**Engine Control Unit:**

Engine Control Unit is a module that consists electronic differential system, PID controller.

**Engine Control Unit (ECU) Codes:**

#include <PID\_v1.h>

#include <SoftwareSerial.h>

//Serial definitions

#define tachometerSerial1 Serial1

#define tachometerSerial2 Serial2

#define motorDriverSerial1 Serial3

#define motorDriverSerial2 Serial

SoftwareSerial sensorKitSerial(52,53);

//Pin definitions

#define pedalPin1 0

#define pedalPin2 1

#define gearPin 51

#define encoderPin 3

//Constants

#define gasPedalTreshold 100

#define tresholdPID 50

#define consKp 1

#define consKi 1

#define consKd 1

#define aggKp 5

#define aggKi 5

#define aggKd 5

//Varible definitions

int currMotor1 = 0;

int currMotor2 = 0;

int tempMotor1 = 0;

int tempMotor2 = 0;

int battStat = 0;

int speed = 0;

char gear = 'D';

int gasPedal = 0;

String gasPedalString = "0";

double rpm1 = 0;

double rpm2 = 0;

int rpm1\_int = 0;

int rpm2\_int = 0;

double desRpm1 = 0;

double desRpm2 = 0;

double pwm1 = 0;

double pwm2 = 0;

int angle = 0;

int counter = 0;

//PID cons

PID motorPID1(&rpm1, &pwm1, &desRpm1, consKp, consKi, consKd, DIRECT);

PID motorPID2(&rpm2, &pwm1, &desRpm2, consKp, consKi, consKd, DIRECT);

void setup() {

sensorKitSerial.begin(9600);

tachometerSerial1.begin(9600);

tachometerSerial2.begin(9600);

motorDriverSerial1.begin(9600);

motorDriverSerial2.begin(9600);

motorPID1.SetMode(AUTOMATIC);

motorPID2.SetMode(AUTOMATIC);

pinMode(gearPin,INPUT);

}

void loop() {

readGear();

readGasPedal();

readTachometer();

readEncoder();

calculateDesiredRPM();

computePID();

readMotorCurrent();

sendPWM();

sendToSensorKit();

}

void readGasPedal(){

int gas1 = analogRead(pedalPin1);

int gas2 = analogRead(pedalPin2);

if(abs(gas1-gas2) > gasPedalTreshold){

gasPedal = min(gas1,gas2);

}else{

gasPedal= gas1;

//gasPedal = gasPedal >> 2;

}

if(gear == 'D')

gasPedal+=1000;

else if(gear == 'R')

gasPedal+=2000;

gasPedalString = String(gasPedal);

}

void readTachometer(){

if(tachometerSerial1.available()){

String inData1 = tachometerSerial1.readStringUntil('?');

char inDataArray1[inData1.length()+1];

inData1.toCharArray(inDataArray1,inData1.length());

sscanf(inDataArray1, "\*%d",&rpm1\_int);

rpm1 = rpm1\_int;

}

if(tachometerSerial2.available()){

String inData2 = tachometerSerial2.readStringUntil('?');

char inDataArray2[inData2.length()+1];

inData2.toCharArray(inDataArray2,inData2.length());

sscanf(inDataArray2, "\*%d",&rpm2\_int);

rpm2 = rpm2\_int;

speed = (rpm1+rpm2)/2;

}

}

void readMotorCurrent(){

if(motorDriverSerial1.available()){

String inData1 = motorDriverSerial1.readStringUntil('?');

char inDataArray1[inData1.length()+1];

inData1.toCharArray(inDataArray1,inData1.length());

sscanf(inDataArray1, "\*%d",&currMotor1);

}

if(motorDriverSerial2.available()){

String inData2 = motorDriverSerial2.readStringUntil('?');

char inDataArray2[inData2.length()+1];

inData2.toCharArray(inDataArray2,inData2.length());

sscanf(inDataArray2, "\*%d",&currMotor2);

}

}

void sendPWM(){

if(motorDriverSerial1.available()){

Serial.println("1");

String strPWM1 = String((int)pwm1);

motorDriverSerial1.print("\*"+strPWM1+"?");

}

if(motorDriverSerial2.available()){

String strPWM2 = String((int)pwm2);

motorDriverSerial1.print("\*"+strPWM2+"?");

}

}

void sendToSensorKit(){

String currMotorString1 = String(currMotor1);

String currMotorString2 = String(currMotor2);

String speedString = String((int)speed);

Serial.println(speed);

String tempMotorString1 = String(tempMotor1);

String tempMotorString2 = String(tempMotor2);

String outData = "\*"+currMotorString1+"/"+currMotorString2+"/"+speedString+"/"+tempMotorString1+"/"+tempMotorString2+"/"+gear+"?";

sensorKitSerial.print(outData);

Serial.println(outData);

}

void readGear(){

if(digitalRead(gearPin) == LOW)

gear = 'R';

else

gear = 'D';

}

void readEncoder(){

angle = analogRead(encoderPin);

angle = map(angle , 0, 1024, 0, 360);

}

void calculateDesiredRPM(){

//calculations according to formula

}

void computePID(){

double gap1 = abs(rpm1-desRpm1);

if(gap1 > tresholdPID)

motorPID1.SetTunings(aggKp,aggKi,aggKd);

else

motorPID1.SetTunings(aggKp,aggKi,aggKd);

double gap2 = abs(rpm2-desRpm2);

if(gap2 > tresholdPID)

motorPID2.SetTunings(aggKp,aggKi,aggKd);

else

motorPID2.SetTunings(aggKp,aggKi,aggKd);

motorPID1.Compute();

motorPID2.Compute();

}

**Sensor Kit:**

SensorKit is a module that collects data such as speed, battery status, battery and motor driver tempeture etc. from Engine Control Unit (ECU), Battery Management System (BMS) and sends them to Telemetri module, shows necessary informations on HMI screen and saves them for drive test.

**SensorKit Codes:**

#include <VirtualWire.h>

#include <SPI.h>

#include <SD.h>

//Serial definetions

#define ECUSerial Serial1

#define BMSSerial Serial3

#define HMISerial Serial2

#define RFSerial Serial

//Function prototypes

void readFromECU(String inData);

void saveToSD();

void setupSD();

void setupRF();

void sentToRF();

void updateHMI();

//Sensor variables

int speed = 100;

String speedString = "0";

int currMotor1 = 0;

String currMotorString1= "0";

int currMotor2 = 0;

String currMotorString2= "0";

int tempMotor1 = 0;

String tempMotorString1 = "0";

int tempMotor2 = 0;

String tempMotorString2 = "0";

int tempBattery = 0;

String tempBatteryString = "0";

int battStatus = 0;

String battStatusString = "0";

char gear = 'o';

int counter = 0;

//Data logger variables

Sd2Card sdCard;

SdVolume sdVolume;

SdFile root;

int pinCS = 53;

File logFile;

//RF Module Variables

int pinRF = 12;

void setup() {

ECUSerial.begin(9600);

BMSSerial.begin(9600);

HMISerial.begin(9600);

setupSD();

setupRF();

}

void loop() {

updateHMI();

readFromECU();

readFromBMS();

saveToSD();

sendToRF();

}

void setupSD(){

if (!sdCard.init(SPI\_QUARTER\_SPEED, pinCS)) {

return;

}else {

if(!SD.begin(pinCS)) {

return;

}else {

SD.begin(pinCS);

logFile = SD.open("data.csv",FILE\_WRITE);

logFile.println("Motor current 1,Motor Current 2,Motor Temperature 1,tMotor Temperature 2,Speed,Gear,Battery Temperature,Battery Status,Time");

logFile.close();

}

}

}

void saveToSD(){

logFile = SD.open("data.csv",FILE\_WRITE);

if(logFile){

logFile.print(currMotorString1);

logFile.print(",");

logFile.print(currMotorString1);

logFile.print(",");

logFile.print(tempMotorString1);

logFile.print(",");

logFile.print(tempMotorString2);

logFile.print(",");

logFile.print(speedString);

logFile.print(",");

logFile.print(gear);

logFile.print(",");

logFile.print(tempBatteryString);

logFile.print(",");

logFile.print(battStatusString);

logFile.print(",");

logFile.print(millis());

logFile.println();

logFile.close();

}

}

void sendToRF(){

String RFMessage = "\*"+currMotorString1+"/"+currMotorString2+"/"+tempMotorString1+"/"+tempMotorString2+"/"+speedString+"/"+tempBatteryString+"/"+battStatusString+"?";

char RFArray[RFMessage.length()+1];

RFMessage.toCharArray(RFArray,RFMessage.length());

vw\_send((uint8\_t \*)RFArray, RFMessage.length());

vw\_wait\_tx();

}

void setupRF(){

vw\_set\_ptt\_inverted(true);

vw\_set\_tx\_pin(pinRF);

vw\_setup(4000);

}

void readFromECU(){

if(ECUSerial.available()){

String inData = ECUSerial.readStringUntil('?');

char inDataArray[inData.length()+1];

//Serial.println(inData);

inData.toCharArray(inDataArray,inData.length());

sscanf(inDataArray, "\*%d/%d/%d/%d/%d/%c",&currMotor1,&currMotor2,&speed,&tempMotor1,&tempMotor2,&gear);

//double to Str convert

currMotorString1 = String(currMotor1);

currMotorString2 = String(currMotor2);

speedString = String(speed);

tempMotorString1 = String(tempMotor1);

tempMotorString2 = String(tempMotor2);

}

}

void updateHMI(){

if(counter == 10000){

HMISerial.print("speedGauge.val=");

HMISerial.print(speed);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("motorTemp1.txt=");

HMISerial.write(0x22);

HMISerial.print(tempMotor1);

HMISerial.write(0x22);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("motorTemp2.txt=");

HMISerial.write(0x22);

HMISerial.print(tempMotor2);

HMISerial.write(0x22);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("batteryTemp.txt=");

HMISerial.write(0x22);

HMISerial.print(tempBattery);

HMISerial.write(0x22);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("batteryStat.txt=");

HMISerial.write(0x22);

HMISerial.print(battStatus);

HMISerial.write(0x22);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("gearText.txt=");

HMISerial.write(0x22);

HMISerial.print(gear);

HMISerial.write(0x22);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("yearNum.val=rtc0");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("yearNum.val=rtc0");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("monthNum.val=rtc1");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("dayNum.val=rtc2");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("hourNum.val=rtc3");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("minuteNum.val=rtc4");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.print("secondNum.val=rtc5");

HMISerial.write(0xff);

HMISerial.write(0xff);

HMISerial.write(0xff);

counter = 0;

}

counter++;

}

void readFromBMS(){

String inData = BMSSerial.readStringUntil('?');

char inDataArray[inData.length()+1];

inData.toCharArray(inDataArray,inData.length());

sscanf(inDataArray,"\*%d?",&battStatus);

battStatusString = String(battStatus);

}

**Tachometer:**

Tachometer is a module that measures the speed of the vehicle with contrast sensors.

**Tachometer Codes:**

int val=0;

long last=0;

int stat=LOW;

int stat2;

int count=0;

int sens=200;

double rpm=0;

String rpmStr;

int nLine=3; // the number of blades of the propeller

int milisecond=10; // the time it takes each reading

void setup()

{

Serial.begin(9600);

pinMode(13,OUTPUT);

}

void loop()

{

val=analogRead(0);

if(val>sens)

stat=HIGH;

else

stat=LOW;

digitalWrite(13,stat);

if(stat2!=stat){ //counts when the state change, thats from (dark to light) or

//from (light to dark), remmember that IR light is invisible for us.

count++;

stat2=stat;

}

if(millis()-last>=milisecond){

rpm=((double)count/nLine)\*600;

rpmStr=String((int)rpm);

Serial.print("\*"+rpmStr+"?");

count=0;

last=millis();

}

}

**Motor Driver Codes:**

Motor driver is a module that controls motor speed according to the values of gas pedal and angle of the steering wheel.

unsigned short hall;

unsigned char MoveTableF[8] = {0, 33, 6, 36, 24, 9, 18, 0};

unsigned char MoveTableR[8] = {0, 18, 9, 24, 36, 6, 33, 0};

char receivedText[5]={'1','0','0','0','/'};

int dir=1;

char amp[3]={0,0,0};

int amp\_int=0;

//Uart data receiving function

void serialDataRead()

{

int i=0;

while(i<5){

if(UART1\_Data\_Ready())

{

receivedText[i]=UART1\_Read();

i++;

}

}

}

//Data converting function

void convertReceivedData()

{

dir=receivedText[0]-'0';

amp\_int+=(receivedText[1]-'0')\*100;

amp\_int+=(receivedText[2]-'0')\*10;

amp\_int+=receivedText[3]-'0';

}

//Stop Button Function

void Interrupt(){

if(INTF\_bit){

INTCON = 0;

PORTB.F2 = 0;

PORTD = 0;

PWM1\_Stop();

}

if (RBIF\_bit){

RBIF\_bit = 0;

hall = PORTB;

hall = hall & 112;

hall = hall >> 4;

if(dir==1)

PORTD = MoveTableF[hall];

else if(dir==2)

PORTD = MoveTableR[hall];

}

}

void InitInterrupts(){

OPTION\_REG = 0;

TMR1IE\_bit = 0;

INTCON = 0xD8;

}

void main() {

ADCON1 = 14;

CMCON = 7;

PORTB = 0;

TRISB = 243;

PORTD = 0;

TRISD = 0;

InitInterrupts();

PWM1\_Init(10000);

UART1\_Init(9600);

while(1){

PORTB.F2 = 1;

serialDataRead();

convertReceivedData();

PORTB.F2 = 0;

hall = PORTB;

hall = hall & 112;

hall = hall >> 4;

if(dir==1)

PORTD = MoveTableF[hall];

else if(dir==2){

PORTD = MoveTableR[hall];

}

PORTB.F2 = 0;

PWM1\_Set\_Duty(amp\_int);

amp\_int=0;

PWM1\_Start();

}

}

**Note:** All codes except Motor Driver had written in Arduino IDE, Motor Driver Codes had written in MicroC.