The Porter:

A Suitcase Transportation Solution

ME102B Final Project Report

Andrew Zhu, Hubert Liu

Professor Hannah Stuart

Dec. 7, 2022

Opportunities

A potential opportunity we wished to tackle was to prolong the physical health of an individual in physically demanding tasks by relieving the load of heavy items in some way, shape, or form. As a team, we decided to pursue this by focusing on a common issue found in airports. When traveling, people often are faced with many logistics considerations. This includes managing many suitcases that are often large or heavy. For our project, we wished to find a streamlined solution for that.

High-Level Strategy

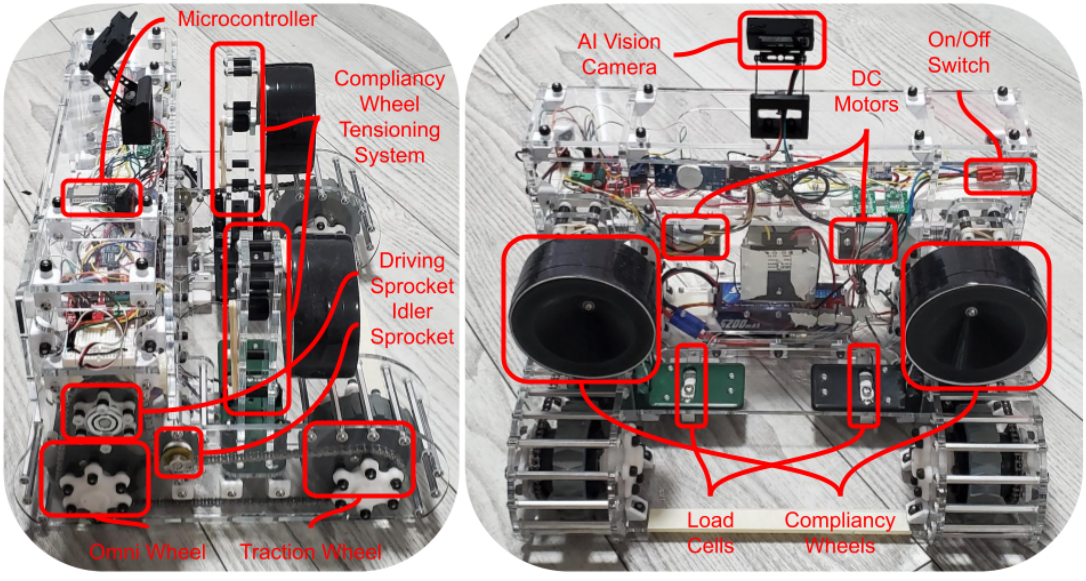
To achieve this goal, we created a wheeled robot that has the ability to hold a carry-on suitcase while following the owner at an average walking pace. As a secondary feature, it can detect the weight of the carry-on suitcase to alert the owner if the suitcase is flight acceptable.

Initially, the idea of holding the suitcase was very simple. The suitcase would simply sit on top of our robot. However, that introduced the issue of creating a low-and-wide profile robot such that the center of gravity wouldn't be too high, which is an issue during abrupt starts and stops. Thus, we pivoted to a design that only required two wheels on the suitcase to be mounted on the robot. In addition to lowering the center of gravity, it also reduced the weight experienced by the robot.

The initial idea for the robot to follow autonomously involved creating a 3D-spatial coordinate system using bluetooth triangulation to track a custom key fob on the user. However, after realizing that commercial bluetooth modules are not fast nor accurate enough, we pivoted to using an AI camera tracking. This solved a couple problems. Initially, we wanted to implement a lidar distance sensor to prevent robot collisions against other obstacles but this consumed a significant amount of time for data processing, which interfered with motor PID control and bluetooth communications. We were able to program the camera to follow a targeted person at a relatively close and maintained distance.

Before beginning our project, we decided that the average walking pace would be around 0.7m/s in an airport. While it could go faster, our robot achieved a comfortable continuous speed of 0.532m/s.

Labeled Diagram of The Porter



Function-Critical Decision & Calculations

Motor Selection: Speed Goals and Torque Calculation- Two 30:1 DC motors were used to actuate a tank drive configuration. Tank drives have no steering capabilities, so omni-wheels were used to eliminate transverse friction loads when performing turns. $v_{goal} = 0.7\,m/s\,(1.5\,mph),\ m = 15\,Kg\,(robot + suitcase),\ equivalent\ output\ wheel\ \# = 2$ (using two chains). $R_{wheel} = 2\,in = 5.08\,cm.$ Assume recommended torque $7.5\,Kg * cm$, $\mu_{rubber\ on\ concrete} = 0.7,\ \mu_{rolling} = 0.0125.$

$$T_{output-actual} = 7.5 - \mu_{rolling} mR = 7.5 - 2.54 = 6.55\,Kg * cm$$

$$F_{output-wheel} = \frac{T_{output-actual}}{R_{wheel}} g = 12.64\,N,\ F_{output-total} = 2F_{output-wheel} = 25.3\,N$$

$$rpm_{rec\ torque} = 150,\ v_{actual} = rpm_{rec\ torque} \times 2\pi R_{wheel} \times \frac{1\,min}{60\,s} \times \frac{1\,m}{100\,cm} = 0.798\,m/s$$ v justified

$$a_{theoretical} = \frac{F_{output-total}}{m} = 25.3/15 = 1.686\,m^2/s,\ t_{reach-v} = \frac{v-v0}{a_{theoretical}} = \frac{0.798}{1.686} = 0.473\,s$$

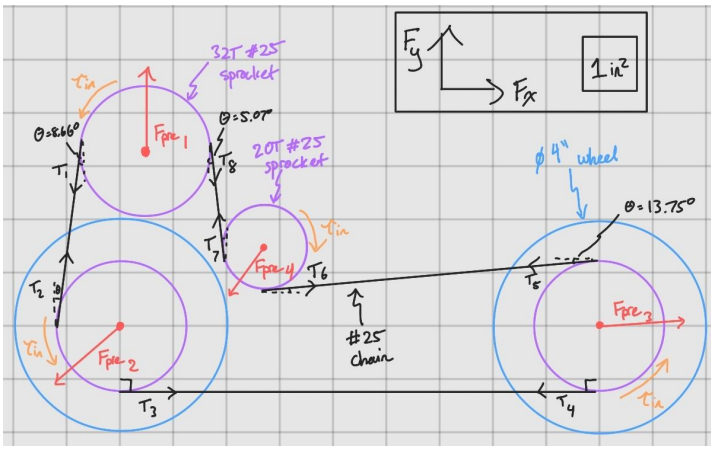
Acceleration results are reasonable (below 0.5s, reasonable threshold), justified for automatic follow and manual control.

If use 19:1 motor, the recommended torque reduces to 5 Kg*cm, which results in $t_{reach-v} = 0.766\,s > 0.5$. If use 50:1 motor, $rpm = 100,\ v_{actual} = 0.532\,m/s < v_{goal}$

Therefore, 30:1 motor was the final choice.

Load Cell Selection: Max Weight Calculation- While max weight varies between airlines, the most common figure is 35 lbs. We made sure to cover this upper bound. Since only two wheels sit on the robot, roughly half the suitcase weight–17.5 lbs–can be registered. With two load cells, each only needs to recognize 8.75 lbs (3.85 kg). In the end, we selected 20kg load cells since our design required them to be structural and were potential points of failure.

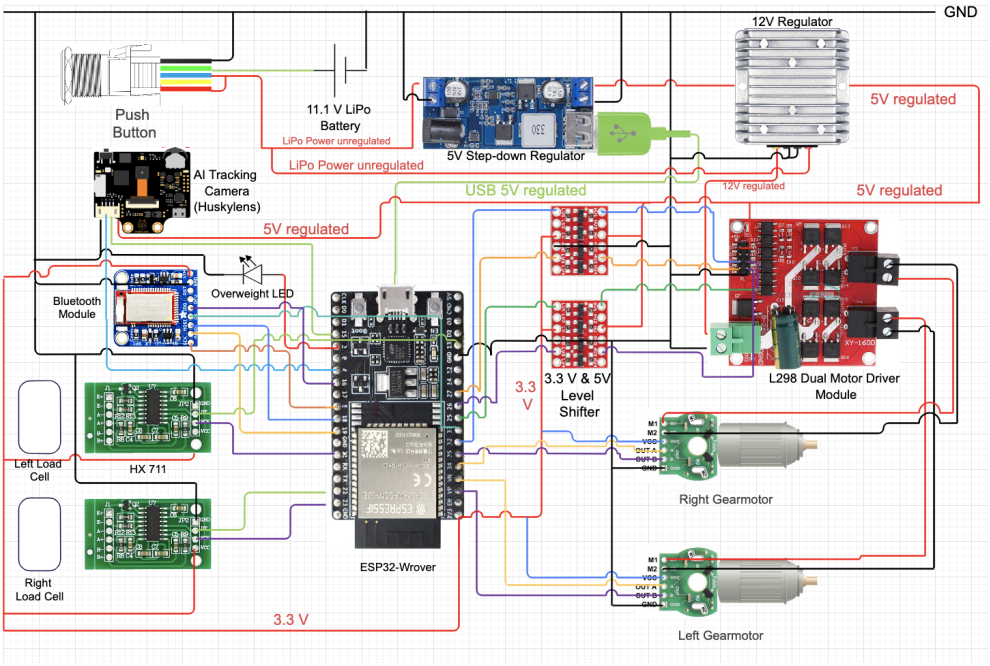
Bearing and Axle Selection: Preloading bearings and determining axle deflection-



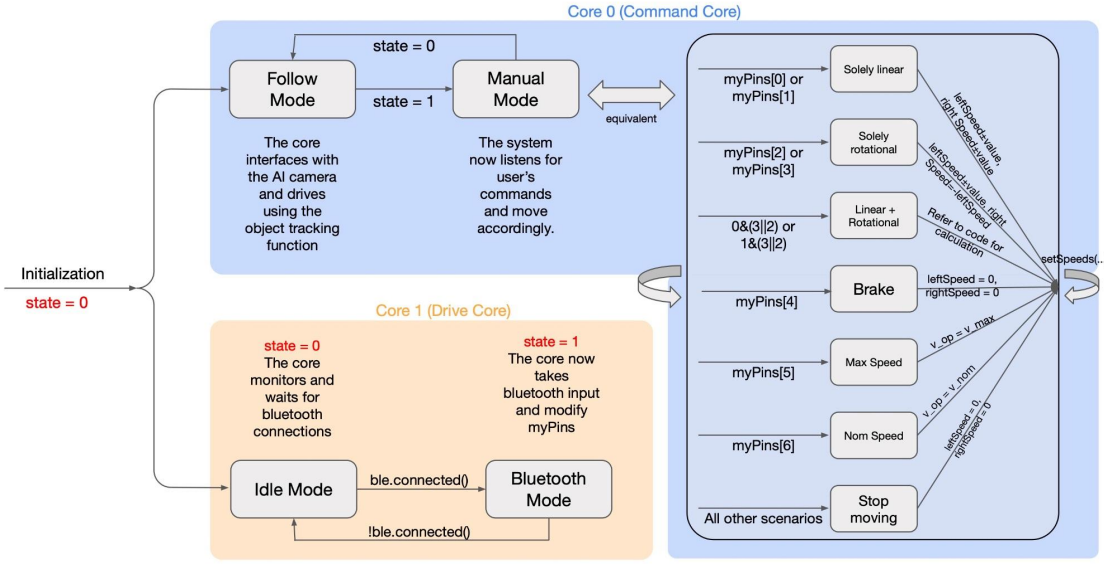
While no axles were cantilevered, it is still important to determine the effects of the radial loads from the robot and suitcase weight: Given $m = 15Kg$, each wheel axle experiences: $F = 15(9.8)/4 = 36.75N$. Checking for deflection for end-fixed beam: $y_{max} = \frac{WL^3}{192EI}$ where W is the load, L is the length, E is the Young's modulus, I is the second area of moment. With $W = 36.75N,\ L = 0.098,\ E = 4e8,\ I = \pi 4.06e\text{-}10$, the max deflection is $3.531e\text{-}4m$ which is practically zero.

Bearing preload equations: $T_1 = T_i - \tau/d \geq 0$, where T_1 is the tension in chain, T_i is the initial force, τ is the torque, and d is the diameter and $F_{Pre} = T_1 + T_2$, where F_{Pre} is the bearing preload. For the driving gear, $T_1 = T_i - \frac{0.74Nm}{0.065m} \geq 0$ and $T_8 = T_i + \frac{0.74Nm}{0.065m} \geq 0.$ T_{i-min} has to be a $11.38N$ to prevent slack. The minimum$F_{Pre1} = 2(11.38)(cos(\frac{8.66+5.07}{2})) = 20.2N.$ Repeating process for the other sprockets $F_{Pre2} = 17.26N,\ F_{Pre3} = 17.89N,\ F_{Pre4} = 17.26N.$

Wiring Diagram



State Diagram



Reflection

A big thing we learned from this project is to be prepared for ideas and designs to change. Sometimes, when something doesn't succeed, it is fine to go back to the drawing board and rethink a goal with a different approach. This cannot happen without frequent and synergistic communication. Being on a team with similar work ethics and goals is what facilitates this communication. In addition, each member should understand their responsibilities. As a two member team, we were able to delegate the tasks between mechanical and electrical work.

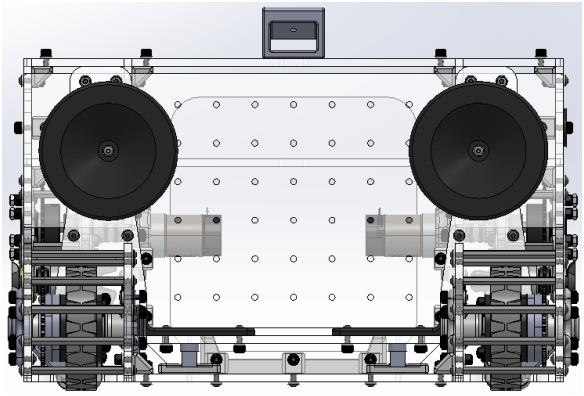
If we were to do one thing differently, it would have been to explore more vendors so that we could have specced out better DC motors and AI cameras. This would translate to a faster and more responsive robot. For future iterations, we could also raise the ground clearance.

APPENDIX

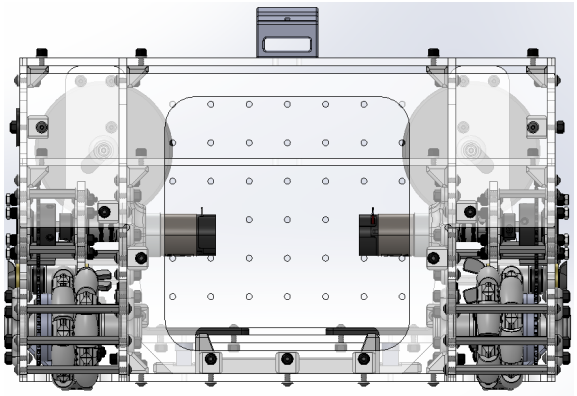
Complete Bill of Materials

ME102B Bill of Materials

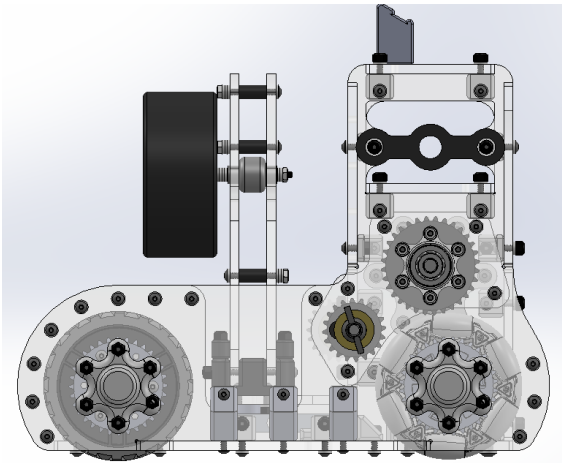
CAD Design (Six Sides)



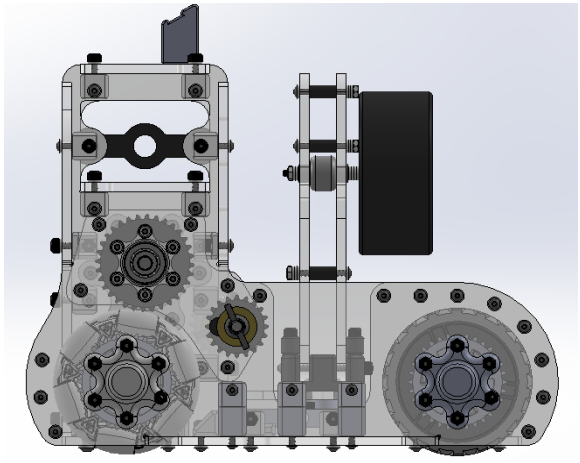
Back View



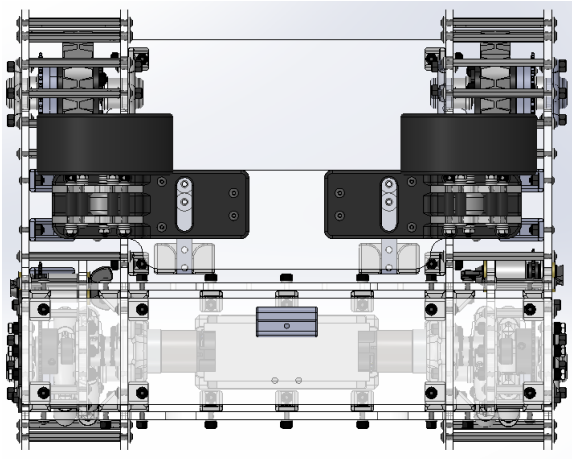
Front View



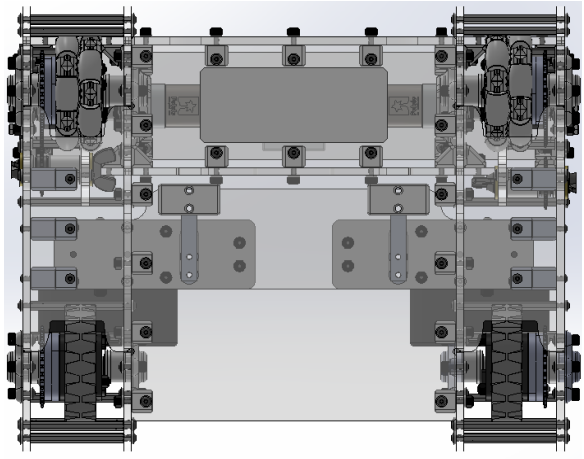
Right View



Left View

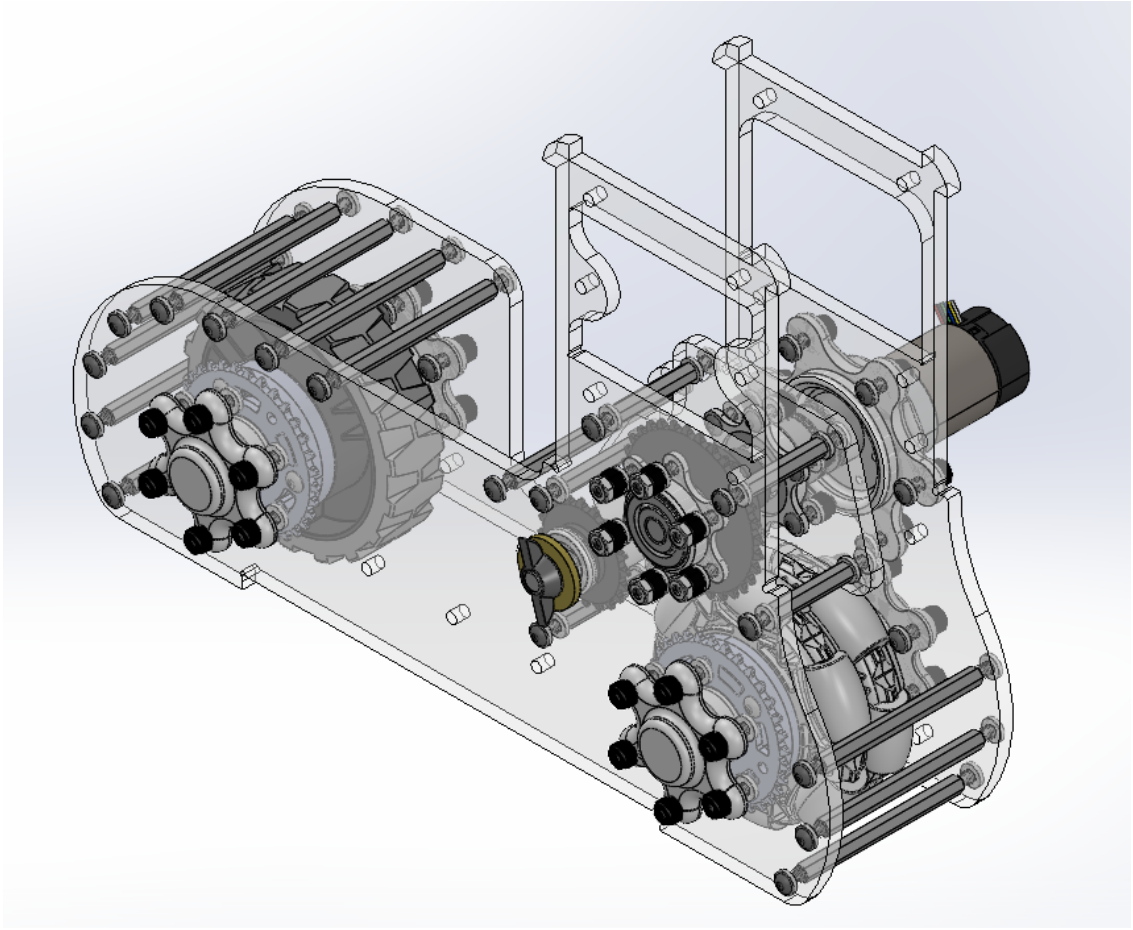


Top View

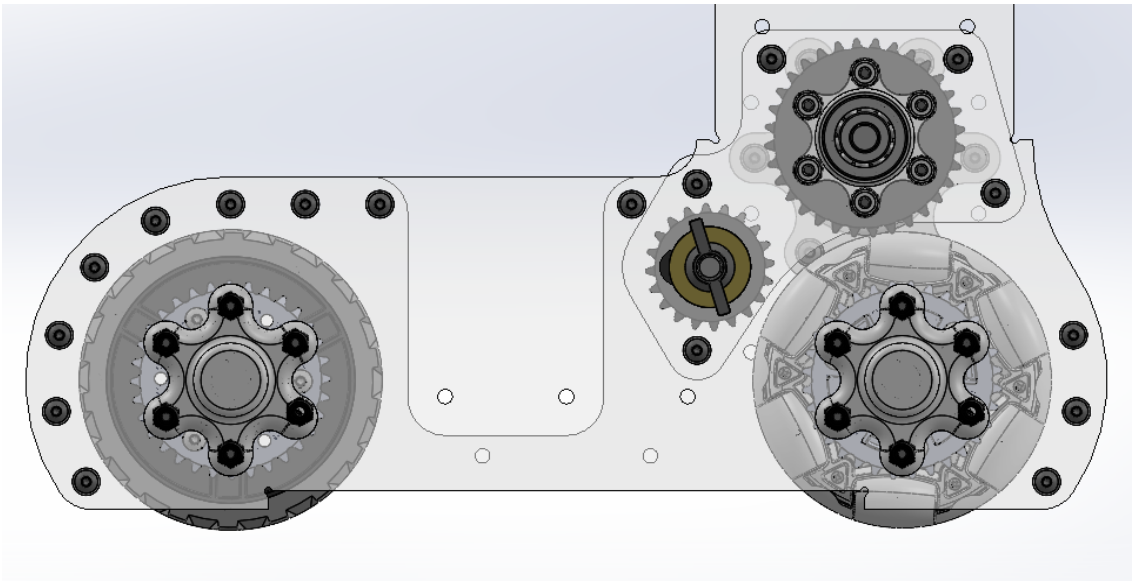


Bottom View

CAD Design (Transmission)

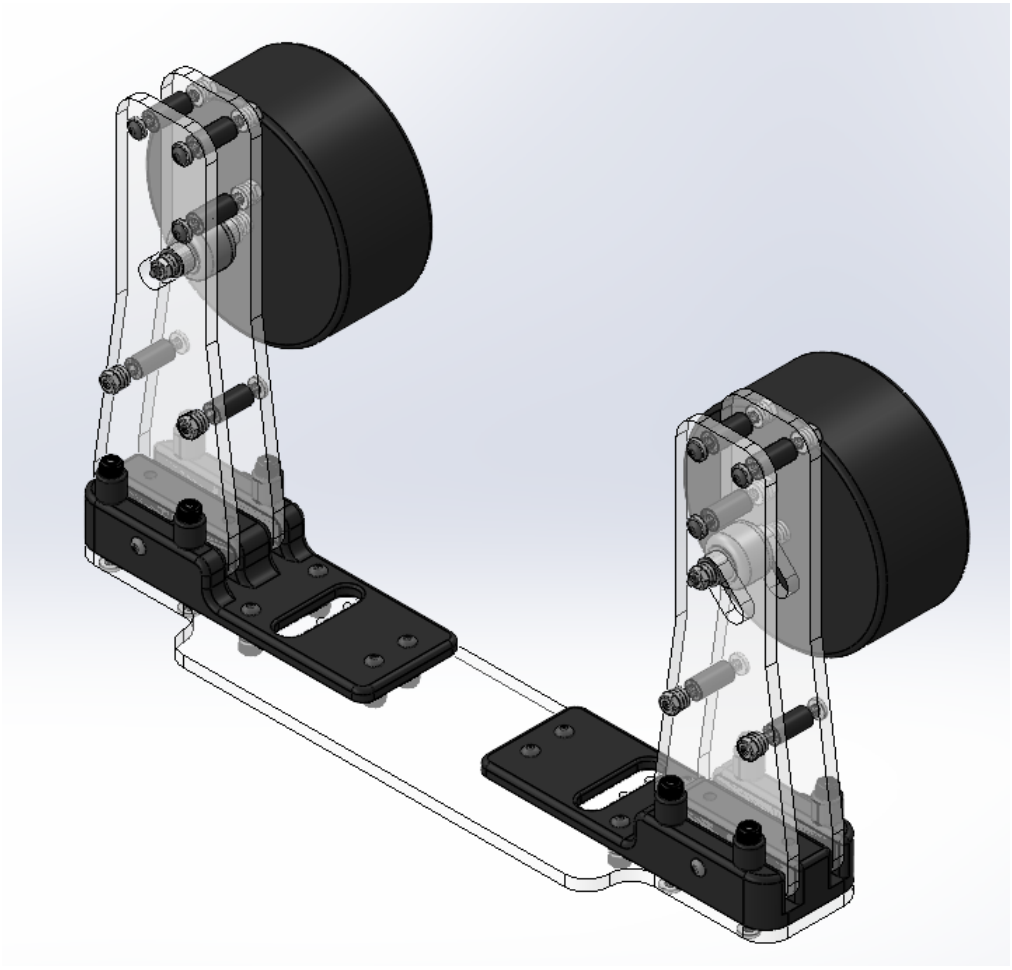


Isometric View

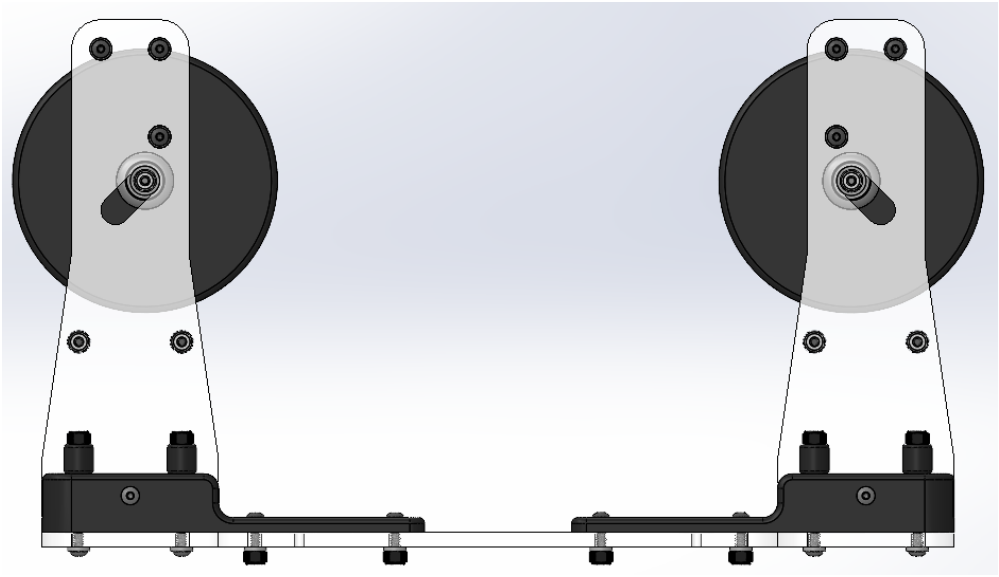


Side View

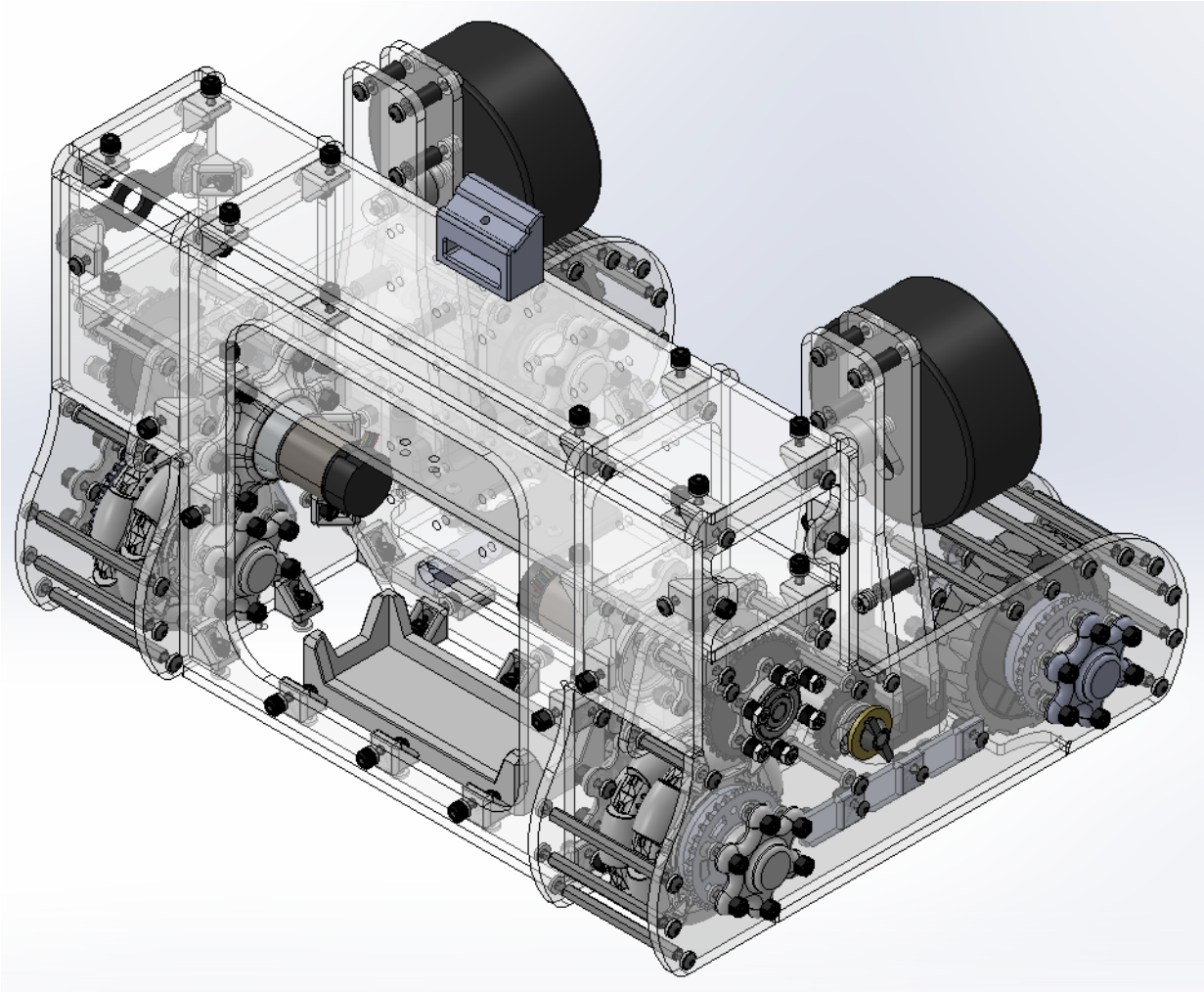
CAD Design (Suitcase Interface System)



Isometric View



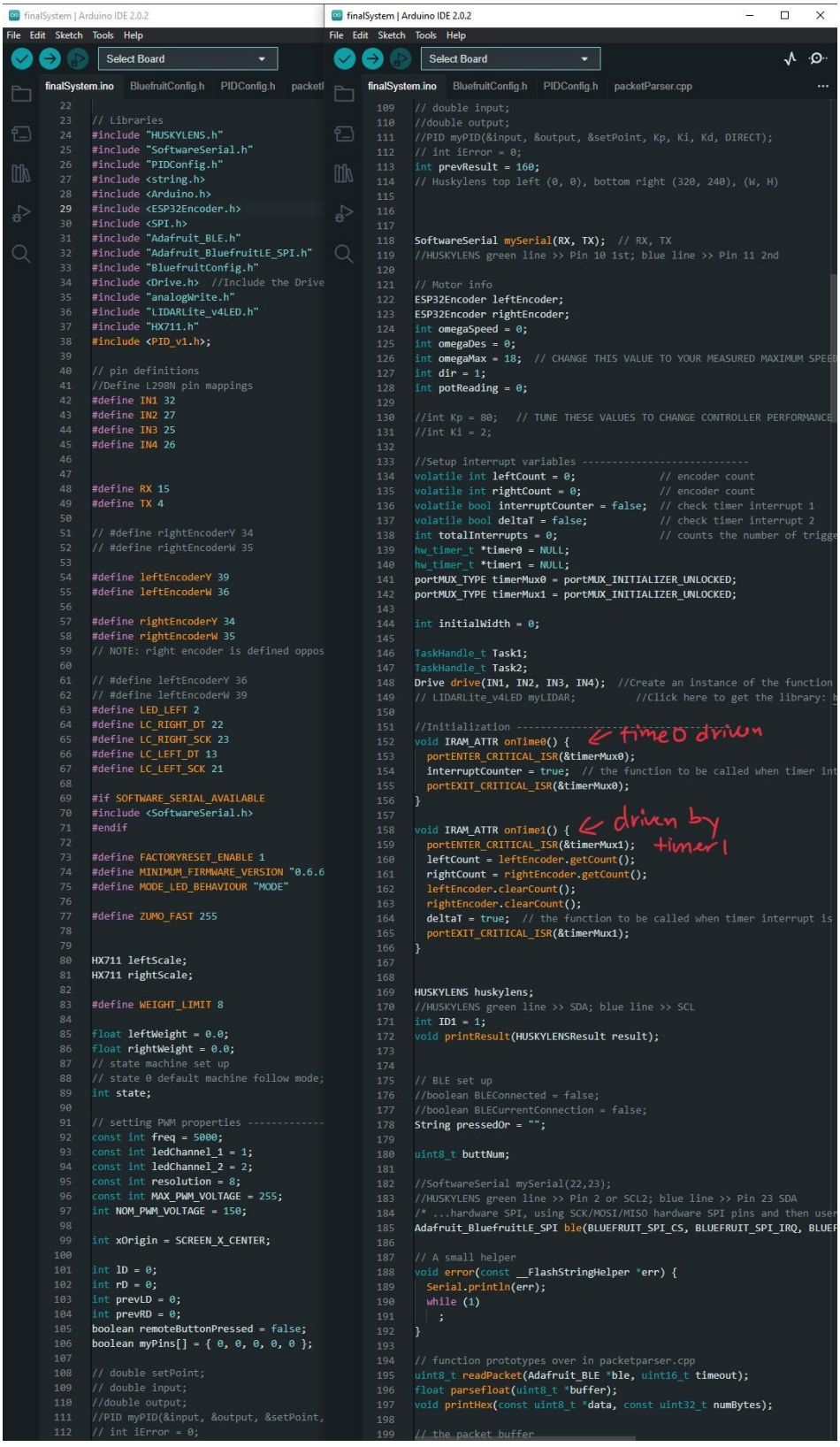
Front View
CAD Design (Complete Overview)



Isometric View

Comments: A big part of designing is planning for how the components will be manufactured and assembled together. Sometimes, parts that are designed in CAD may not work as expected. For example, when designing in CAD, it is important not to neglect the potential effects of manufacturing defects or tolerances. There were many redesigns before we arrived at the current design. Our team spent a lot of time finalizing the design in SolidWorks before going into full scale build mode. At some point, the CAD model became so detailed that we ended up using it as our build manual. For us, putting in the hard work early saved a lot of time and effort in the end.

CODE
Final System (main code) File



finalSystem | Arduino IDE 2.0.2

File Edit Sketch Tools Help

Select Board

finalSystem.ino BluefruitConfig.h PIDConfig.h packetI

```
22
23  // Libraries
24  #include "HUSKYLENS.h"
25  #include "SoftwareSerial.h"
26  #include "PIDConfig.h"
27  #include <string.h>
28  #include <Arduino.h>
29  #include <ESP32Encoder.h>
30  #include <SPI.h>
31  #include "Adafruit_BLE.h"
32  #include "Adafruit_BluefruitLE_SPI.h"
33  #include "BluefruitConfig.h"
34  #include <Drive.h>  //Include the Drive
35  #include "analogWrite.h"
36  #include "LIDARLite_v4LED.h"
37  #include "HX711.h"
38  #include <PID_v1.h>;
39
40  // pin definitions
41  //Define L298N pin mappings
42  #define IN1 32
43  #define IN2 27
44  #define IN3 25
45  #define IN4 26
46
47
48  #define RX 15
49  #define TX 4
50
51  // #define rightEncoderY 34
52  // #define rightEncoderW 35
53
54  #define leftEncoderY 39
55  #define leftEncoderW 36
56
57  #define rightEncoderY 34
58  #define rightEncoderW 35
59  // NOTE: right encoder is defined oppos
60
61  // #define leftEncoderY 36
62  // #define leftEncoderW 39
63  #define LED_LEFT 2
64  #define LC_RIGHT_DT 22
65  #define LC_RIGHT_SCK 23
66  #define LC_LEFT_DT 13
67  #define LC_LEFT_SCK 21
68
69  #if SOFTWARE_SERIAL_AVAILABLE
70  #include <SoftwareSerial.h>
71  #endif
72
73  #define FACTORYRESET_ENABLE 1
74  #define MINIMUM_FIRMWARE_VERSION "0.6.6
75  #define MODE_LED_BEHAVIOUR "MODE"
76
77  #define ZUMO_FAST 255
78
79
80  HX711 leftScale;
81  HX711 rightScale;
82
83  #define WEIGHT_LIMIT 8
84
85  float leftWeight = 0.0;
86  float rightWeight = 0.0;
87  // state machine set up
88  // state 0 default machine follow mode;
89  int state;
90
91  // setting PWM properties ------------
92  const int freq = 5000;
93  const int ledChannel_1 = 1;
94  const int ledChannel_2 = 2;
95  const int resolution = 8;
96  const int MAX_PWM_VOLTAGE = 255;
97  int NOM_PWM_VOLTAGE = 150;
98
99  int xOrigin = SCREEN_X_CENTER;
100
101 int lD = 0;
102 int rD = 0;
103 int prevLD = 0;
104 int prevRD = 0;
105 boolean remoteButtonPressed = false;
106 boolean myPins[] = { 0, 0, 0, 0, 0 };
107
108 // double setPoint;
109 // double input;
110 //double output;
111 //PID myPID(&input, &output, &setPoint,
112 // int iError = 0;
```

finalSystem | Arduino IDE 2.0.2

File Edit Sketch Tools Help

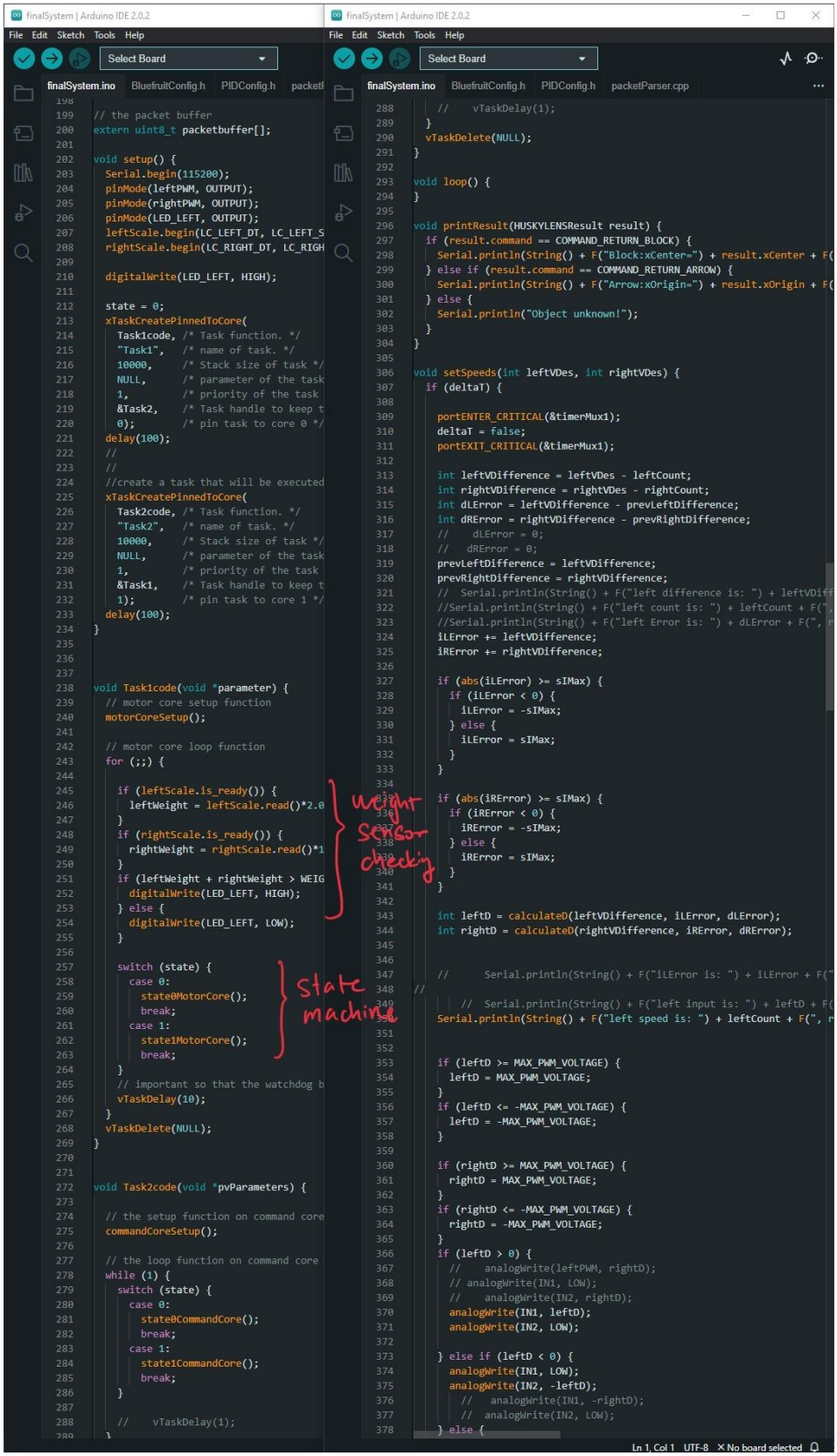
Select Board

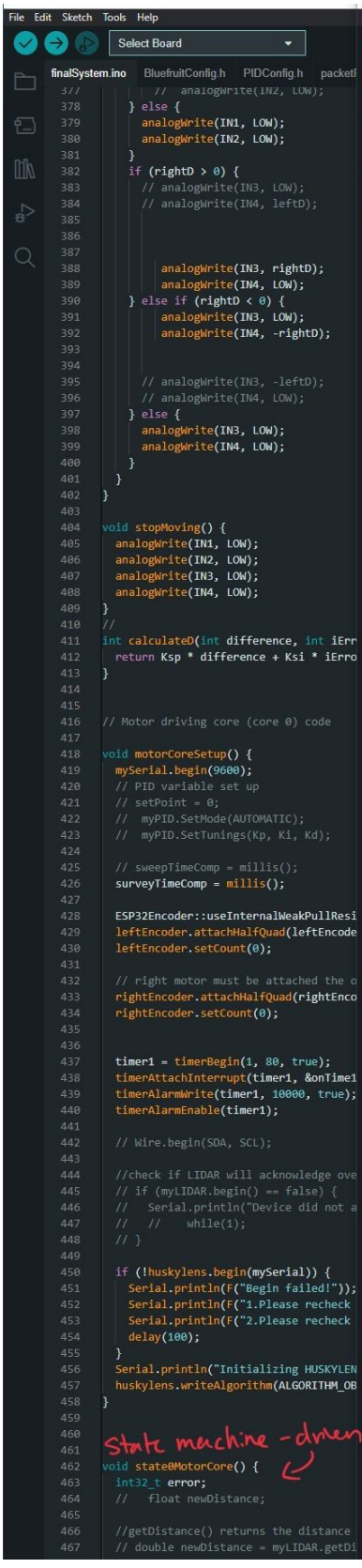
finalSystem.ino BluefruitConfig.h PIDConfig.h packetParser.cpp

```
109 // double input;
110 //double output;
111 //PID myPID(&input, &output, &setPoint, Kp, Ki, Kd, DIRECT);
112 // int iError = 0;
113 int prevResult = 160;
114 // Huskylens top left (0, 0), bottom right (320, 240), (W, H)
115
116
117
118 SoftwareSerial mySerial(RX, TX);  // RX, TX
119 //HUSKYLENS green line >> Pin 10 1st; blue line >> Pin 11 2nd
120
121 // Motor info
122 ESP32Encoder leftEncoder;
123 ESP32Encoder rightEncoder;
124 int omegaSpeed = 0;
125 int omegaDes = 0;
126 int omegaMax = 18;  // CHANGE THIS VALUE TO YOUR MEASURED MAXIMUM SPEED
127 int dir = 1;
128 int potReading = 0;
129
130 //int Kp = 80;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
131 //int Ki = 2;
132
133 //Setup interrupt variables ---------------------------
134 volatile int leftCount = 0;              // encoder count
135 volatile int rightCount = 0;             // encoder count
136 volatile bool interruptCounter = false;  // check timer interrupt 1
137 volatile bool deltaT = false;            // check timer interrupt 2
138 int totalInterrupts = 0;                 // counts the number of trigge
139 hw_timer_t *timer0 = NULL;
140 hw_timer_t *timer1 = NULL;
141 portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
142 portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
143
144 int initialWidth = 0;
145
146 TaskHandle_t Task1;
147 TaskHandle_t Task2;
148 Drive drive(IN1, IN2, IN3, IN4);  //Create an instance of the function
149 // LIDARLite_v4LED myLIDAR;          //Click here to get the library: h
150
151 //Initialization ---------------------------------------
152 void IRAM_ATTR onTime0() {
153   portENTER_CRITICAL_ISR(&timerMux0);
154   interruptCounter = true;  // the function to be called when timer int
155   portEXIT_CRITICAL_ISR(&timerMux0);
156 }
157
158 void IRAM_ATTR onTime1() {
159   portENTER_CRITICAL_ISR(&timerMux1);
160   leftCount = leftEncoder.getCount();
161   rightCount = rightEncoder.getCount();
162   leftEncoder.clearCount();
163   rightEncoder.clearCount();
164   deltaT = true;  // the function to be called when timer interrupt is
165   portEXIT_CRITICAL_ISR(&timerMux1);
166 }
167
168
169 HUSKYLENS huskylens;
170 //HUSKYLENS green line >> SDA; blue line >> SCL
171 int ID1 = 1;
172 void printResult(HUSKYLENSResult result);
173
174
175 // BLE set up
176 //boolean BLEConnected = false;
177 //boolean BLECurrentConnection = false;
178 String pressedOr = "";
179
180 uint8_t buttNum;
181
182 //SoftwareSerial mySerial(22,23);
183 //HUSKYLENS green line >> Pin 2 or SCL2; blue line >> Pin 23 SDA
184 /* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user
185 Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEF
186
187 // A small helper
188 void error(const __FlashStringHelper *err) {
189   Serial.println(err);
190   while (1)
191     ;
192 }
193
194 // function prototypes over in packetparser.cpp
195 uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
196 float parsefloat(uint8_t *buffer);
197 void printHex(const uint8_t *data, const uint32_t numBytes);
198
199 // the packet buffer
```

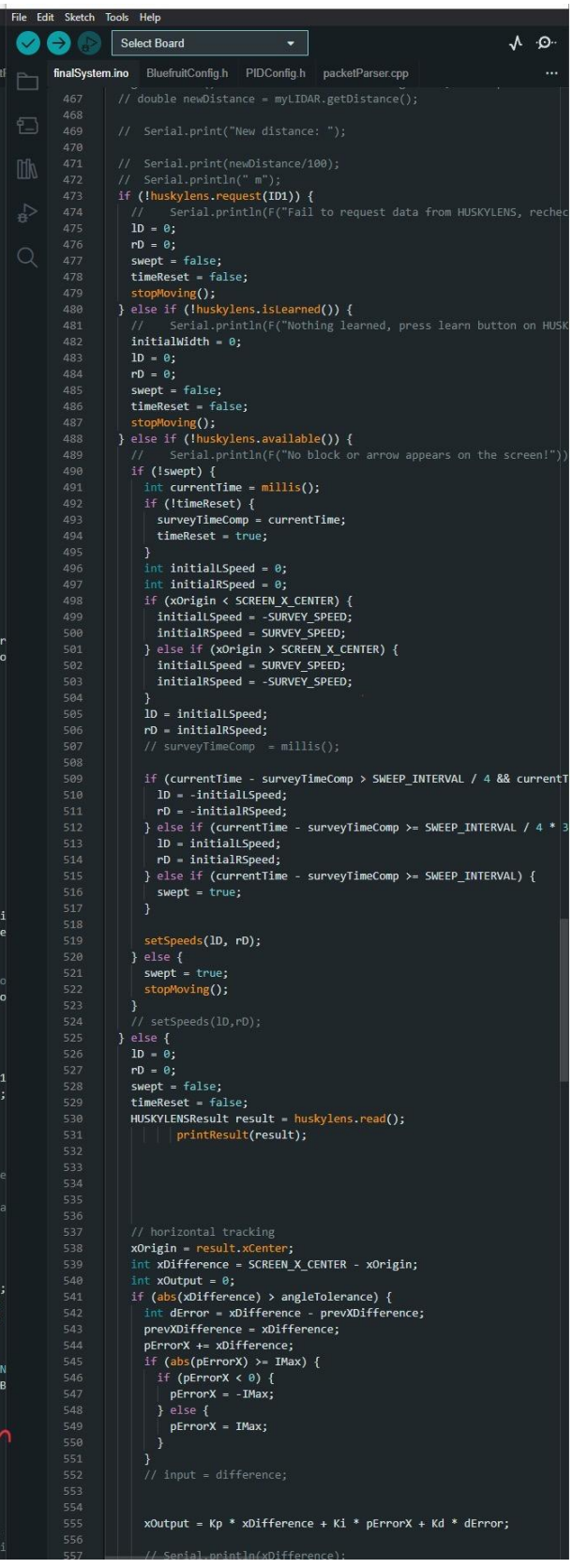
timer0 driven

driven by timer1

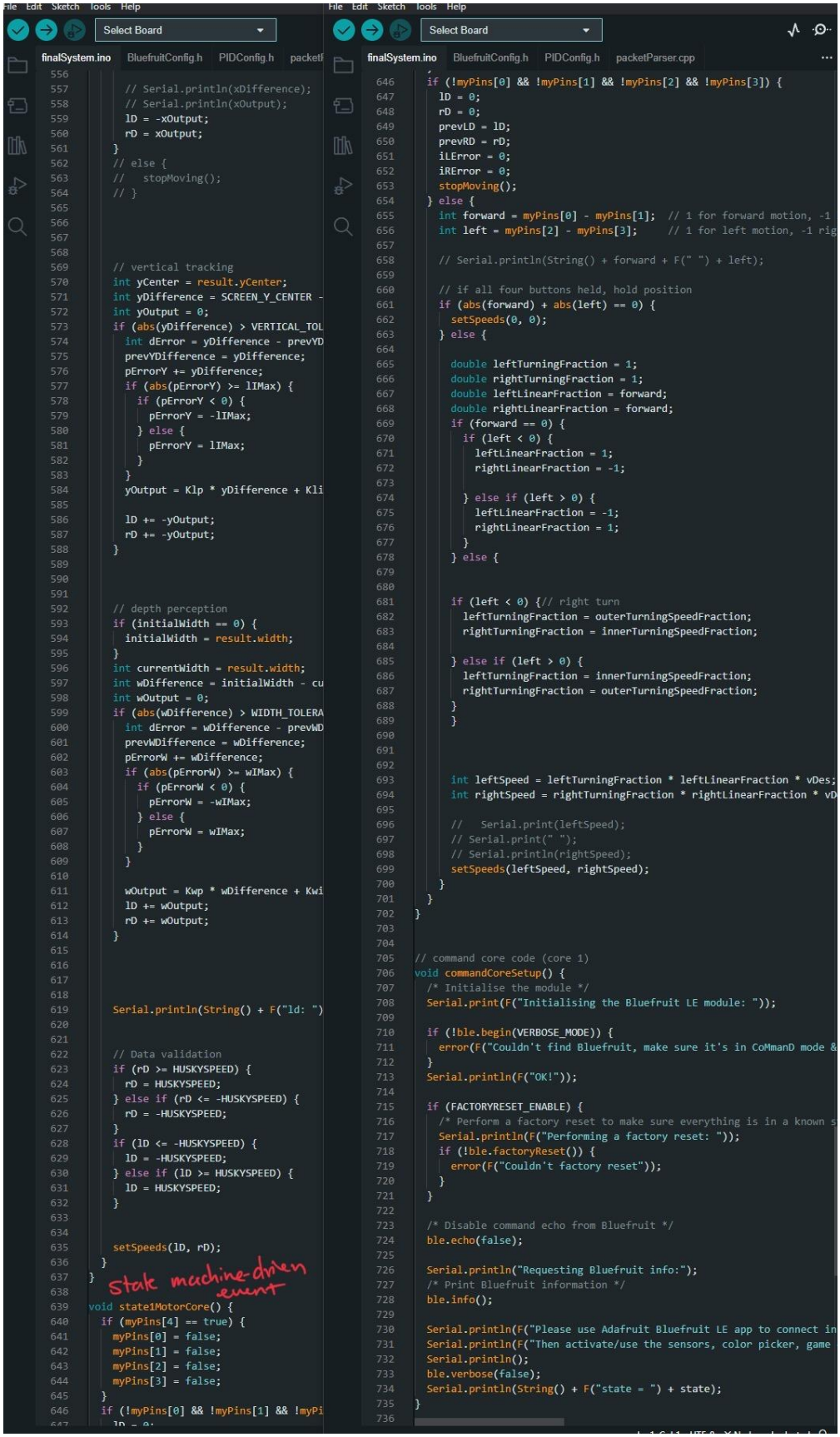

finalSystem | Arduino IDE 2.0.2
File Edit Sketch Tools Help
Select Board
finalSystem.ino
BluefruitConfig.h
PIDConfig.h
packetParser.cpp
// the packet buffer
extern uint8_t packetbuffer[];
void setup() {
Serial.begin(115200);
pinMode(leftPWM, OUTPUT);
pinMode(rightPWM, OUTPUT);
pinMode(LED_LEFT, OUTPUT);
digitalWrite(LED_LEFT, HIGH);
state = 0;
delay(100);
void Task1code(void *parameter) {
motorCoreSetup();
for (;;) {
if (leftScale.is_ready()) {
leftWeight = leftScale.read()*2.0
}
if (rightScale.is_ready()) {
rightWeight = rightScale.read()*1
}
if (leftWeight + rightWeight > WEIG
digitalWrite(LED_LEFT, HIGH);
} else {
digitalWrite(LED_LEFT, LOW);
}
switch (state) {
case 0:
state0MotorCore();
break;
case 1:
state1MotorCore();
break;
}
vTaskDelay(10);
}
vTaskDelete(NULL);
}
void Task2code(void *pvParameters) {
commandCoreSetup();
while (1) {
switch (state) {
case 0:
state0CommandCore();
break;
case 1:
state1CommandCore();
break;
}
vTaskDelay(1);
}
vTaskDelete(NULL);
}
void loop() {
}
void printResult(HUSKYLENSResult result) {
if (result.command == COMMAND_RETURN_BLOCK) {
Serial.println(String() + F("Block:xCenter=") + result.xCenter + F(
} else if (result.command == COMMAND_RETURN_ARROW) {
Serial.println(String() + F("Arrow:xOrigin=") + result.xOrigin + F(
} else {
Serial.println("Object unknown!");
}
}
void setSpeeds(int leftVDes, int rightVDes) {
if (deltaT) {
portENTER_CRITICAL(&timerMux1);
deltaT = false;
portEXIT_CRITICAL(&timerMux1);
int leftVDifference = leftVDes - leftCount;
int rightVDifference = rightVDes - rightCount;
int dLError = leftVDifference - prevLeftDifference;
int dRError = rightVDifference - prevRightDifference;
prevLeftDifference = leftVDifference;
prevRightDifference = rightVDifference;
iLError += leftVDifference;
iRError += rightVDifference;
if (abs(iLError) >= sIMax) {
if (iLError < 0) {
iLError = -sIMax;
} else {
iLError = sIMax;
}
}
if (abs(iRError) >= sIMax) {
if (iRError < 0) {
iRError = -sIMax;
} else {
iRError = sIMax;
}
}
int leftD = calculateD(leftVDifference, iLError, dLError);
int rightD = calculateD(rightVDifference, iRError, dRError);
Serial.println(String() + F("left speed is: ") + leftCount + F(",
if (leftD >= MAX_PWM_VOLTAGE) {
leftD = MAX_PWM_VOLTAGE;
}
if (leftD <= -MAX_PWM_VOLTAGE) {
leftD = -MAX_PWM_VOLTAGE;
}
if (rightD >= MAX_PWM_VOLTAGE) {
rightD = MAX_PWM_VOLTAGE;
}
if (rightD <= -MAX_PWM_VOLTAGE) {
rightD = -MAX_PWM_VOLTAGE;
}
if (leftD > 0) {
analogWrite(IN1, leftD);
analogWrite(IN2, LOW);
} else if (leftD < 0) {
analogWrite(IN1, LOW);
analogWrite(IN2, -leftD);
} else {
Ln 1, Col 1 UTF-8 No board selected
weight sensor checking
state machine



```
File Edit Sketch Tools Help
Select Board
finalSystem.ino  BluefruitConfig.h  PIDConfig.h  packetf
377          //  analogWrite(IN2, LOW);
378        } else {
379          analogWrite(IN1, LOW);
380          analogWrite(IN2, LOW);
381        }
382        if (rightD > 0) {
383          // analogWrite(IN3, LOW);
384          // analogWrite(IN4, leftD);
385
386
387
388            analogWrite(IN3, rightD);
389            analogWrite(IN4, LOW);
390        } else if (rightD < 0) {
391            analogWrite(IN3, LOW);
392            analogWrite(IN4, -rightD);
393
394
395          // analogWrite(IN3, -leftD);
396          // analogWrite(IN4, LOW);
397        } else {
398          analogWrite(IN3, LOW);
399          analogWrite(IN4, LOW);
400        }
401      }
402    }
403
404    void stopMoving() {
405      analogWrite(IN1, LOW);
406      analogWrite(IN2, LOW);
407      analogWrite(IN3, LOW);
408      analogWrite(IN4, LOW);
409    }
410    //
411    int calculateD(int difference, int iErr
412      return Ksp * difference + Ksi * iErro
413    }
414
415
416    // Motor driving core (core 0) code
417
418    void motorCoreSetup() {
419      mySerial.begin(9600);
420      // PID variable set up
421      // setPoint = 0;
422      //  myPID.SetMode(AUTOMATIC);
423      //  myPID.SetTunings(Kp, Ki, Kd);
424
425      // sweepTimeComp = millis();
426      surveyTimeComp = millis();
427
428      ESP32Encoder::useInternalWeakPullResi
429      leftEncoder.attachHalfQuad(leftEncode
430      leftEncoder.setCount(0);
431
432      // right motor must be attached the o
433      rightEncoder.attachHalfQuad(rightEnco
434      rightEncoder.setCount(0);
435
436
437      timer1 = timerBegin(1, 80, true);
438      timerAttachInterrupt(timer1, &onTime1
439      timerAlarmWrite(timer1, 10000, true);
440      timerAlarmEnable(timer1);
441
442      // Wire.begin(SDA, SCL);
443
444      //check if LIDAR will acknowledge ove
445      // if (myLIDAR.begin() == false) {
446      //   Serial.println("Device did not a
447      //   //    while(1);
448      // }
449
450      if (!huskylens.begin(mySerial)) {
451        Serial.println(F("Begin failed!"));
452        Serial.println(F("1.Please recheck
453        Serial.println(F("2.Please recheck
454        delay(100);
455      }
456      Serial.println("Initializing HUSKYLEN
457      huskylens.writeAlgorithm(ALGORITHM_OB
458    }
459
460
461    state machine - driven
462    void state0MotorCore() {
463      int32_t error;
464      //   float newDistance;
465
466      //getDistance() returns the distance
467      // double newDistance = myLIDAR.getDi
```



```
File Edit Sketch Tools Help
Select Board
finalSystem.ino  BluefruitConfig.h  PIDConfig.h  packetParser.cpp
467    // double newDistance = myLIDAR.getDistance();
468
469    //  Serial.print("New distance: ");
470
471    //  Serial.print(newDistance/100);
472    //  Serial.println(" m");
473    if (!huskylens.request(ID1)) {
474      //    Serial.println(F("Fail to request data from HUSKYLENS, rechec
475      lD = 0;
476      rD = 0;
477      swept = false;
478      timeReset = false;
479      stopMoving();
480    } else if (!huskylens.isLearned()) {
481      //    Serial.println(F("Nothing learned, press learn button on HUSK
482      initialWidth = 0;
483      lD = 0;
484      rD = 0;
485      swept = false;
486      timeReset = false;
487      stopMoving();
488    } else if (!huskylens.available()) {
489      //    Serial.println(F("No block or arrow appears on the screen!"))
490      if (!swept) {
491        int currentTime = millis();
492        if (!timeReset) {
493          surveyTimeComp = currentTime;
494          timeReset = true;
495        }
496        int initialLSpeed = 0;
497        int initialRSpeed = 0;
498        if (xOrigin < SCREEN_X_CENTER) {
499          initialLSpeed = -SURVEY_SPEED;
500          initialRSpeed = SURVEY_SPEED;
501        } else if (xOrigin > SCREEN_X_CENTER) {
502          initialLSpeed = SURVEY_SPEED;
503          initialRSpeed = -SURVEY_SPEED;
504        }
505        lD = initialLSpeed;
506        rD = initialRSpeed;
507        // surveyTimeComp  = millis();
508
509        if (currentTime - surveyTimeComp > SWEEP_INTERVAL / 4 && currentT
510          lD = -initialLSpeed;
511          rD = -initialRSpeed;
512        } else if (currentTime - surveyTimeComp >= SWEEP_INTERVAL / 4 * 3
513          lD = initialLSpeed;
514          rD = initialRSpeed;
515        } else if (currentTime - surveyTimeComp >= SWEEP_INTERVAL) {
516          swept = true;
517        }
518
519        setSpeeds(lD, rD);
520      } else {
521        swept = true;
522        stopMoving();
523      }
524      // setSpeeds(lD,rD);
525    } else {
526      lD = 0;
527      rD = 0;
528      swept = false;
529      timeReset = false;
530      HUSKYLENSResult result = huskylens.read();
531            printResult(result);
532
533
534
535
536
537    // horizontal tracking
538    xOrigin = result.xCenter;
539    int xDifference = SCREEN_X_CENTER - xOrigin;
540    int xOutput = 0;
541    if (abs(xDifference) > angleTolerance) {
542      int dError = xDifference - prevXDifference;
543      prevXDifference = xDifference;
544      pErrorX += xDifference;
545      if (abs(pErrorX) >= IMax) {
546        if (pErrorX < 0) {
547          pErrorX = -IMax;
548        } else {
549          pErrorX = IMax;
550        }
551      }
552      // input = difference;
553
554
555      xOutput = Kp * xDifference + Ki * pErrorX + Kd * dError;
556
557      // Serial.println(xDifference);
```



File Edit Sketch Tools Help

Select Board

finalSystem.ino | BluefruitConfig.h | PIDConfig.h | packetParser.cpp

```
556
557       // Serial.println(xDifference);
558       // Serial.println(xOutput);
559       lD = -xOutput;
560       rD = xOutput;
561     }
562     // else {
563     //   stopMoving();
564     // }
565
566
567
568
569     // vertical tracking
570     int yCenter = result.yCenter;
571     int yDifference = SCREEN_Y_CENTER -
572     int yOutput = 0;
573     if (abs(yDifference) > VERTICAL_TOL
574       int dError = yDifference - prevYD
575       prevYDifference = yDifference;
576       pErrorY += yDifference;
577       if (abs(pErrorY) >= lIMax) {
578         if (pErrorY < 0) {
579           pErrorY = -lIMax;
580         } else {
581           pErrorY = lIMax;
582         }
583       }
584       yOutput = Klp * yDifference + Kli
585
586       lD += -yOutput;
587       rD += -yOutput;
588     }
589
590
591
592     // depth perception
593     if (initialWidth == 0) {
594       initialWidth = result.width;
595     }
596     int currentWidth = result.width;
597     int wDifference = initialWidth - cu
598     int wOutput = 0;
599     if (abs(wDifference) > WIDTH_TOLERA
600       int dError = wDifference - prevWD
601       prevWDifference = wDifference;
602       pErrorW += wDifference;
603       if (abs(pErrorW) >= wIMax) {
604         if (pErrorW < 0) {
605           pErrorW = -wIMax;
606         } else {
607           pErrorW = wIMax;
608         }
609       }
610
611       wOutput = Kwp * wDifference + Kwi
612       lD += wOutput;
613       rD += wOutput;
614     }
615
616
617
618
619     Serial.println(String() + F("ld: ")
620
621
622     // Data validation
623     if (rD >= HUSKYSPEED) {
624       rD = HUSKYSPEED;
625     } else if (rD <= -HUSKYSPEED) {
626       rD = -HUSKYSPEED;
627     }
628     if (lD <= -HUSKYSPEED) {
629       lD = -HUSKYSPEED;
630     } else if (lD >= HUSKYSPEED) {
631       lD = HUSKYSPEED;
632     }
633
634
635     setSpeeds(lD, rD);
636   }
637 }
638
639 void state1MotorCore() {
640   if (myPins[4] == true) {
641     myPins[0] = false;
642     myPins[1] = false;
643     myPins[2] = false;
644     myPins[3] = false;
645   }
646   if (!myPins[0] && !myPins[1] && !myPi
647     lD = 0;
```

Stak machine-driven event

```
646   if (!myPins[0] && !myPins[1] && !myPins[2] && !myPins[3]) {
647     lD = 0;
648     rD = 0;
649     prevLD = lD;
650     prevRD = rD;
651     iLError = 0;
652     iRError = 0;
653     stopMoving();
654   } else {
655     int forward = myPins[0] - myPins[1];  // 1 for forward motion, -1
656     int left = myPins[2] - myPins[3];     // 1 for left motion, -1 rig
657
658     // Serial.println(String() + forward + F(" ") + left);
659
660     // if all four buttons held, hold position
661     if (abs(forward) + abs(left) == 0) {
662       setSpeeds(0, 0);
663     } else {
664
665       double leftTurningFraction = 1;
666       double rightTurningFraction = 1;
667       double leftLinearFraction = forward;
668       double rightLinearFraction = forward;
669       if (forward == 0) {
670         if (left < 0) {
671           leftLinearFraction = 1;
672           rightLinearFraction = -1;
673
674         } else if (left > 0) {
675           leftLinearFraction = -1;
676           rightLinearFraction = 1;
677         }
678       } else {
679
680
681       if (left < 0) {// right turn
682         leftTurningFraction = outerTurningSpeedFraction;
683         rightTurningFraction = innerTurningSpeedFraction;
684
685       } else if (left > 0) {
686         leftTurningFraction = innerTurningSpeedFraction;
687         rightTurningFraction = outerTurningSpeedFraction;
688       }
689       }
690
691
692
693       int leftSpeed = leftTurningFraction * leftLinearFraction * vDes;
694       int rightSpeed = rightTurningFraction * rightLinearFraction * vD
695
696       //   Serial.print(leftSpeed);
697       // Serial.print(" ");
698       // Serial.println(rightSpeed);
699       setSpeeds(leftSpeed, rightSpeed);
700     }
701   }
702 }
703
704
705 // command core code (core 1)
706 void commandCoreSetup() {
707   /* Initialise the module */
708   Serial.print(F("Initialising the Bluefruit LE module: "));
709
710   if (!ble.begin(VERBOSE_MODE)) {
711     error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &
712   }
713   Serial.println(F("OK!"));
714
715   if (FACTORYRESET_ENABLE) {
716     /* Perform a factory reset to make sure everything is in a known s
717     Serial.println(F("Performing a factory reset: "));
718     if (!ble.factoryReset()) {
719       error(F("Couldn't factory reset"));
720     }
721   }
722
723   /* Disable command echo from Bluefruit */
724   ble.echo(false);
725
726   Serial.println("Requesting Bluefruit info:");
727   /* Print Bluefruit information */
728   ble.info();
729
730   Serial.println(F("Please use Adafruit Bluefruit LE app to connect in
731   Serial.println(F("Then activate/use the sensors, color picker, game
732   Serial.println();
733   ble.verbose(false);
734   Serial.println(String() + F("state = ") + state);
735 }
736
```


```
737  void state0CommandCore() {
738    // event checker
739    if (ble.isConnected()) {
740      state = 1;
741      Serial.println(String() + F("state = ") + state);
742      Serial.println(F("*****************************"));
743
744      // LED Activity command is only supported from 0.6.6
745      if (ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION)) {
746        // Change Mode LED Activity
747        Serial.println(F("Change LED activity to " MODE_LED
748        ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEH
749      }
750
751      // Set Bluefruit to DATA mode
752      Serial.println(F("Switching to DATA mode!"));
753      ble.setMode(BLUEFRUIT_MODE_DATA);
754
755      Serial.println(F("*****************************"));
756    }
757  }
758
759  void state1CommandCore() {
760
761    // event checker
762    if (ble.isConnected()) {
763
764      // services
765      uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT
766      if (len == 0) return;
767      // Buttons
768      if (packetbuffer[1] == 'B') {
769        uint8_t buttnum = packetbuffer[2] - '0';
770        buttNum = buttnum;  // 5 forward, 6 backward, 7 lef
771        boolean pressed = packetbuffer[3] - '0';
772        Serial.print("Button ");
773        Serial.print(buttnum);
774        if (pressed) {
775          pressedOr = " pressed";
776          switch (buttnum) {
777            case 5:
778              myPins[0] = true;
779              break;
780            case 6:
781              myPins[1] = true;
782              break;
783            case 7:
784              myPins[2] = true;
785              break;
786            case 8:
787              myPins[3] = true;
788              break;
789            case 1:
790              myPins[4] = true;
791              break;
792            case 2:
793              vDes = MAX_VDES;
794              break;
795            case 3:
796              vDes = NOM_VDES;
797              break;
798          }
799          Serial.println(" pressed");
800        } else {
801          switch (buttNum) {
802            case 1:
803              myPins[4] = false;
804              break;
805            case 5:
806              myPins[0] = false;
807              break;
808            case 6:
809              myPins[1] = false;
810              break;
811            case 7:
812              myPins[2] = false;
813              break;
814            case 8:
815              myPins[3] = false;
816              break;
817          }
818          pressedOr = " released";
819          Serial.println(" released");
820        }
821      }
822    } else {
823      state = 0;
824      Serial.println(String() + F("state = ") + state);
825    }
826  }
827
```

Packet Parser (for Bluetooth) File

luefruitConfig.h | packetParser.cpp | PIDConfig.h | finalSystem.ino

```cpp
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
// #if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined (_VARIANT_ARDUINO_ZERO_) && n
//   #include <SoftwareSerial.h>
// #endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
// #include "Adafruit_BluefruitLE_UART.h"


#define PACKET_ACC_LEN                  (15)
#define PACKET_GYRO_LEN                 (15)
#define PACKET_MAG_LEN                  (15)
#define PACKET_QUAT_LEN                 (19)
#define PACKET_BUTTON_LEN               (5)
#define PACKET_COLOR_LEN                (6)
#define PACKET_LOCATION_LEN             (15)

//    READ_BUFSIZE            Size of the read buffer for incoming packets
#define READ_BUFSIZE                    (20)


/* Buffer to hold incoming characters */
uint8_t packetbuffer[READ_BUFSIZE+1];

/**************************************************************************/
/*!
    @brief  Casts the four bytes at the specified address to a float
*/
/**************************************************************************/
float parsefloat(uint8_t *buffer)
{
  float f;
  memcpy(&f, buffer, 4);
  return f;
}

/**************************************************************************/
/*!
    @brief  Prints a hexadecimal value in plain characters
    @param  data      Pointer to the byte data
    @param  numBytes  Data length in bytes
*/
/**************************************************************************/
void printHex(const uint8_t * data, const uint32_t numBytes)
{
  uint32_t szPos;
  for (szPos=0; szPos < numBytes; szPos++)
  {
    Serial.print(F("0x"));
    // Append leading 0 for small values
    if (data[szPos] <= 0xF)
```

```
54      if (data[szPos] <= 0xF)
55      {
56        Serial.print(F("0"));
57        Serial.print(data[szPos] & 0xf, HEX);
58      }
59      else
60      {
61        Serial.print(data[szPos] & 0xff, HEX);
62      }
63      // Add a trailing space if appropriate
64      if ((numBytes > 1) && (szPos != numBytes - 1))
65      {
66        Serial.print(F(" "));
67      }
68    }
69    Serial.println();
70  }
71
72  /**************************************************************************/
73  /*!
74      @brief  Waits for incoming data and parses it
75  */
76  /**************************************************************************/
77  uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout)
78  {
79    uint16_t origtimeout = timeout, replyidx = 0;
80
81    memset(packetbuffer, 0, READ_BUFSIZE);
82
83    while (timeout--) {
84      if (replyidx >= 20) break;
85      if ((packetbuffer[1] == 'A') && (replyidx == PACKET_ACC_LEN))
86        break;
87      if ((packetbuffer[1] == 'G') && (replyidx == PACKET_GYRO_LEN))
88        break;
89      if ((packetbuffer[1] == 'M') && (replyidx == PACKET_MAG_LEN))
90        break;
91      if ((packetbuffer[1] == 'Q') && (replyidx == PACKET_QUAT_LEN))
92        break;
93      if ((packetbuffer[1] == 'B') && (replyidx == PACKET_BUTTON_LEN))
94        break;
95      if ((packetbuffer[1] == 'C') && (replyidx == PACKET_COLOR_LEN))
96        break;
97      if ((packetbuffer[1] == 'L') && (replyidx == PACKET_LOCATION_LEN))
98        break;
99
100     while (ble->available()) {
101       char c =  ble->read();
102       if (c == '!') {
103         replyidx = 0;
104       }
105       packetbuffer[replyidx] = c;
106       replyidx++;
107       timeout = origtimeout;
```

```
    if ((packetbuffer[1] == 'M') && (replyidx == PACKET_MAG_LEN))
      break;
    if ((packetbuffer[1] == 'Q') && (replyidx == PACKET_QUAT_LEN))
      break;
    if ((packetbuffer[1] == 'B') && (replyidx == PACKET_BUTTON_LEN))
      break;
    if ((packetbuffer[1] == 'C') && (replyidx == PACKET_COLOR_LEN))
      break;
    if ((packetbuffer[1] == 'L') && (replyidx == PACKET_LOCATION_LEN))
      break;

    while (ble->available()) {
      char c =  ble->read();
      if (c == '!') {
        replyidx = 0;
      }
      packetbuffer[replyidx] = c;
      replyidx++;
      timeout = origtimeout;
    }

    if (timeout == 0) break;
    delay(1);
  }

  packetbuffer[replyidx] = 0;  // null term

  if (!replyidx)  // no data or timeout
    return 0;
  if (packetbuffer[0] != '!')  // doesn't start with '!' packet beginning
    return 0;

  // check checksum!
  uint8_t xsum = 0;
  uint8_t checksum = packetbuffer[replyidx-1];

  for (uint8_t i=0; i<replyidx-1; i++) {
    xsum += packetbuffer[i];
  }
  xsum = ~xsum;

  // Throw an error message if the checksum's don't match
  if (xsum != checksum)
  {
    Serial.print("Checksum mismatch in packet : ");
    printHex(packetbuffer, replyidx+1);
    return 0;
  }

  // checksum passed!
  return replyidx;
}
```

PID and Bluetooth Configuration Files

finalSystem.ino | BluefruitConfig.h | PIDConfig.h | packetf

```
// COMMON SETTINGS
// ------------------------------------
// These settings are used in both SW U
// ------------------------------------
#define BUFSIZE
#define VERBOSE_MODE
#define BLE_READPACKET_TIMEOUT


// SOFTWARE UART SETTINGS
// ------------------------------------
// The following macros declare the pin
// You should use this option if you ar
// ------------------------------------
#define BLUEFRUIT_SWUART_RXD_PIN
#define BLUEFRUIT_SWUART_TXD_PIN
#define BLUEFRUIT_UART_CTS_PIN
#define BLUEFRUIT_UART_RTS_PIN


// HARDWARE UART SETTINGS
// ------------------------------------
// The following macros declare the HW
// this line if you are connecting the
// ------------------------------------
#ifdef Serial1    // this makes it not
  #define BLUEFRUIT_HWSERIAL_NAME
#endif


// SHARED UART SETTINGS
// ------------------------------------
// The following sets the optional Mode
// ------------------------------------
#define BLUEFRUIT_UART_MODE_PIN


// SHARED SPI SETTINGS
// ------------------------------------
// The following macros declare the pin
// SCK, MISO and MOSI should be connect
// using HW SPI.  This should be used w
// that use SPI (Bluefruit LE SPI Frien
// ------------------------------------
#define BLUEFRUIT_SPI_CS
#define BLUEFRUIT_SPI_IRQ
#define BLUEFRUIT_SPI_RST

// SOFTWARE SPI SETTINGS
// ------------------------------------
// The following macros declare the pin
// This should be used with nRF51822 ba
// (Bluefruit LE SPI Friend).
// ------------------------------------
#define BLUEFRUIT_SPI_SCK
#define BLUEFRUIT_SPI_MISO
#define BLUEFRUIT_SPI_MOSI
```

finalSystem.ino | BluefruitConfig.h | PIDConfig.h | packetParser.cpp

```
// COMMON SETTINGS
// ------------------------------------------------------------------------

#define HUSKYSPEED 20

// k values for horizontal tracking
#define Kp 0.05
#define Ki 0.01
#define Kd 0.3

#define IMax 100

#define SCREEN_X_CENTER 160
#define angleTolerance 50
int prevXDifference = 0;
int pErrorX = 0;



// k value for vertical tracking
#define Klp 0.3
#define Kli 0.3
#define Kld 0.8

#define lIMax 30
#define SCREEN_Y_CENTER 120
#define VERTICAL_TOLERANCE 45
int prevYDifference = 0;
int pErrorY = 0;


// k value for width tracking
#define Kwp 0.2
#define Kwi 3
#define Kwd 0.1
#define wIMax 100
#define WIDTH_TOLERANCE 50
int prevWDifference = 0;
int pErrorW = 0;

// speeds
#define Ksp 8
#define Ksi 3
#define Ksd 0

#define sIMax 50

#define outerTurningSpeedFraction 2
#define innerTurningSpeedFraction 0.7

int iError = 0;

// 14 desireable speed when motor running

#define NOM_VDES 16
#define MAX_VDES 44
int vDes = NOM_VDES;


int prevLeftDifference = 0;
int prevRightDifference = 0;

int iLError = 0;
int iRError = 0;

#define SURVEY_SPEED 5
#define SWEEP_INTERVAL 4000
bool swept = false;
int surveyTimeComp;
bool timeReset = false;

// PWM channel
// right motor motor 1, left motor motor 2
#define leftPWM 23
#define rightPWM 13
```

Comments: There are a total of three files posted here.