write address byte
  Wire.write(byte(0b0000010)); // set pointer to go to upper temp limit register
  Wire.write(highT_up); // set high byte
  Wire.write(lowT_up); // set low byte
  Wire.endTransmission();     // stop transmitting

  // Set T_crit
  float t_crit = t_limit; //C . Max resolution is 0.25 deg
  int intT_crit = round(t_crit * 4);
  byte highT_crit = intT_crit >> 6;
  byte lowT_crit = intT_crit << 2;
  Wire.beginTransmission(0b0011001);  // this is our 7bit device address 0011000 default mcp9808 . Write address byte
  Wire.write(byte(0b0000100)); // set pointer to go to critical temp limit register
  Wire.write(highT_crit); // set high byte
  Wire.write(lowT_crit); // set low byte
  Wire.endTransmission();     // stop transmitting
}

void Desk::printThermistorDiagnostics() { // print digital input and CONFIG register for reference and troubleshooting
  Serial.println();
  Serial.print("AlertPin: ");
  Serial.println(digitalRead(2));

  // Calculate Temp for troubleshooting/reference
  Wire.beginTransmission(0b0011001);  // this is our 7bit device address 0011000 default mcp9808 . Write address byte
  Wire.write(byte(0x05));           // set pointer to read temp
  Wire.endTransmission();     // stop transmitting
  Wire.requestFrom(0b0011001, 2); // request 2 bits from slave device
  if (Wire.available()) { // is there data to be read? If more than 0 bytes, read it
    byte highByt = Wire.read(); // read first byte off buffer, high byte, largest number
    byte lowByt = Wire.read(); // read first byte off buffer, high byte, largest number
    boolean negative = bitRead(highByt, 4); // we want the sign of the number
    highByt = highByt & 0b00001111; // bitwise And
    int intTemperature = (highByt << 8) + lowByt;// int is two bytes. bitshift operator has order of operations
    float MCP9808Temp = intTemperature / 16.0;
    if (negative) MCP9808Temp = -1 * MCP9808Temp;
    Serial.print("Temperature: "); Serial.println(MCP9808Temp, 6);
  }
}

void Desk::printByte(byte myByte) {
  for (int i = 7; i > -1; i--) {
    Serial.print(bitRead(myByte, i));
  }
  Serial.println();
}

/*==================== MISC ====================*/
void Desk::Troubleshoot() { // likely problems I've forgotten about
  Serial.println("MOTORS NOT WORKING");
  Serial.println("Are pin wires contacting the stripped wire?");
  Serial.println("Are the power blocks plugged into the outlet?");
  Serial.println("Is the power bank turned on?");
}
```