```
/*
-----------------------------------------------------------------------
--------------
  Main Controller Code to run on Teensy 4.0
  Author: Daniel Hoven Date: 2/24/2021 Project: Senior Capstone
  Requires: Adafruit Unified Sensor Lib. Adafruit BNO055
  avr/io header
  avr/interrupt header
  PWMServo Library (interchangeable with Servo.h stdlib)

-----------------------------------------------------------------------
--------------*/

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <Servo.h>

/* ====================================================================
======================
  Define controller constants */

#define kp 2                                    // Altitude PID proportional
constant
#define ki 0//0.5                               // Altitude PID integral
constant
#define kd 0.01                                 // Altitude PID derivative
constant
#define filtAlt 0.95                            // Altitude Estimation Filter
constant
#define LQRmult 0.6                             // Scaling factor for control
law, varies between 0.5-1
#define LQR_P 0.35                              // LQR_P constant
#define LQR_E 1                                 // Integrator
#define INTEGRATOR_CLAMP 0.5                    // Clamping term for integrator
#define filtPID 0.95
#define SLEW_LIMIT 10                           // controller gimbal limit
```

```
#define SLEW_FILTER 0.25          // Controller rate limiter (0-1),
higher = slower/stabler


/* ==================================================================
=======================
  Define communication setup */

#ifndef RADIO_BAUDRATE
#define RADIO_BAUDRATE 57600    // Telemetry radio baudrates (use 57600)
#endif

#ifndef LIDAR_BAUDRATE          // LiDAR sensor UART speed. default is
115200
#define LIDAR_BAUDRATE 115200
#endif

#ifndef USB_BAUDRATE
#define USB_BAUDRATE 115200     // USB Serial port baudtate. (N/A for
USB mode)
#endif

#ifndef SERIAL_USB
#define SERIAL_USB Serial       // Serial port for USB communication
(always Serial)
#endif

#ifndef RADIO_SERIAL
#define RADIO_SERIAL Serial4    // Serial port for radio communication
#endif

#ifndef LIDAR_SERIAL
#define LIDAR_SERIAL Serial1    // Serial port for LiDAR sensor
#endif


/* ==================================================================
=======================
  Define motor setup */
```

```
#define ESC1 6
#define ESC2 7
#define ESC3 5
#define ESC4 4
#define MAXVAL 1500
#define MINVAL 1000


/* Set the delay between iterations */

#define MAIN_DELAY 1.5

/* ==================================================================
======================
  Declare Library Objects

*/

Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28);

Servo esc1,
      esc2,
      esc3,
      esc4;

/* ==================================================================
=======================
  Declare globals */

int cycletime = 0,
    pos = 0,
    t = 0,
    i = 0,
    E_old,
    tim_old = 0;

float e1 = 0,
      e2 = 0,
      e3 = 0,
      e4 = 0,
```

```
      Ecal[4] = {0, 0, 0, 0};

volatile double _q0 = 0,
                _q1 = 1,
                _q2 = 0,
                _q3 = 0;

volatile float liDARold = 0,
               _lidar = 0;

unsigned int checksum = 0,
             check2 = 0,
             check1,
             altSet = 20,
             Xrot = 360,
             Yrot = 360;

volatile int liDARval = 0,
             strength = 0;

double pitch,
       roll,
       yaw,
       d_roll,
       d_pitch,
       d_yaw,
       alt = 0,
       dt = 0,
       I;

double EulerHist[3][5];

double X_Full [6] = {0, 0, 0, 0, 0, 0}, // RAM1 arrays
       R [6] = {0, 0, 0, 0, 0, 0},
       X_int [6],
       X_old [6];

double K [4][6] = {{
    0, 0, 0, 0, 0, 0,
  }, {
```

```
    25, 0, 0, -35, 0, 0,
  }, {
    0, 25, 0, 0, -35, 0,
  }, {
    0, 0, 0, 0, 0, 0,
  },
};

char Dcode[3];

uint16_t Ncode;

String STATE = "STARTUP";

double U [4] = {25, 0, 0, 0},
               Rcal[3] = {0, 0, 0};

/*= == == == == == == == == == == == == == == == == == == == == == ==
== == == == == == == == == == = == == == == == == == =
       SETUP */

void set_liDAR() {
  LIDAR_SERIAL.write(0x42);
  LIDAR_SERIAL.write(0x57);
  LIDAR_SERIAL.write(0x02);
  LIDAR_SERIAL.write(0x00);
  LIDAR_SERIAL.write(0x00);
  LIDAR_SERIAL.write(0x00);
  LIDAR_SERIAL.write(0x01);
  LIDAR_SERIAL.write(0x06);
}

void calibrateESCs() {

  esc1.attach(ESC1);
  esc2.attach(ESC2);
  esc3.attach(ESC3);
  esc4.attach(ESC4);
  delay(100);
  esc1.write(30);
```

```
  esc2.write(30);
  esc3.write(30);
  esc4.write(30);
  delay(1000);

  for (int i = 30; i > 20; i--) {
    esc1.write(i);
    esc2.write(i);
    esc3.write(i);
    esc4.write(i);
    delay(50);
  }
  for (int i = 20; i < 35; i++) {
    esc1.write(i);
    esc2.write(i);
    esc3.write(i);
    esc4.write(i);
    delay(150);
  }
  delay(2000);
  esc1.write(50);
  esc2.write(50);
  esc3.write(50);
  esc4.write(50);
  delay(2000);
}
void setup()

{
  /* Open Serial Ports*/

  SERIAL_USB.begin(250000);
  LIDAR_SERIAL.begin(115200);
  RADIO_SERIAL.begin(RADIO_BAUDRATE);
  delay(100);

  /* put liDAR in std. output mode */

  set_liDAR();
```

```
  while (!bno.begin()) {
    /* There was a problem detecting the BNO055 ... check your
connections */
    SERIAL_USB.print("Ooops, no BNO055 detected ... Check your wiring
or I2C ADDR!");
    delay(200);
  }

  SERIAL_USB.print("IMU found\n");
  bno.setExtCrystalUse(true);

  for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 6; j++) {
      K[i][j] *= LQRmult;
    }
  }

  calibrateESCs();
  delay(500);

  get_IMU_sample();
  delay(10);

  get_IMU_sample();

  float q0 = _q0;
  float q1 = _q1;
  float q2 = _q2;
  float q3 = _q3;

  R[0] = -atan2(2.0 * (q3 * q2 + q0 * q1) , 1.0 - 2.0 * (q1 * q1 + q2 *
q2));
  R[1] = asin(2.0 * (q2 * q0 - q3 * q1));
  R[2] = -atan2(2.0 * (q3 * q0 + q1 * q2) , - 1.0 + 2.0 * (q0 * q0 + q1
* q1));

  Rcal[0] = R[0];
  Rcal[1] = R[1];
  Rcal[2] = R[2];
```

```
}

/*-------------------------------------------------------------------
---------
 * main

-------------------------------------------------------------------
------*/
void loop(void) {

  dt = micros() - t;
  t = micros();
  dt = dt / 1000000;


  get_Distance_sample();

  _lidar = (1 - filtAlt) * liDARval + filtAlt * liDARold;
  liDARold = _lidar;
  alt = _lidar * cos(roll) * cos(pitch);


  // cosine error removal (altitude)

  get_IMU_sample();
  IntegralTracker();
  ELQR_calc();
  AltitudePID();
  commandESCs();
  printData();

  delay(MAIN_DELAY);
}
/*-------------------------------------------------------------------
---
 * Write external functions

-------------------------------------------------------------------
---*/
FASTRUN void get_IMU_sample() {
```

```
  /* get quaternions */

  imu::Quaternion quat = bno.getQuat();
  _q0 = quat.w();
  _q1 = quat.x();
  _q2 = quat.y();
  _q3 = quat.z();

  double q0 = _q0;
  double q1 = _q1;
  double q2 = _q2;
  double q3 = _q3;

  double roll_old = roll;
  double pitch_old = pitch;
  double yaw_old = yaw;

  //quaternion conversion

  roll = (-Rcal[0]) - atan2(2.0 * (q3 * q2 + q0 * q1), 1.0 - 2.0 * (q1
* q1 + q2 * q2)); //* (180/PI);
  pitch = (-Rcal[1]) + asin(2.0 * (q2 * q0 - q3 * q1));// * (180/PI);
  yaw = (-Rcal[2]) - atan2(2.0 * (q3 * q0 + q1 * q2), -1.0 + 2.0 * (q0
* q0 + q1 * q1)); //* (180/PI);

  for (int i = 0; i < 6; i++) {
    X_old[i] = X_Full[i];
  }

  X_Full[0] = roll;
  X_Full[1] = pitch;
  X_Full[2] = yaw;

  // Compute Derivatives using 5 point stencil
  double h = dt;
  for (int i = 0; i < 3; i++) {
    for (int j = 1; j < 5; j++) {
      EulerHist[i][j] = EulerHist[i][j - 1];
    }
```

```
    EulerHist[i][0] = X_Full[i];
    double temp;
    temp = (-1) * EulerHist[i][0] + (8) * EulerHist[i][1] + (-8) *
EulerHist[i][3] + (1) * EulerHist[i][4];
    temp /= 12 * h; X_Full[i + 3] = temp;
  }
}

void get_Distance_sample() {

  if (LIDAR_SERIAL.available() >= 9) // When at least 9 bytes of data
available (expected number of bytes for 1 signal), then read
  {
    if ((0x59 == LIDAR_SERIAL.read()) && (0x59 == LIDAR_SERIAL.read()))
// byte 1and byte 2
    {
      unsigned int t1 = LIDAR_SERIAL.read(); // byte 3 = Dist_L
      unsigned int t2 = LIDAR_SERIAL.read(); // byte 4 = Dist_H
      t2 <<= 8;
      t2 += t1;
      liDARval = t2;
      t1 = LIDAR_SERIAL.read(); // byte 5 = Strength_L
      t2 = LIDAR_SERIAL.read(); // byte 6 = Strength_H
      t2 <<= 8;
      t2 += t1;
      strength = t2;
      for (int i = 0; i < 3; i++)LIDAR_SERIAL.read(); // ignore
remaining bytes
    }
  }
}

FASTRUN void ELQR_calc() {

  for (int i = 1; i < 4; i++) {
    double iter = 0;
    for (int j = 0; j < 3; j++) {
      iter += K[i][j] * ((X_Full[j] - R[j]) + LQR_E * (X_int[j]));
    }
    for (int j = 3; j < 6; j++) {
```

```
      iter += K[i][j] * ((X_Full[j] - R[j]) + LQR_P * (X_int[j]));
    }
    if (abs(iter) < SLEW_LIMIT) {
      U[i] = iter;
    }
    else if (iter > 0) {
      U[i] = SLEW_LIMIT;
    }
    else if (iter < 0) {
      U[i] = -SLEW_LIMIT;
    }

  }
}

FASTRUN void IntegralTracker() {

  for (int i = 0; i < 3; i++) {
    X_int[i] += (dt / 2) * (X_old[i] + X_Full[i]); X_int[i + 3] =
X_Full[i];
  }
  for (int i = 0; i < 3; i++) {
    if (abs(X_int[i] - R[i]) > INTEGRATOR_CLAMP) {
      if ((X_int[i] - R[i]) > 0) {
        X_int[i] = INTEGRATOR_CLAMP + R[i] - 0.01;
      } else if ((X_int[i] - R[i]) < 0) {
        X_int[i] = -(INTEGRATOR_CLAMP + R[i]) + 0.01;
      }
    }
  }
}

FASTRUN void AltitudePID() {

  float E = altSet - alt;
  float P = E;

  if ((I < 30) && (I > (-30))) {
    I += E * dt;
  } else {
```

```
    if (I > 0)I = 29;
    if (I < 0)I = -29;
  }
  float D = (E - E_old) / dt;
  //U[0] = filtPID * U[0] + (1 - filtPID) * (kp * P + ki * I + kd * D);
  U[0] = U[0];
  E_old = E;

}

void commandESCs() {
  // Motor Mixing Algorithm

  float _e1 = U[0] - U[1] + U[2] + U[3];
  float _e2 = U[0] - U[1] - U[2] - U[3];
  float _e3 = U[0] + U[1] - U[2] + U[3];
  float _e4 = U[0] + U[1] + U[2] - U[3];

  _e1 = map(_e1, 0, 180, 1000, 2000);
  _e2 = map(_e2, 0, 180, 1000, 2000);
  _e3 = map(_e3, 0, 180, 1000, 2000);
  _e4 = map(_e4, 0, 180, 1000, 2000);

  e1 *= (SLEW_FILTER);
  e2 *= (SLEW_FILTER);
  e3 *= (SLEW_FILTER);
  e4 *= (SLEW_FILTER);

  e1 += (1 - SLEW_FILTER) * _e1;
  e2 += (1 - SLEW_FILTER) * _e2;
  e3 += (1 - SLEW_FILTER) * _e3;
  e4 += (1 - SLEW_FILTER) * _e4;

  if ((Ecal[0] + Ecal[1] + Ecal[2] + Ecal[3]) == 0) {

    float eAv = (e1 + e2 + e3 + e4) / 4;
    Ecal[0] = eAv - e1;
    Ecal[1] = eAv - e2;
    Ecal[2] = eAv - e3;
    Ecal[3] = eAv - e4;
```

```
  }

  e1 += Ecal[0];
  e2 += Ecal[1];
  e3 += Ecal[2];
  e4 += Ecal[3];

  if ((e1 < MAXVAL) && (e1 > MINVAL)) {
    esc1.writeMicroseconds((int)e1);
  } else if (e1 < MINVAL) {
    e1 = MINVAL + 1;
  } else if (e1 > MAXVAL) {
    e1 = MAXVAL - 1;
  }

  if ((e2 < MAXVAL) && (e2 > MINVAL)) {
    esc2.writeMicroseconds((int)e2);
  } else if (e2 < MINVAL) {
    e2 = MINVAL + 1;
  } else if (e2 > MAXVAL) {
    e2 = MAXVAL - 1;
  }
  if ((e3 < MAXVAL) && (e3 > MINVAL)) {
    esc3.writeMicroseconds((int)e3);
  } else if (e3 < MINVAL) {
    e3 = MINVAL + 1;
  } else if (e3 > MAXVAL) {
    e3 = MAXVAL - 1;
  }
  if ((e4 < MAXVAL) && (e4 > MINVAL)) {
    esc4.writeMicroseconds((int)e4);
  } else if (e4 < MINVAL) {
    e4 = MINVAL + 1;
  } else if (e4 > MAXVAL) {
    e4 = MAXVAL - 1;
  }
}
void printData() {

  // RADIO_SERIAL.println();
```

```
  // RADIO_SERIAL.print("1:"), RADIO_SERIAL.print((int)e1);
  // RADIO_SERIAL.print(", 2:"), RADIO_SERIAL.print((int)e2);
  // RADIO_SERIAL.print(", 3:"), RADIO_SERIAL.print((int)e3);
  // RADIO_SERIAL.print(", 4:"), RADIO_SERIAL.println((int)e4);

  SERIAL_USB.print("1:"), SERIAL_USB.print((int)e1);
  SERIAL_USB.print(", 2:"), SERIAL_USB.print((int)e2);
  SERIAL_USB.print(", 3:"), SERIAL_USB.print((int)e3);
  SERIAL_USB.print(", 4:"), SERIAL_USB.println((int)e4);

  // SERIAL_USB.print("X_int1: "),
  // Serial.println(X_int[0]);
  // Serial.println(dt, 4);

  // // //
  //
  //
  // Serial.println();
  // SERIAL_USB.println(U[0]);
  // SERIAL_USB.println(U[1]);
  // SERIAL_USB.println(U[2]);
  // SERIAL_USB.println(U[3]);
  // Serial.println();
  //
  // SERIAL_USB.print("X[Full] = ");
  // SERIAL_USB.print("1: "),SERIAL_USB.print(X_Full[0], 4);
  // SERIAL_USB.print("2: "),SERIAL_USB.print(X_Full[1], 4);
  // SERIAL_USB.print("3: "),SERIAL_USB.println(X_Full[2], 4);
  // RADIO_SERIAL.print(", "),RADIO_SERIAL.print(X_Full[3], 3);
  // RADIO_SERIAL.print(", "),RADIO_SERIAL.print(X_Full[4], 3);
  // RADIO_SERIAL.print(", "),RADIO_SERIAL.println(X_Full[5], 3);
  // SERIAL_USB.print("4 "), SERIAL_USB.println(e4);
  // SERIAL_USB.print("R[0] : "), SERIAL_USB.println(R[0]);
  // SERIAL_USB.print("R[1] : "), SERIAL_USB.println(R[1]);
}
void receiveData() {

  if (RADIO_SERIAL.available() >= 5) {

    if ((RADIO_SERIAL.read() == 0x20) && (RADIO_SERIAL.read() == 0x20))
```

```
{

      char temp = RADIO_SERIAL.read();
      if (temp == 'a') {
        Dcode[0] = '-', Dcode[1] = 'x';
        R[0] += 0.01;
        digitalWrite(13, HIGH);
      } else if (temp == 'w') {
        Dcode[0] = '+', Dcode[1] = 'y';
        R[1] += 0.01;
        digitalWrite(13, HIGH);
      } else if (temp == 's') {
        Dcode[0] = '-', Dcode[1] = 'y'; R[1] -= 0.01;
        digitalWrite(13, HIGH);
      } else if (temp == 'd') {
        Dcode[0] = '+', Dcode[1] = 'x'; R[0] -= 0.01;
        digitalWrite(13, HIGH);
      } else if (temp == 'H') {
        Dcode[0] = 'H', Dcode[1] = 'O'; R[0] = Rcal[0];
        R[1] = Rcal[1];
        digitalWrite(13, HIGH);
      } else if (temp == 'r') {
        Dcode[0] = '+', Dcode[1] = 'z'; U[0] += 1;
      } else if (temp == 'f') {
        Dcode[0] = '-', Dcode[1] = 'z'; U[0] -= 1;
        digitalWrite(13, HIGH);
      } else if (temp == 'K') {
        altSet = 20;
      } else {
        Dcode[0] = 'N', Dcode[1] = 'A';
      }
      uint16_t t1 = RADIO_SERIAL.read();
      uint16_t t2 = RADIO_SERIAL.read();

      t2 <<= 8;
      t1 += t2;

      if (!((Dcode[0] == 'N') || (Dcode[1] == 'A'))) {
        Ncode = t1;
      }
```

```
      digitalWrite(13, LOW);
    }
  } else {
    digitalWrite(13, LOW);
  }
}
```