```
/*************************************************************
 * Programmer: Stephen D Dunn, Shawn Mull, Tarang Hirani
 * Program: Universal Vehicle Chassis Controller
 * Date: 9/16/2015
 * Time: 22:45
 * Description: This program is intended to be a command and control
 * module for vehicle automation. It will have multiple modules with
 * classes and functions that will be called from a C# Console Application
 * via COM port and return values to the command interface
 * to indicate Val. InteriorLightingModule, ExteriorLightingModule,
 * WindowControlModule, ActiveSuspensionModule, ActiveAerodynamicsModule
 * and EnvironmentalControlModule.
 * Revision Date: 11/10/2015
 *************************************************************/
/*Libraries*/
#include <Stepper.h>
#include <Servo.h>
#include "Arduino.h"
#include <SPI.h>
#include <SoftwareSerial.h>

//Active Suspension Module
//Front Left Compression and Rebound Motor Pins
//  const int FLCompMotorPin1
//  const int FLCompMotorPin2
//  const int FLRebMotorPin1
//  const int FLRebMotorPin2
//  const int AnalogHeight1

//Front Right Compression and Rebound Motor Pins
//  const int FRCompMotorPin1
//  const int FRCompMotorPin2
//  const int FRRebMotorPin1
//  const int FRRebMotorPin2
//  const int AnalogHeight2

//Rear Left Compression and Rebound Motor Pins
//  const int RLCompMotorPin1
//  const int RLCompMotorPin2
//  const int RLRebMotorPin1
//  const int RLRebMotorPin2
//  const int AnalogHeight3

//Rear Right Compression and Rebound Motor Pins
```

```
//  const int RRCompMotorPin1
//  const int RRCompMotorPin2
//  const int RRRebMotorPin1
//  const int RRRebMotorPin2
//  const int AnalogHeight4

//Windshield Motor Control Variables
const int WiperSwitchPos1 = 11;
const int WiperSwitchPos2 = 12;
const int WiperSwitchPos3 = 13;
const int WiperMotorPin1 = 14;
const int WiperMotorPin2 = 15;
const int WiperMotorSpeed = 16;

//Running Light Constant Variables
const int RunningLight = 17;
const int RunningLight_IO = 18;

//LOW Beam Lighting Constant Variables
const int LowBeams = 19;
const int LowBeam_IO = 20;

//High Beam Lighting Constant Variables
const int HighBeams = 21;
const int HighBeam_IO = 22;

//Left Blinker Lighting Constant Variables
const int LeftBlinker = 23;
const int LBlinker_IO = 24;

//Right Blinker Lighting Constant Variables
const int RightBlinker = 25;
const int RBlinker_IO = 26;

//Emergency Lighting Constant Variables
const int EmergencyLights = 27;

//Brake Lights Constant Variables
const int BrakeLights = 28;
const int BrakeLights_IO = 29;

//Interior Lighting Constant Variables
const int dsideDoor = 30;
const int psideDoor = 31;
```

```
const int rHatchDoor = 32;
const int InteriorLighting_IO = 33;
const int DimmerSwitch = 34;

//Drivers Side Window Inputs Variables
const int iFLWindowUp = 35;
const int iFLWindowDn = 36;

//Passengers Side Front Window Inputs Variables
const int iFRWindowUp = 37;
const int iFRWindowDn = 38;

//Rear Passengers Side Left Window Inputs Variables
const int iRLWindowUp = 39;
const int iRLWindowDn = 40;

//Rear Passengers Side Right Window Inputs Variables
const int iRRWindowUp = 41;
const int iRRWindowDn = 42;

//Sunroof Inputs Variables
const int iSunroofOpen = 43;
const int iSunroofClosed = 44;

//Front Left Window Outputs Variables
const int oFLWindowMotorPin1 = 45;
const int oFLWindowMotorPin2 = 46;

//Front Right Window Outputs Variables
const int oFRWindowMotorPin1 = 47;
const int oFRWindowMotorPin2 = 48;

//Rear Left Window Outputs Variables
const int oRLWindowMotorPin1 = 49;
const int oRLWindowMotorPin2 = 50;

//Rear Right Window Outputs Variables
const int oRRWindowMotorPin1 = 51;
const int oRRWindowMotorPin2 = 52;

//SunRoof Outputs Variables
const int oSunroofMotorPin1 = 53;
const int oSunroofMotorPin2 = 54;
```

```
//Set Pin Assignents
const int gear1 = 7;
const int gear2 = 6;
const int gear3 = 5;
const int gear4 = 4;
const int gear5 = 3;
const int gearR = 8;

//Define Variables
int leftPin = A0;
int rightPin = A0;
int reversePin = A0;
int neutralPin = A0;
int left_tap = 0;
int right_tap = 0;
int reverse_tap = 0;
int neutral_tap = 0;
int gear = 0;

//Wiper Control Module Variables
int WiperMotorPin1Val;
int WiperMotorPin2Val;
int WiperSwitchPos1Val;
int WiperSwitchPos2Val;
int WiperSwitchPos3Val;
int WiperMotorSpeedVal;

// Active Suspension Variables
//  int ModeSelectPassiveVal;
//  int ModeSelectActiveVal;
//  int ModeSelectOffVal;
//
// Analog Height Sensor Values
//  int AnalogHeight1Val;
//  int AnalogHeight2Val;
//  int AnalogHeight3Val;
//  int AnalogHeight4Val;
//
// Front Left Compression and Rebound Motor Values
//  int FLCompMotorPin1Val;
//  int FLCompMotorPin2Val;
//  int FLRebMotorPin1Val;
//  int FLRebMotorPin2Val;
//
```

```
// Front Right Compression and Rebound Motor Values
//  int FRCompMotorPin1;
//  int FRCompMotorPin2;
//  int FRRebMotorPin1;
//  int FRRebMotorPin2;
//
// Rear Left Compression and Rebound Motor Values
//  int RLCompMotorPin1;
//  int RLCompMotorPin2;
//  int RLRebMotorPin1;
//  int RLRebMotorPin2;
//
// Rear Right Compression and Rebound Motor Values
//  int RRCompMotorPin1;
//  int RRCompMotorPin2;
//  int RRRebMotorPin1;
//  int RRRebMotorPin2;
//  int AnalogHeight4;

//Active Aerodynamics
Servo wing1;
int wing;
int throttle;
int brake;
int steer;
int thottleThresh = 90;

//Roaming Variables
int HeadLightVal = 0;
int LowBeamVal = 0;
int EmergencyLightVal = 0;
int BrakeLightVal = 0;
int RunningLightVal = 0;

//Roaming variables
int DoorVal = 0;

//Declare Variable Vals
int FLWindowUpVal = 0;
int FLWindowDnVal = 0;
int FRWindowUpVal = 0;
int FRWindowDnVal = 0;
int RLWindowUpVal = 0;
int RLWindowDnVal = 0;
```

```
int RRWindowUpVal = 0;
int RRWindowDnVal = 0;
int SunroofOpenVal = 0;
int SunroofClosedVal = 0;

void setup(){

  //Define Window Control Module
  pinMode(iFLWindowUp, INPUT);
  pinMode(iFLWindowDn, INPUT);
  pinMode(iFRWindowUp, INPUT);
  pinMode(iFRWindowDn, INPUT);
  pinMode(iRLWindowUp, INPUT);
  pinMode(iRLWindowDn, INPUT);
  pinMode(iRRWindowUp, INPUT);
  pinMode(iRRWindowDn, INPUT);

  //Define Interior Lighting Module
  pinMode(InteriorLighting_IO, OUTPUT);
  pinMode(dsideDoor, INPUT);
  pinMode(psideDoor, INPUT);
  pinMode(rHatchDoor, INPUT);

  // Define Sequential Shifter
  pinMode(gear1, OUTPUT);
  pinMode(gear2, OUTPUT);
  pinMode(gear3, OUTPUT);
  pinMode(gear4, OUTPUT);
  pinMode(gear5, OUTPUT);
  pinMode(gearR, OUTPUT);

  //Initialize Gears
  digitalWrite(gear1, LOW);
  digitalWrite(gear2, LOW);
  digitalWrite(gear3, LOW);
  digitalWrite(gear4, LOW);
  digitalWrite(gear5, LOW);
  digitalWrite(gearR, LOW);

  //Define Exterior Lighting Module
  pinMode(LowBeams, INPUT);
  pinMode(LowBeam_IO, OUTPUT);
  pinMode(HighBeams, INPUT);
  pinMode(HighBeam_IO, OUTPUT);
```

```
  pinMode(RunningLight, INPUT);
  pinMode(RunningLight_IO, OUTPUT);
  pinMode(LeftBlinker, INPUT);
  pinMode(LBlinker_IO, OUTPUT);
  pinMode(RightBlinker, INPUT);
  pinMode(RBlinker_IO, OUTPUT);
  pinMode(EmergencyLights, INPUT);
  pinMode(BrakeLights, INPUT);
  pinMode(BrakeLights_IO, OUTPUT);

  //Define Window Motor Control Outputs
  pinMode(oFLWindowMotorPin1, OUTPUT);
  pinMode(oFLWindowMotorPin2, OUTPUT);

  //Front Right Window Outputs
  pinMode(oFRWindowMotorPin1, OUTPUT);
  pinMode(oFRWindowMotorPin2, OUTPUT);

  //Rear Left Window Outputs
  pinMode(oRLWindowMotorPin1, OUTPUT);
  pinMode(oRLWindowMotorPin2, OUTPUT);

  //Rear Right Window Outputs
  pinMode(oRRWindowMotorPin1, OUTPUT);
  pinMode(oRRWindowMotorPin2, OUTPUT);

  //SunRoof Outputs
  pinMode(oSunroofMotorPin1, OUTPUT);
  pinMode(oSunroofMotorPin2, OUTPUT);

  wing1.attach(9);
  //Setup Communication
  Serial.begin(9600);
}

void loop()
{
  SequentialShiftControl();
  // ActiveSuspensionModule();
  EnvironmentalControlModule();
  ExteriorLightingModule();
  InteriorLightingModule();
  WindowControlModule();
  // IgnitionControlModule();
```

```
 WiperControlModule();

 // read the input on analog pin 0 (Throttle Position):
 throttle = analogRead(A4);
 // read the input on analog pin 1 (Brake pressure);
 brake = analogRead(A5);
 // read the input on analog pin 2 (Steering Possition);
 steer = analogRead(A6);

 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 // Throttle
 float throttlePos = ((4.4 - (throttle * (5.0 / 1023.0))) / (4.4 - 1.3)) * 100;
 // Brake
 float brakePres = (brake * (5.0 / 1023.0));
 // Steering
 float SteeringPos = (steer * (5.0 / 1023.0));

 // controling rear wing
 // change wing if throttle goes high and steering straight
 if ((throttlePos > thottleThresh) && (2 < SteeringPos < 3)) {
  wing1.write(0);
 }
 //air brake if steering straight and brakes go high
 else if ((2 < SteeringPos < 3) && (brakePres > 4.9)) {
  wing1.write(180);
 }
 //lift wing if turning
 else if ((brakePres < 4.9) && (SteeringPos > 3)) {
  wing1.write(80);
 }
 else {
  wing1.write(40);
 }
}
void ActiveSuspensionModule(){

}

void EnvironmentalControlModule(){

}
void ExteriorLightingModule() {
 HeadLightVal = digitalRead(HighBeams);
 if (HeadLightVal == HIGH) {
```

```
  digitalWrite(HighBeam_IO, LOW);
 }
 else
 {
  digitalWrite(HighBeam_IO, HIGH);
 }
 LowBeamVal = digitalRead(LowBeams);
 if (LowBeamVal == HIGH) {
  digitalWrite(LowBeam_IO, LOW);
 }
 else
 {
  digitalWrite(LowBeam_IO, HIGH);
 }
 EmergencyLightVal = digitalRead(EmergencyLights);
 if (EmergencyLightVal == LOW) {
  digitalWrite(RBlinker_IO, LOW);
  digitalWrite(LBlinker_IO, LOW);
  delay(1000);
  digitalWrite(RBlinker_IO, HIGH);
  digitalWrite(LBlinker_IO, HIGH);
  delay(1000);
 }
 else
 {
  digitalWrite(RBlinker_IO, HIGH);
  digitalWrite(LBlinker_IO, HIGH);
 }
 RunningLightVal = digitalRead(RunningLight);
 if (RunningLightVal == HIGH) {
  digitalWrite(RunningLight_IO, LOW);
 }
 else
 {
  digitalWrite(RunningLight_IO, HIGH);
 }
}

void IgnitionControlModule(){

}

void InteriorLightingModule() {
```

```
  DoorVal = digitalRead(dsideDoor);
  if (DoorVal == HIGH) {
    digitalWrite(InteriorLighting_IO, LOW);
  }
  else
  {
    digitalWrite(InteriorLighting_IO, HIGH);
  }
  DoorVal = digitalRead(psideDoor);
  if (DoorVal == HIGH) {
    digitalWrite(InteriorLighting_IO, LOW);
  }
  else
  {
    digitalWrite(InteriorLighting_IO, HIGH);
  }
  DoorVal = digitalRead(rHatchDoor);
  if (DoorVal == HIGH) {
    digitalWrite(InteriorLighting_IO, LOW);
  }
  else
  {
    digitalWrite(InteriorLighting_IO, HIGH);
  }
}

int SequentialShiftControl() {
  left_tap = analogRead(leftPin);
  right_tap = analogRead(rightPin);
  reverse_tap = analogRead(reversePin);
  neutral_tap = analogRead(neutralPin);

  if (left_tap > 500) {
    switch (gear) {
      case -1:
        //do nothing
        break;

      case 0:
        //do nothing
        break;

      case 1:
        //do nothing
```

```
      break;

    case 2:
      //downshift to gear 1
      digitalWrite(gear2, LOW);
      delay(100);
      digitalWrite(gear1, HIGH);
      gear = 1;
      break;

    case 3:
      //downshift to gear 2
      digitalWrite(gear3, LOW);
      delay(100);
      digitalWrite(gear2, HIGH);
      gear = 2;
      break;

    case 4:
      //downshift to gear 3
      digitalWrite(gear4, LOW);
      delay(100);
      digitalWrite(gear3, HIGH);
      gear = 3;
      break;

    case 5:
      //downshift to gear 4
      digitalWrite(gear5, LOW);
      delay(100);
      digitalWrite(gear4, HIGH);
      gear = 4;
      break;
  }
}

if (right_tap > 500) {
  switch (gear) {
    case -1:
      //upshift to gear 0
      digitalWrite(gearR, LOW);
      delay(100);
      gear = 0;
      break;
```

```
    case 0:
      //upshift to gear 1
      digitalWrite(gear1, HIGH);
      gear = 1;
      break;

    case 1:
      //upshift to gear 2
      digitalWrite(gear1, LOW);
      delay(100);
      digitalWrite(gear2, HIGH);
      gear = 2;
      break;

    case 2:
      //upshift to gear 3
      digitalWrite(gear2, LOW);
      delay(100);
      digitalWrite(gear3, HIGH);
      gear = 3;
      break;

    case 3:
      //upshift to gear 4
      digitalWrite(gear3, LOW);
      delay(100);
      digitalWrite(gear4, HIGH);
      gear = 4;
      break;

    case 4:
      //upshift to gear 5
      digitalWrite(gear4, LOW);
      delay(100);
      digitalWrite(gear5, HIGH);
      gear = 1;
      break;

    case 5:
      //do nothing
      break;
  }
}
```

```
  if (reverse_tap > 500) {
    switch (gear) {
      case -1:
        //do nothing
        break;

      case 0:
        //shift into reverse
        digitalWrite(gearR, HIGH);
        gear = -1;
        break;

      case 1:
        //do nothing
        break;
      case 2:
        //do nothing
        break;
      case 3:
        //do nothing
        break;
      case 4:
        //do nothing
        break;
      case 5:
        //do nothing
        break;
    }
  }

  if (neutral_tap > 500) {
    digitalWrite(gear1, LOW);
    digitalWrite(gear2, LOW);
    digitalWrite(gear3, LOW);
    digitalWrite(gear4, LOW);
    digitalWrite(gear5, LOW);
    digitalWrite(gearR, LOW);
    gear = 0;
  }
}

int StartupCalibrationModule(){
```

```
}

void WindowControlModule() {

 FLWindowUpVal == digitalRead(iFLWindowUp);
 if (FLWindowUpVal == HIGH) {
  digitalWrite(oFLWindowMotorPin1, LOW);
  digitalWrite(oFLWindowMotorPin2, HIGH);
 }
 else
 {
  digitalWrite(oFLWindowMotorPin1, HIGH);
  digitalWrite(oFLWindowMotorPin2, HIGH);
 }
 FLWindowDnVal == digitalRead(iFLWindowDn);
 if (FLWindowDnVal == HIGH) {
  digitalWrite(oFLWindowMotorPin1, HIGH);
  digitalWrite(oFLWindowMotorPin2, LOW);
 }
 else
 {
  digitalWrite(oFLWindowMotorPin1, HIGH);
  digitalWrite(oFLWindowMotorPin2, HIGH);
 }
 FRWindowUpVal = digitalRead(iFRWindowUp);
 if (FRWindowUpVal == HIGH) {
  digitalWrite(oFRWindowMotorPin1, LOW);
  digitalWrite(oFRWindowMotorPin2, HIGH);
 }
 else
 {
  digitalWrite(oFRWindowMotorPin1, HIGH);
  digitalWrite(oFRWindowMotorPin2, HIGH);
 }
 FRWindowDnVal = digitalRead(iFRWindowDn);
 if (FRWindowDnVal == HIGH) {
  digitalWrite(oFRWindowMotorPin1, LOW);
  digitalWrite(oFRWindowMotorPin2, HIGH);
 }
 else
 {
  digitalWrite(oFRWindowMotorPin1, HIGH);
  digitalWrite(oFRWindowMotorPin2, HIGH);
 }
```

```
RLWindowUpVal == digitalRead(iRLWindowUp);
if (RLWindowUpVal == HIGH) {
  digitalWrite(oRLWindowMotorPin1, LOW);
  digitalWrite(oRLWindowMotorPin2, HIGH);
}
else
{
  digitalWrite(oRLWindowMotorPin1, HIGH);
  digitalWrite(oRLWindowMotorPin2, HIGH);
}
RLWindowDnVal == digitalRead(iRLWindowDn);
if (RLWindowDnVal == HIGH) {
  digitalWrite(oRLWindowMotorPin1, HIGH);
  digitalWrite(oRLWindowMotorPin2, LOW);
}
else
{
  digitalWrite(oRLWindowMotorPin1, HIGH);
  digitalWrite(oRLWindowMotorPin2, HIGH);
}
RRWindowUpVal == digitalRead(iRRWindowUp);
if (RRWindowUpVal == HIGH) {
  digitalWrite(oRRWindowMotorPin1, LOW);
  digitalWrite(oRRWindowMotorPin2, HIGH);
}
else
{
  digitalWrite(oRRWindowMotorPin1, HIGH);
  digitalWrite(oRRWindowMotorPin2, HIGH);
}
RRWindowDnVal == digitalRead(iRRWindowDn);
if (RRWindowDnVal == HIGH) {
  digitalWrite(oRRWindowMotorPin1, HIGH);
  digitalWrite(oRRWindowMotorPin2, LOW);
}
else
{
  digitalWrite(oRRWindowMotorPin1, HIGH);
  digitalWrite(oRRWindowMotorPin2, HIGH);
}
SunroofOpenVal, digitalRead(iSunroofOpen);
if (SunroofOpenVal == HIGH) {
  digitalWrite(oSunroofMotorPin1, LOW);
  digitalWrite(oSunroofMotorPin2, HIGH);
```

```
  }
  else
  {
    digitalWrite(oSunroofMotorPin1, HIGH);
    digitalWrite(oSunroofMotorPin2, HIGH);
  }
  SunroofClosedVal == digitalRead(iSunroofClosed);
  if (SunroofClosedVal == HIGH) {
    digitalWrite(oSunroofMotorPin1, HIGH);
    digitalWrite(oSunroofMotorPin2, LOW);
  }
  else
  {
    digitalWrite(oSunroofMotorPin1, HIGH);
    digitalWrite(oSunroofMotorPin2, HIGH);
  }
}

void WiperControlModule() {

  WiperSwitchPos1Val == digitalRead(WiperSwitchPos1);
  while (WiperSwitchPos1Val, HIGH) {
    digitalWrite(WiperMotorPin1, HIGH);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 85);
  }
  WiperSwitchPos1Val == digitalRead(WiperSwitchPos1);
  while (WiperSwitchPos1Val, LOW) {
    digitalWrite(WiperMotorPin1, LOW);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 0);
  }
  WiperSwitchPos2Val == digitalRead(WiperSwitchPos2);
  while (WiperSwitchPos2Val, HIGH) {
    digitalWrite(WiperMotorPin1, HIGH);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 170);
  }
  WiperSwitchPos2Val == digitalRead(WiperSwitchPos2);
  while (WiperSwitchPos2Val, LOW) {
    digitalWrite(WiperMotorPin1, LOW);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 0);
  }
```

```
  WiperSwitchPos3Val == digitalRead(WiperSwitchPos3);
  while (WiperSwitchPos3Val, HIGH) {
    digitalWrite(WiperMotorPin1, HIGH);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 255);
  }
  WiperSwitchPos3Val == digitalRead(WiperSwitchPos3);
  while (WiperSwitchPos3Val, LOW) {
    digitalWrite(WiperMotorPin1, LOW);
    digitalWrite(WiperMotorPin2, LOW);
    digitalWrite(WiperMotorSpeedVal, 0);
  }
}
```