Department of Electrical and Computer Engineering

Embedded Systems
EGC433-01

Design of a Simple Tandem Rotor Drone

Group Members	Department
Nikiforos Fokas	EE
Kyle Harkin	CE

Instructor: M. Otis

Abstract

With the emergence of the drone as a potential tool for future use ranging from package delivery to healthcare services, we created this project as a means of prototyping a simple drone design which could be improved upon and developed into a fully functional product. This prototype was created using skills learned throughout the semester in EGC433-01 and previously learned in EGC331-02.

Table of Contents

Objective

The objective of this final design project was to create a functional prototype of a tandem rotor drone with the ability to hover at a fixed altitude.

Design and Implementation

The Tandem Rotor Drone

For the body of our drone, we settled on a tandem rotor design. The reasoning behind choosing a tandem rotor as opposed to a single rotor is because we had multiple DC motors at our disposal, and it was found through experimentation that a tandem rotor design gave us the most stability and lift while maintaining a small formfactor. The drone itself can be seen in Figure 1.

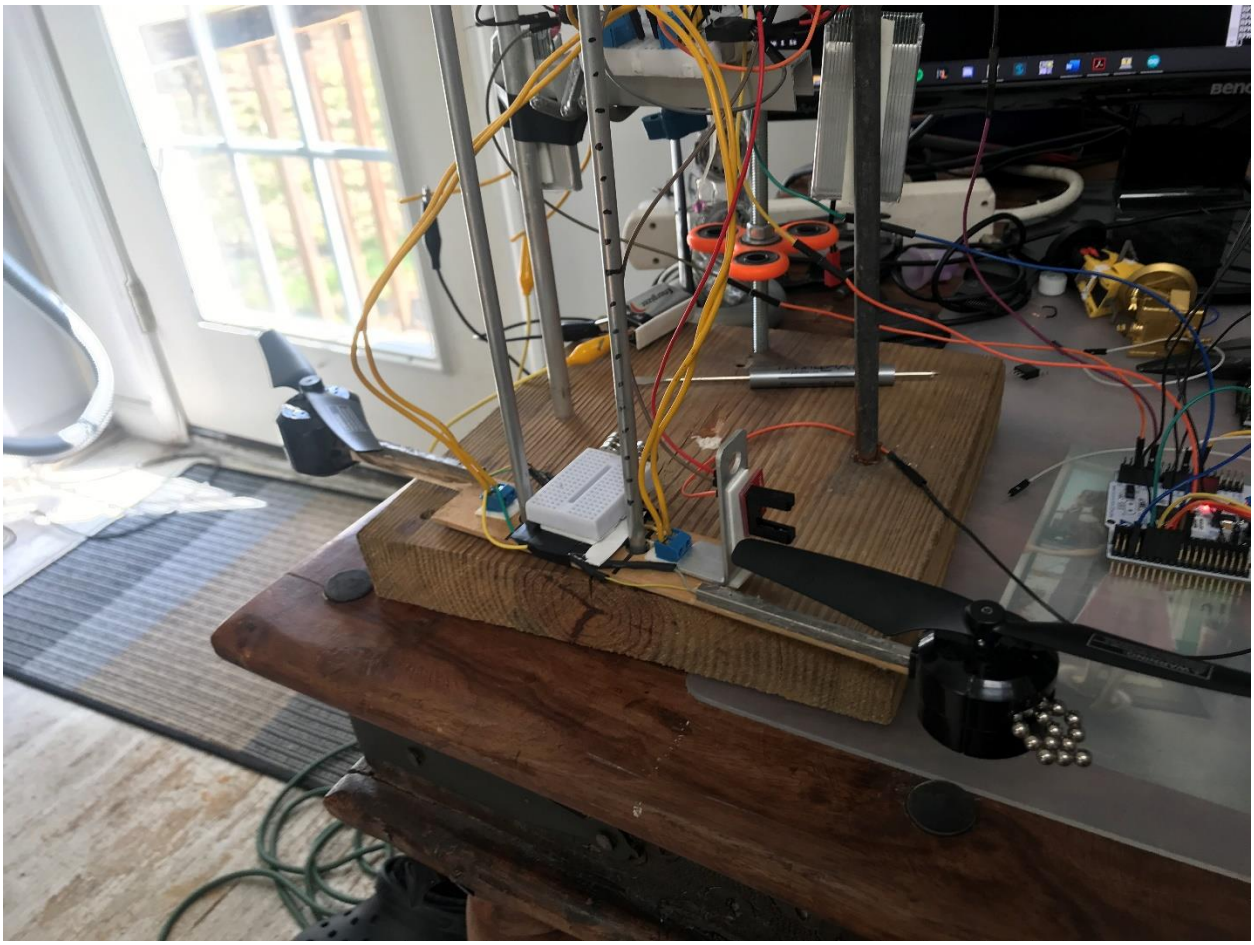


Figure 1. The Tandem Rotor Drone

As shown in Figure 1, there is a photoresistor mounted to an aluminum bracket. This photoresistor is used to find the revolutions per minute (RPM) of the DC motor. The metallic spheres affixed to the motor itself are small magnetic weights used to help balance the drone body when in flight. The small board in the middle of the drone body is used to facilitate wave reflection for the ultra-sonic sensor used as part of the drivers for the drone setup.

The Drivers and Accessories

In order to operate the drone, we had to create multiple driver circuits. Part of the drone drivers is the ultra-sonic sensor. The sensor radiates sound waves to be reflected back to it, and

then calculate the corresponding distance from the sensor. Therefore, we mounted the sensor above the drone to determine the altitude of the drone. If this drone were a single unit, meaning that the drivers were affixed to the body, the sensor would be on the bottom of the drone itself. However, due to the limitations on the amount of lift we were able to generate, we could not affix the drivers to the drone body, therefore they are mounted along the guiding rods. Figure 2 shows the ultra-sonic sensor and guiding rods.

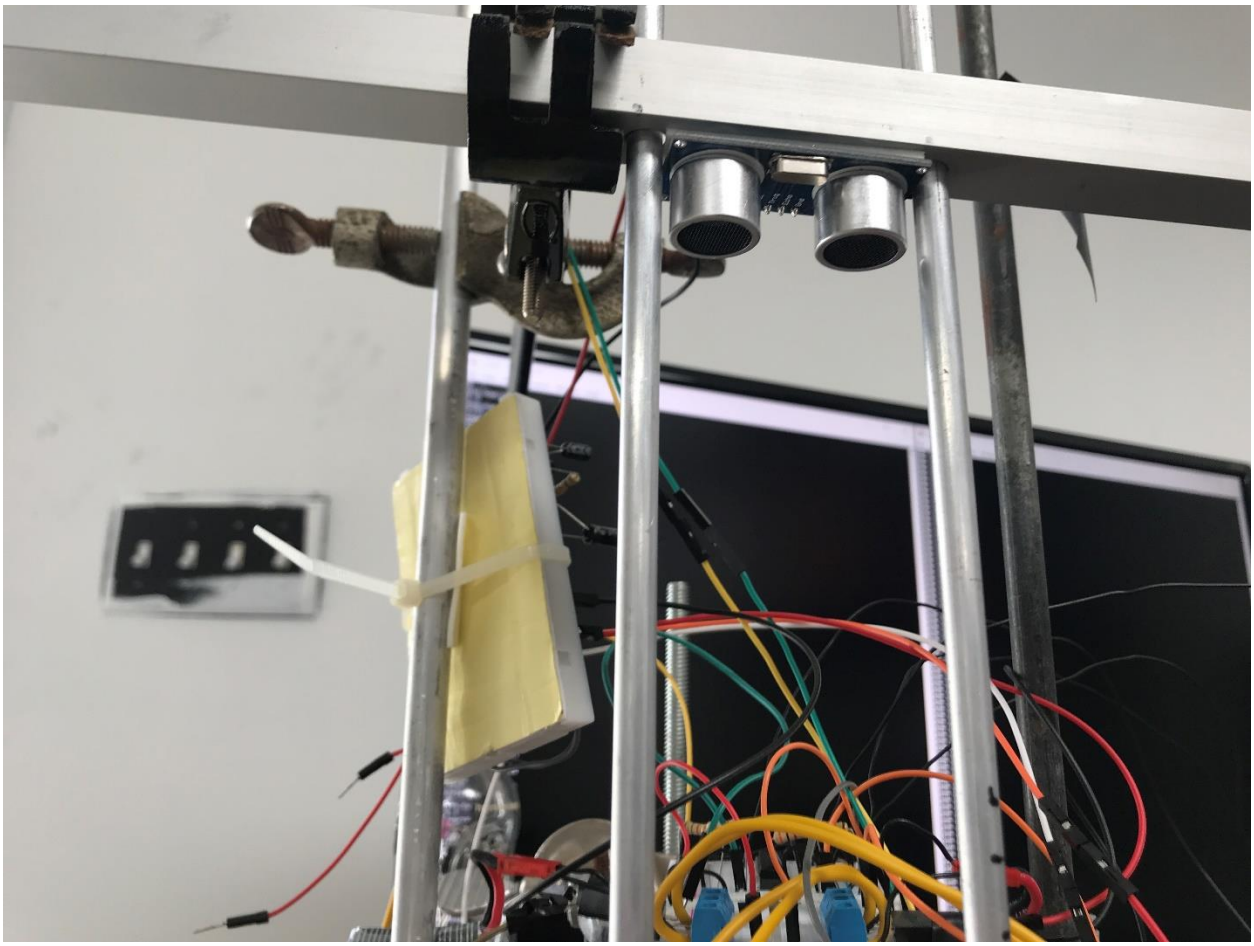


Figure 2. Ultra-Sonic Sensor between the guiding rods

Additionally, shown in Figure 2 is the 5V power supply for the sensor. A better viewing angle of which can be seen in Figure 3.

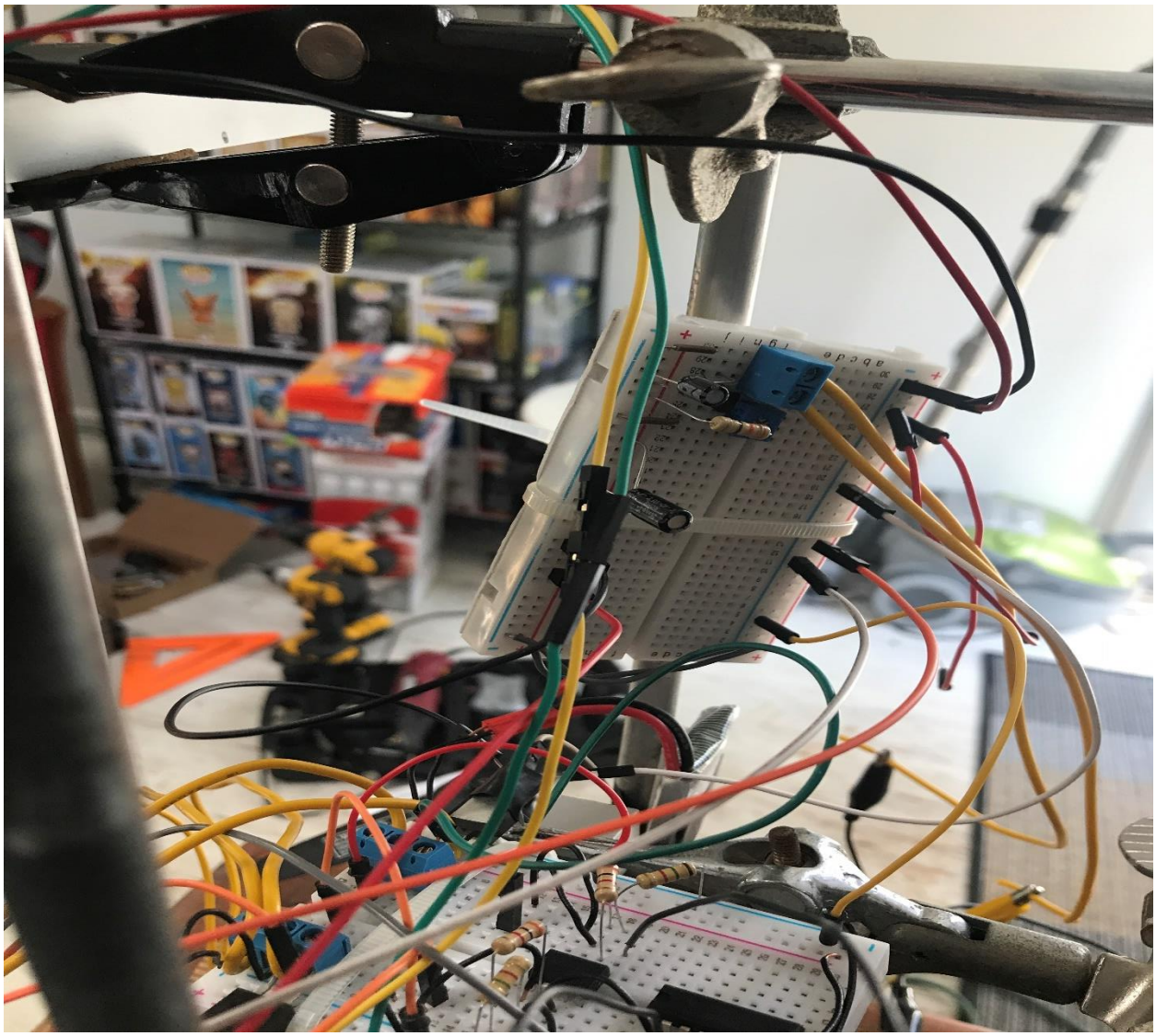


Figure 3. 5V Supply (vertical board)

Also seen in Figure 3 is the motor driver. The motor driver was needed to provide enough amperage and voltage to drive the DC motors, as the microcontroller used did not supply enough. Therefore, a driver was built which took supplemental power from a battery pack and provided the correct amount to the DC motor as required by the microcontroller. The motor driver can be seen in Figure 4.

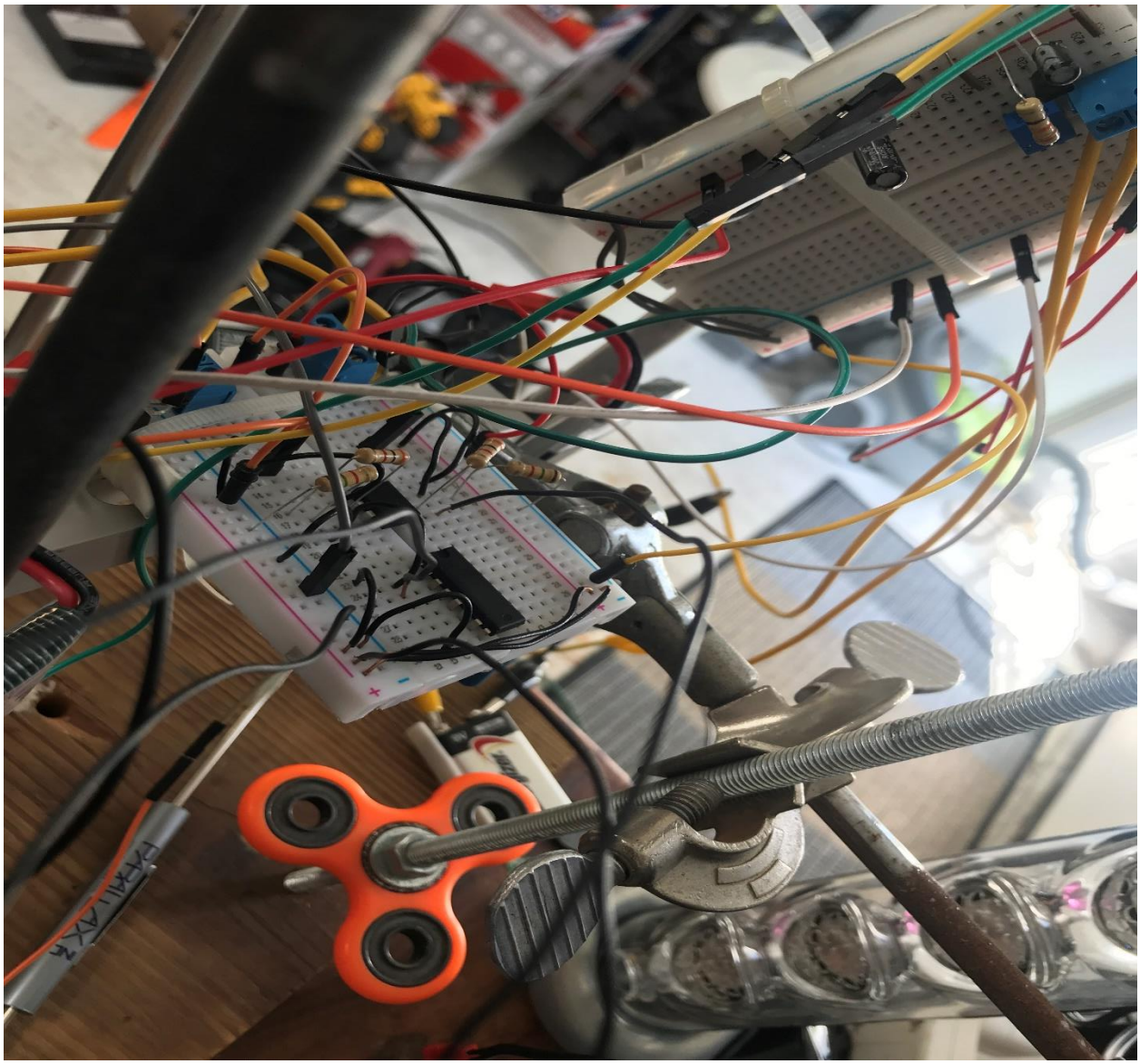


Figure 4. Motor Driver (horizontal board)

All data was controlled using an STM nucleo and an Arduino board. The STM was the main driver for the done, and the Arduino was used to run the phototransistor using an interrupt into the STM, that way we could constantly calculate the RPM as well as control the duty cycle of the motors and height. Figure 5 shows the boards used.

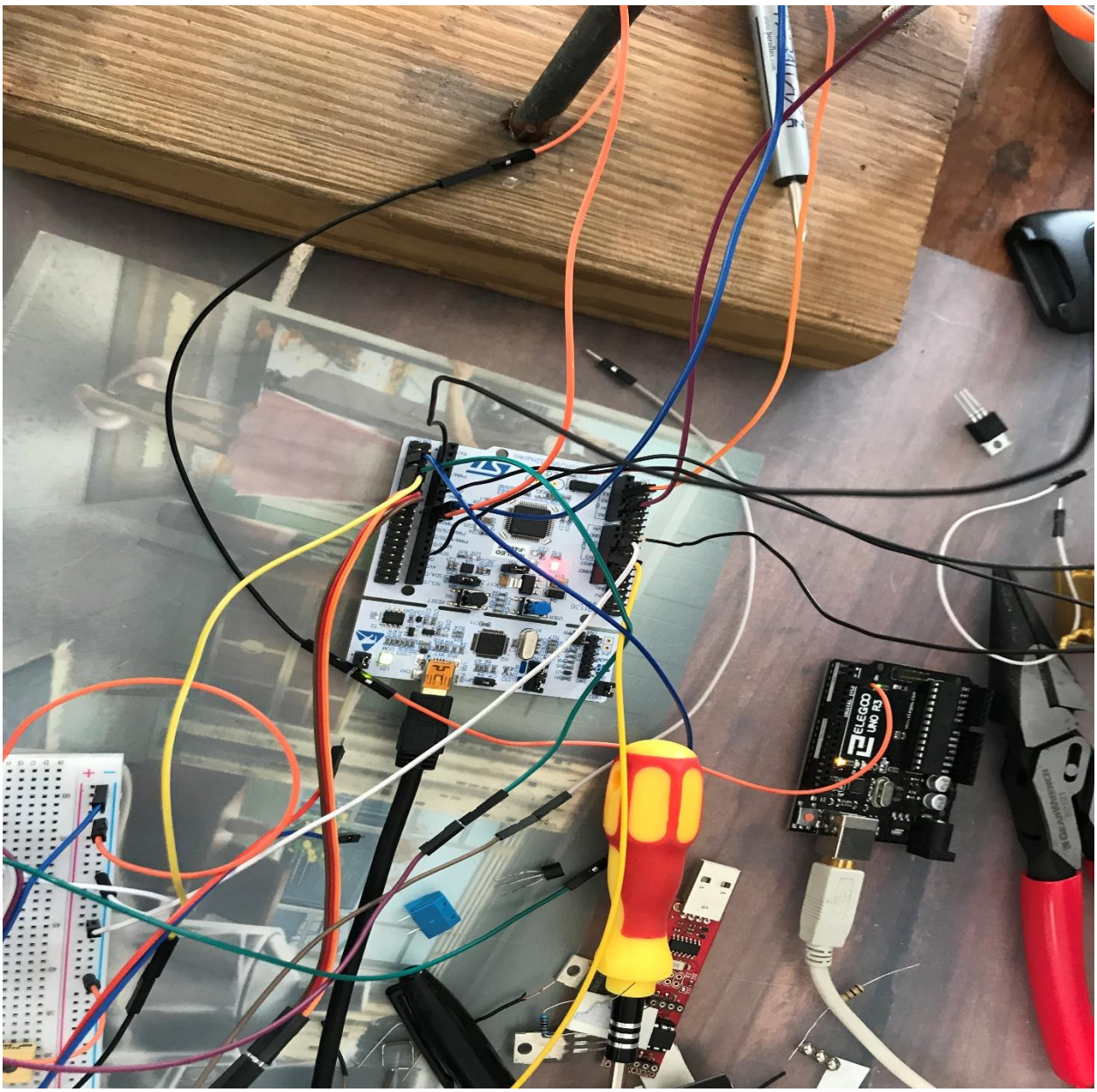


Figure 5. STM and Arduino boards

In order to display relevant information to the user, a liquid crystal display (LCD) was used. The information displayed on the LCD was the current height (H), the desired height (HMax), and the current temperatures of the MOSFETs in the motor driver (T1,T2). Although the heights are easily understood as being needed for the user, perhaps the temperature is not as obvious. The

temperature is output in order to prevent overheating of the motor driver. Should the driver overheat midflight, the drone would come crashing down as the motors would stop. Figure 6 shows an example of LCD output.

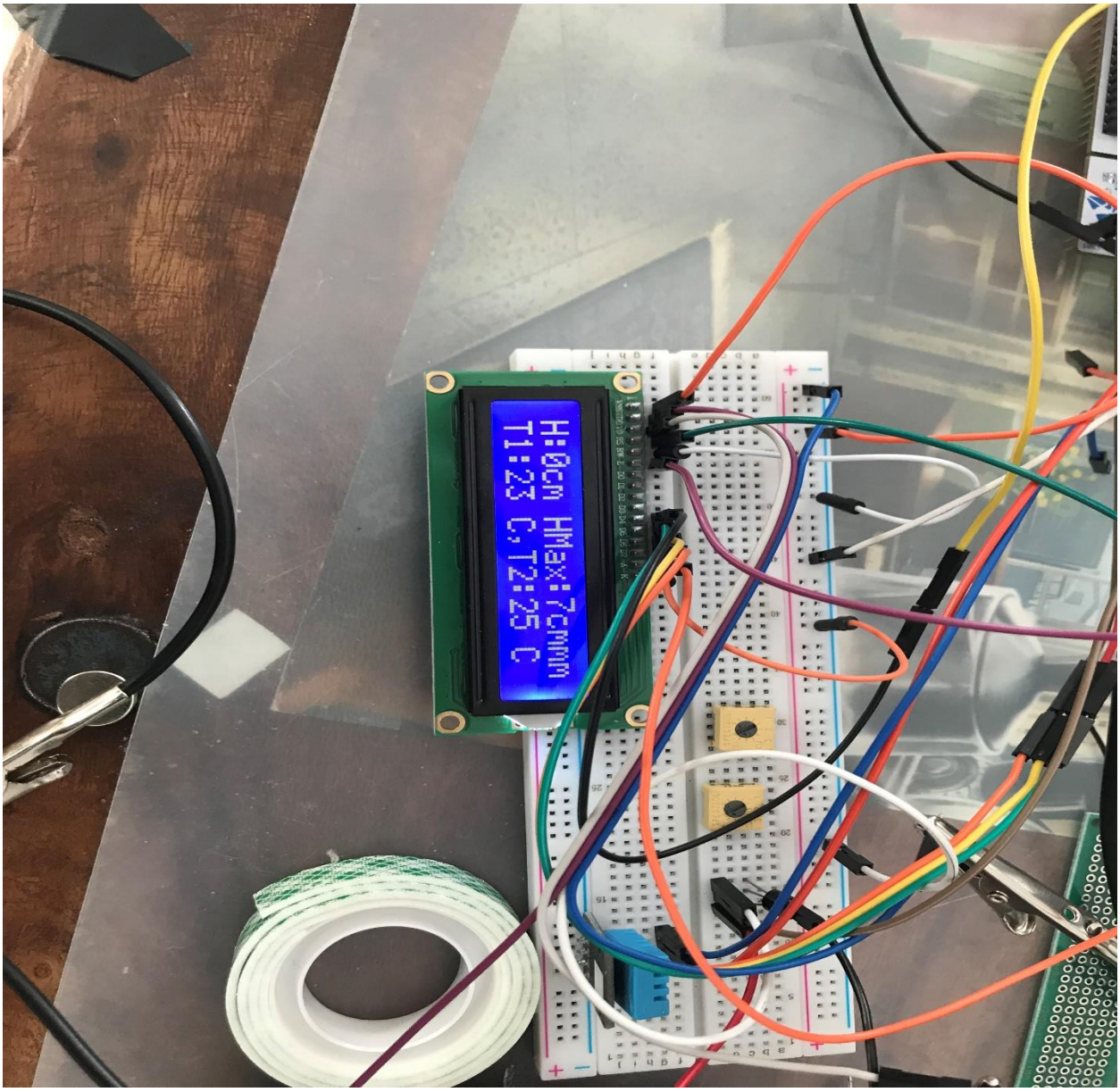


Figure 6. LCD Output

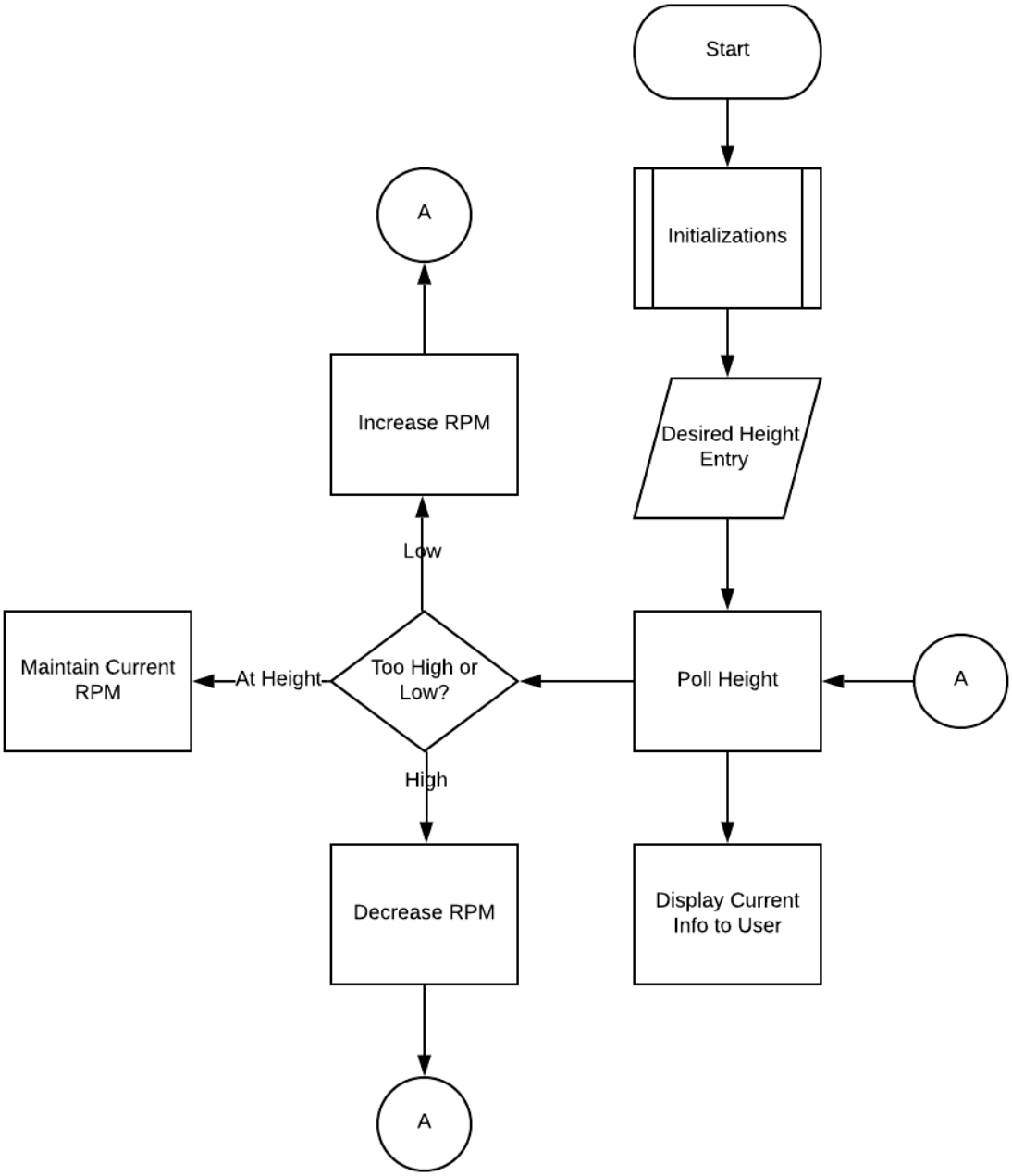
As seen in Figure 6, our set height is 7cm and currently the drone is yet to take off (0cm). The corresponding temperatures are displayed as well to inform the user when they may need to take a break from flying. A project flowchart, the code, compete pinout of all boards as well as schematics of all drivers are available in the Appendix.

Conclusion and Final Thoughts

As a culmination of all skills learned in the Embedded Systems class, EGC433, and previous knowledge obtained form Microcontrollers, EGC331, a tandem rotor drone was designed and successfully implemented. The drone could self-calibrate itself to hover at any given altitude below the upper bound of the guiding rods (approx. 30cm). The most challenging aspect of this project was writing the code to allow the drone to hover and maintain a given altitude. Perhaps this project could be expanded upon in the future to allow Bluetooth control and auto stabilization so it could fly like a commercially available drone.

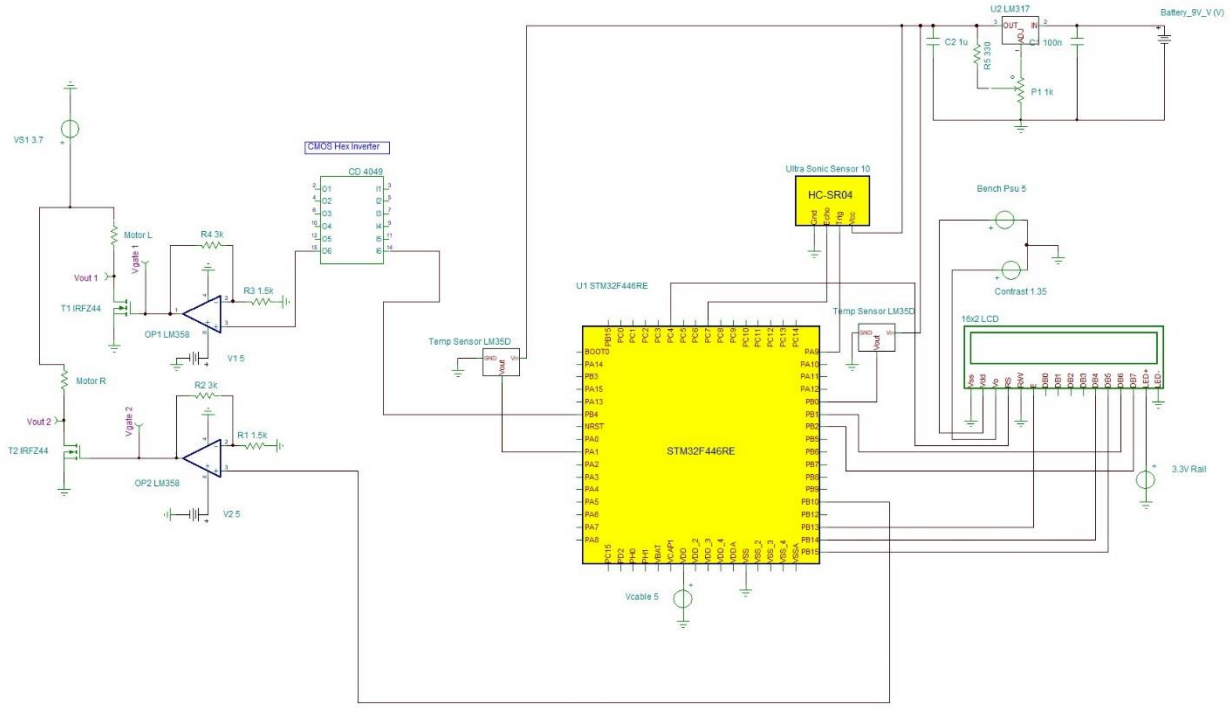
Appendix

Project Flowchart

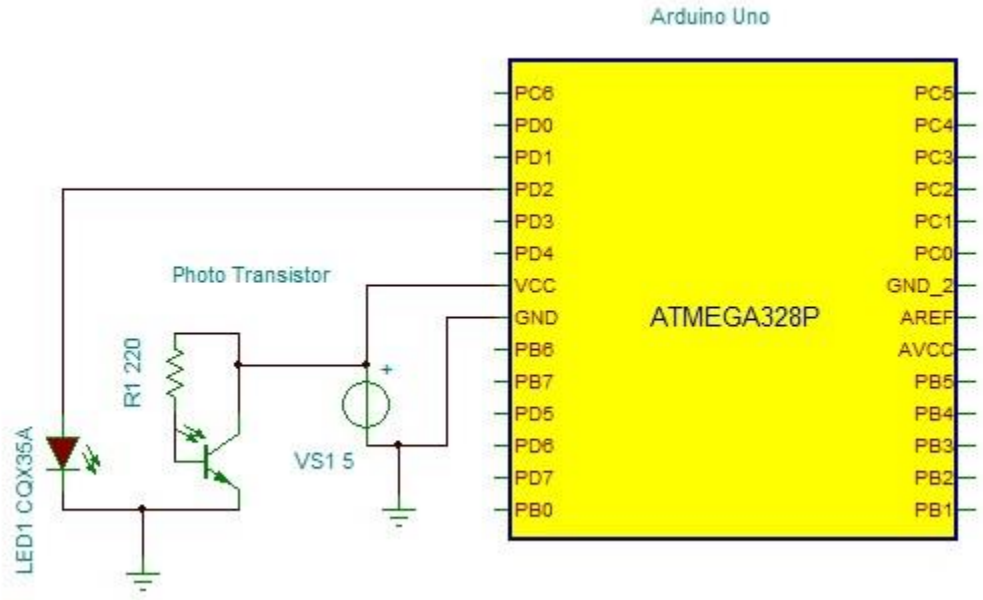


A. Project Flowchart

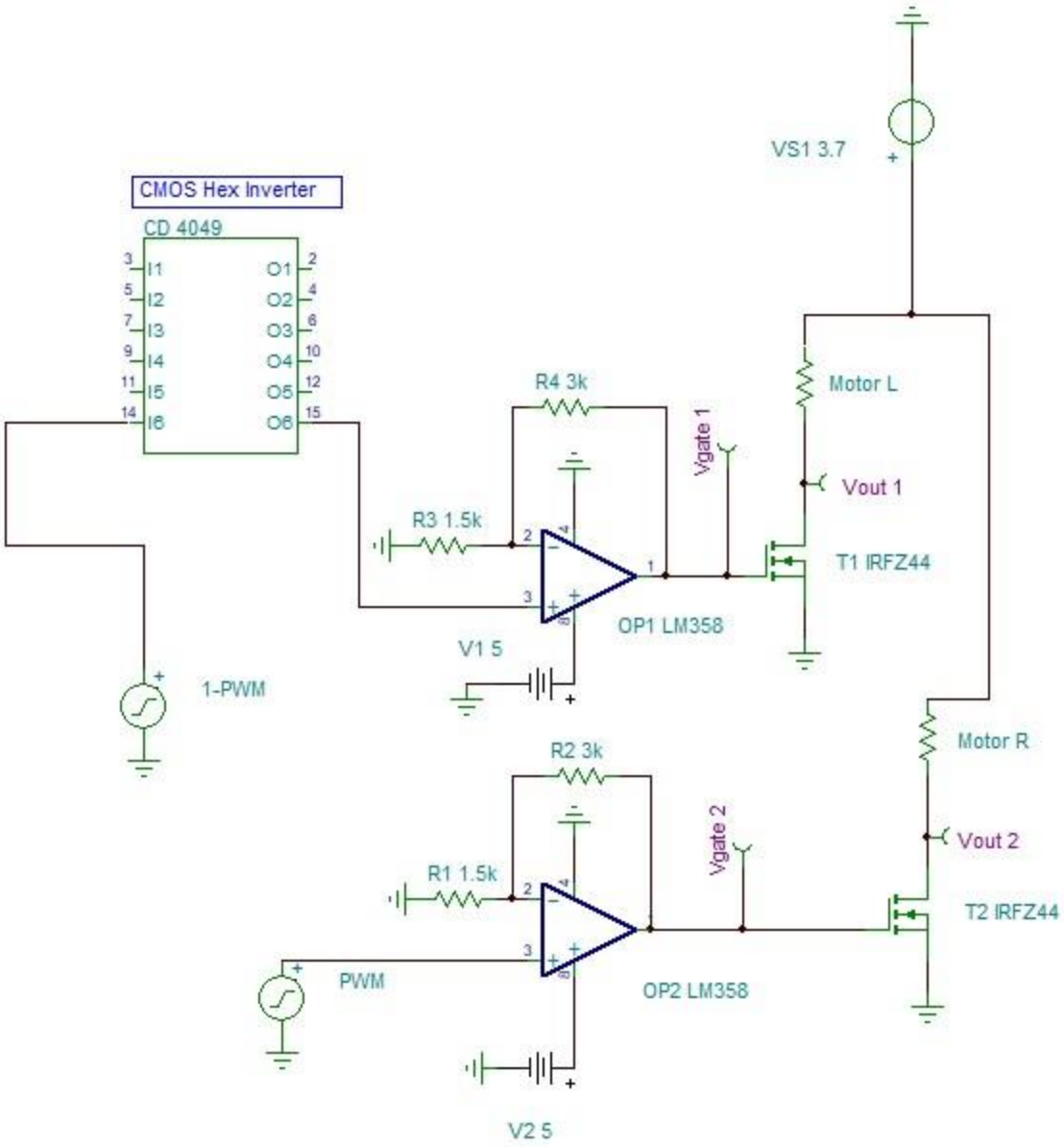
Schematics



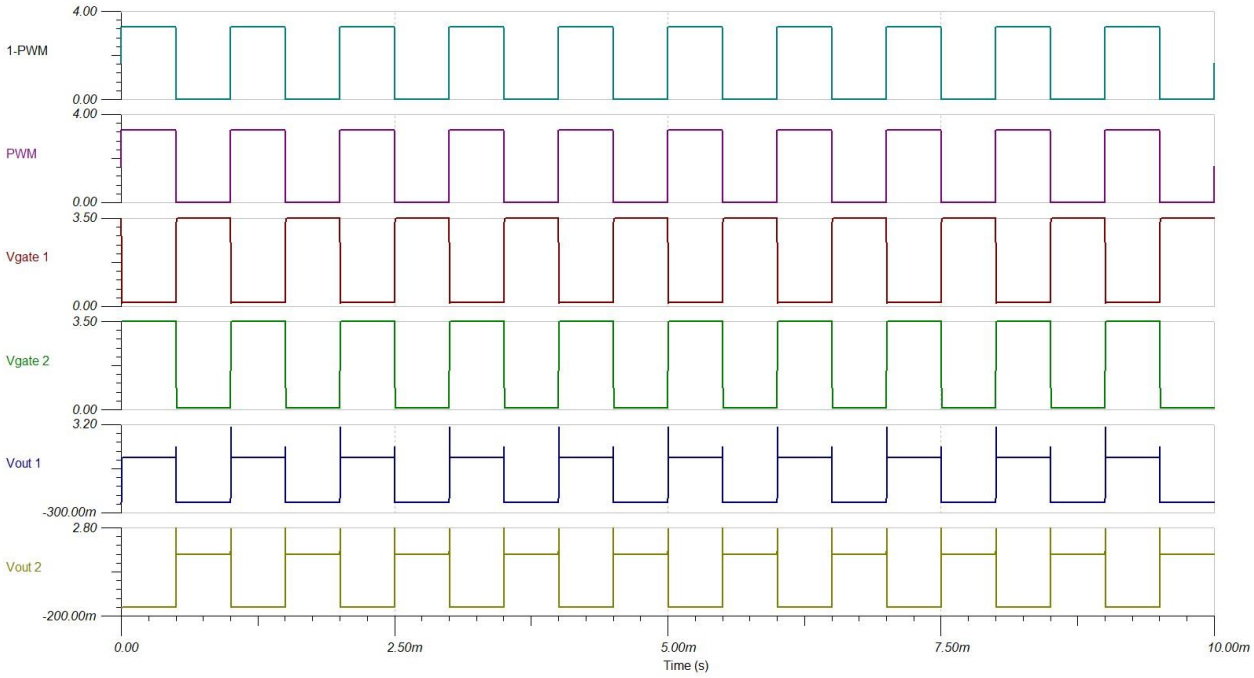
B. STM Pinout



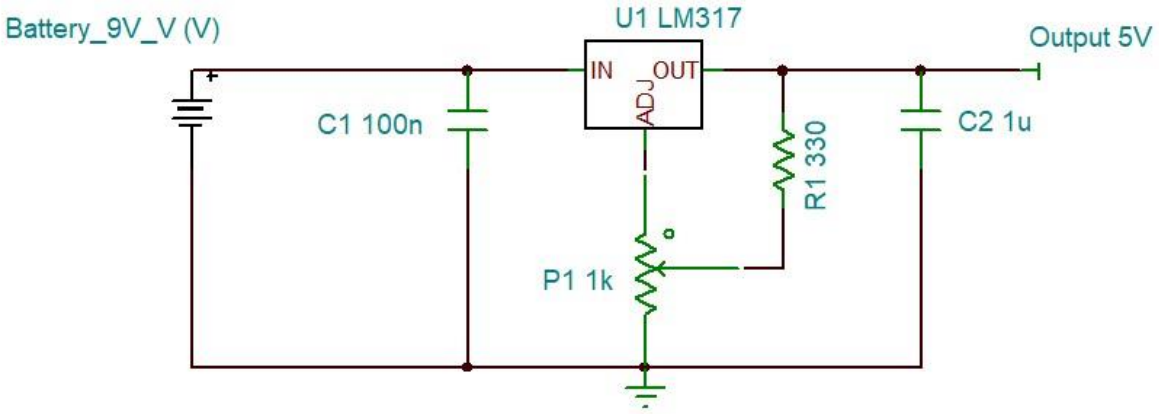
C. Arduino Pinout



D. Motor Driver Schematic



E. Motor Driver Simulation



F. 5V Power Supply Schematic

Project Code

```
#include "mbed.h"
#include "components.h"
#include "hcsr04.h"
#include "TextLCD.h"
#include "math.h"

Timer sysTimer;
Serial pc(SERIAL_TX, SERIAL_RX);
HCSR04 sensor(D8, D9);
PwmOut motor(D6);
PwmOut motor2(D5);
TextLCD lcd(PC_4, PB_13, PB_14, PB_15, PB_1, PB_2); // RS, E, D4-D7


void printInt(int z){
      pc.printf(" %d \n\r",z);
   }

void printFloat(float z){
      pc.printf(" %.1f \n\r",z);
   }

int height = 10;
float mos = 0.0f;
float mos2 = 1.0f - mos;
float x = 0;
int read = 0;
unsigned int avg = 0;
int y = 0;
int i =0;
int t1,t2;
char c;
int l,m,n;



int heliCalibrate(){

   while(i<20){

      read = abs(y - sensor.distance());
      if(read<0 || read > 30){
```

```
        read = abs(y - sensor.distance());
      }
      x = x + read;
      i++;

   }
   unsigned   int d = floor(x/20);
   printInt(d);
   return d;

}


int avgHeight(){

   i =0;
   while(i<10){
      read = abs(y - sensor.distance());
      if(read<0 || read > 30){
         read = abs(y - sensor.distance());
      }
      avg = avg + read;
      i++;
      wait_ms(1);
   }
   return avg/10;
}

int main()
{
   pc.baud(115200);
   motor.period_ms(1);
   motor.write(0.1f);
   motor2.period_ms(1);
   motor2.write(0.1f);
   y = heliCalibrate();
   lcd.cls();
 while(1){
      if(pc.readable()){ //read keyboard input from dummy terminal
            c = pc.getc();
            l = c - '0';
            c = pc.getc();
            m = c - '0';
            n = l+m;
```

```
      pc.printf(" %d \n\r",n);

      height = n;
   }
avg = avgHeight();
while(avg<height && mos<=1.0f && (t1<=60.0f && t2<=60.0f)){
   avg = avgHeight();
   mos = mos+0.05f;
   mos2 = 1.0f - mos;
   motor.write(mos);
   motor2.write(mos2);

   t1 = mosTemp(); t2 = mosTemp2();
 }
while(avg>height && mos>= .75f && (t1<=60.0f && t2<=60.0f)){
   avg = avgHeight();
   mos= mos-0.05f;
   mos2 = 1.0f - mos;
   motor.write(mos);
   motor2.write(mos2);

   t1 = mosTemp(); t2 = mosTemp2();

}

if((t1 || t2) > 60.0f){ //if mosfets get to hot
   i = 0;
   while(i<20){ //ramp motors down over a 200ms window
      mos = mos-0.05f;
      mos2 = 1.0f - mos;
      motor.write(mos);
      motor2.write(mos2);
      wait_ms(10);
      i++;
   }
   wait(60); //wait 1 minute to cool off
   t1 = mosTemp(); t2 = mosTemp2(); //check temp again
}

pc.printf("Distance Is \n\r");
printInt(avg);
pc.printf("Duty Cycle is \n\r");
printFloat(mos);
pc.printf("Temperature 1 is \n\r");
```

```
        printFloat(mosTemp());
        pc.printf("Temperature 2 is \n\r");
        printFloat(mosTemp2());

         t1 = mosTemp(); t2 = mosTemp2();
        lcd.locate(0,0);
        lcd.printf("H:%dcm HMax:%dcm",avg,height);
        lcd.locate(0,1);
        lcd.printf("T1:%d C,T2:%d C",t1,t2);

    }

}
```

G. Project Code -Main