```
#include <buffer.h>
#include <crc.h>
#include <datatypes.h>
#include <VescUart.h>
#include <SoftwareSerial.h>
#include <PID_v1.h>

#include "Global_Variables.h"
#include "functions.h"
/** Initiate vesc1Uart class */
VescUart vesc1;
VescUart vesc2;

PID myPID1(&TrlActual, &TrlOut, &Trl, Kp, Ki, Kd, DIRECT);
PID myPID2(&TrrActual, &TrrOut, &Trr, Kp, Ki, Kd, DIRECT);

float current = 0.0; /** The current in amps */
int throt = 0;
int stear = 0;
int counter = 0;
int thPin = A0;
int stePin = A1;
float v = 0.0;

void setup()
{
 myPID1.SetMode(AUTOMATIC);
 myPID2.SetMode(AUTOMATIC);

  /** Setup Serial port to display data */
 Serial.begin(115200);

  /** Setup SoftwareSerial port */
 Serial1.begin(115200);
 Serial2.begin(115200);

  /** Define which ports to use as UART */
 vesc1.setSerialPort(&Serial1);
 vesc2.setSerialPort(&Serial2);

Serial.println("t,StVin,ThVin,TrlOut,TrrOut,Trl,Trr,deltaTorque");
}

void loop() {
  if (TimeStamp + deltaT <= millis()){
    TimeStamp = millis();
```

```
 dt = double(TimeStamp - prevtime) / 1000.0;
 // put your main code here, to run repeatedly:
 vesc1.getVescValues();
 vesc2.getVescValues();

 //Update the PID Variables through serial
 getSerial();



 throt = analogRead(thPin);
 stear = analogRead(stePin);
 throttleVolts = map(throt, 0, 1023, 0, 5000)*0.001;
 steeringVolts = map(stear, 0, 1023, 0, 5000)*0.001;

 // --- CONVERT INPUTS ---
 // find steering angle of wheels
 fsteeringAngle();

 // find total torque output from input throttle
 TotalTorque();

 // --- UPDATE KINEMATICS ---
 // Inputs from motor controllers
 Nrr = vesc2.data.rpm /14; // RPM of motor 1
 Nrl = vesc1.data.rpm /14; // RPM of motor 2
 Vrr = vesc2.data.inpVoltage;
 Vrl = vesc1.data.inpVoltage;

 // Update kinematics from the new motor controller output data
 Kinematics();

 // --- FIND DESIRED DIFFERENCE IN TORQUE ---
 DeltaTorque();

 // --- FIND TORQUE TO EACH WHEEL ---
 WheelTorque();

 // --- PID FOR EACH TORQUE ---
TrlActual = CurrentToTorque(vesc1.data.avgMotorCurrent,Nrl,Vrl);
TrrActual = CurrentToTorque(vesc2.data.avgMotorCurrent,Nrr,Vrr);
 myPID1.Compute();
 myPID2.Compute();

 // --- CHECK FOR LIMITATIONS
 // Find max torque for slipping
 MaxTorqueSlip();
```

```
  // Find max torque for max power
  MaxTorquePower();

  // Limit torques due do slip and max power
  CheckSlip();
  CheckPower();

  // Convert output torque to current
  currentRR =  TorqueToCurrent(TrrOut, Nrr, Vrr);
  currentRL =  TorqueToCurrent(TrlOut, Nrl, Vrl);
  if (currentRR > 9.0) {
    currentRR = 9.0;
  }
  if (currentRR < -9.0) {
    currentRR = -9.0;
  }
  if (currentRL > 9.0) {
    currentRL = 9.0;
  }
  if (currentRL < -9.0) {
    currentRL = -9.0;
  }


  vesc1.setCurrent(currentRL);
  vesc2.setCurrent(currentRR);

  Serial.print(TimeStamp);//Center on zero
  Serial.print("\t");
  Serial.print(steeringVolts);//Center on zero
  Serial.print("\t");
  Serial.print(throttleVolts);
  Serial.print("\t");
  Serial.print(TrlOut*1000);
  Serial.print("\t");
  Serial.print(TrrOut*1000);
  Serial.print("\t");
  Serial.print(TrlActual*1000);
  Serial.print("\t");
  Serial.print(TrrActual*1000);
  Serial.print("\t");
  Serial.println(deltaTorque*1000);

//  Serial.print("Steering");
//  Serial.println(steeringVolts);
```

```
  //  Serial.println(steeringAngle);
  //  Serial.print(vesc1.data.avgMotorCurrent);
  //   Serial.print("M2 IN  ");
  //  Serial.print(vesc2.data.avgMotorCurrent);
  //   Serial.print("M1 Out  ");
   // Serial.println(currentRR);
  //   Serial.print("M2 Out  ");
  //   Serial.println(currentRL);
    prevtime = TimeStamp;

  }
}

// Send P10I02D01Rx for Kp = 1.0 Ki = 0.2 Kd = 0.1 Run
void getSerial() {
  if (Serial.available())
  {
    char myChar = Serial.read();
    if (myChar == 'p' | myChar == 'P')
    {
      Kp = Serial.parseInt() * 0.01;
      //Serial.print("Kp: ");
      //Serial.println(Kp);
    }
    else if (myChar == 'i' | myChar == 'I')
    {
      Ki = Serial.parseInt() * 0.1;
      //Serial.print("Ki: ");
      //Serial.println(Ki);
    }
    else if (myChar == 'd' | myChar == 'D')
    {
      Kd = Serial.parseInt() * 0.001;
      //Serial.print("Kd: ");
      //Serial.println(Kd);
    }
    else if (myChar == 'v' | myChar == 'V')
    {
      v = Serial.parseInt() * 0.1;
      //Serial.print("v: ");
      //Serial.println(v);
    }
    else if (myChar == 'r' | myChar == 'R')
    {
      // Starts the motors auto on/off cycle
      r = 1;
```

```
    }
    else if (myChar == 's' | myChar == 'S')
    {
      // Stops the motors auto on/off cycle
      r = 0;
    }
   myPID1.SetTunings(Kp, Ki, Kd);
   myPID2.SetTunings(Kp, Ki, Kd);



  }
}
```