#include <TinyGPS.h>

#include <Adafruit\_GPS.h>

#if ARDUINO >= 100

#include <SoftwareSerial.h>

#include "NixieTube.h"

#else

#endif

////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

int speedkph = 0;

int speedmph = 0;

int switchPin = 7; // to switch from kph to mph

int ledPin = 13;

int first = 0;

int second = 0;

volatile byte rpmcount;

unsigned int rpm;

unsigned long timeold;

#define MAX\_DELTA 500

int nixie = 3;

int nixieRPM;

int last\_rpm = 0;

void rpm\_fun()

{

//Each rotation, this interrupt function is run twice, so take that into consideration for

//calculating RPM

//Update count

rpmcount++;

}

NixieTube tube(11, 12, 13, 10, 2); // NixieTube(DIN,ST,SH,OE,NUM)

// DIN(D11) - PIN1

// ST(D12) - PIN2

// SH(D13) - PIN3

// OE(D10) - PIN4

// GND - PIN5

// 5v - PIN6

// The pin number of the module is defined in the datasheet

// Connect the GPS Power pin to 5V

// Connect the GPS Ground pin to ground

// Connect the GPS TX (transmit) pin to Digital 3

// Connect the GPS RX (receive) pin to Digital 2

#if ARDUINO >= 100

SoftwareSerial mySerial(3, 2);

#else

NewSoftSerial mySerial(3, 2);

#endif

Adafruit\_GPS GPS(&mySerial);

#define GPSECHO true // set to true if you want to see the raw data parsed

boolean usingInterrupt = false;

void useInterrupt(boolean);

////////////////////////////////////////////////////////////////////////////////////

////////////////////////////--SETUP--///////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

void setup()

{

pinMode(nixie, OUTPUT);

pinMode(switchPin, INPUT);

pinMode(ledPin, OUTPUT);

Serial.begin(9600);

//Interrupt 0 is digital pin 2, so that is where the IR detector is connected

//Triggers on FALLING (change from HIGH to LOW)

attachInterrupt(0, rpm\_fun, FALLING);

rpmcount = 0;

rpm = 0;

timeold = 0;

////////////////////////////////////////////////////////////////////////////////////

////////////////////////////--GPS SETUP--//////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

// connect at 115200 so we can read the GPS fast enough and echo without dropping chars

Serial.begin(115200);

GPS.begin(9600);

GPS.sendCommand(PMTK\_SET\_NMEA\_OUTPUT\_RMCGGA);

GPS.sendCommand(PGCMD\_ANTENNA);

useInterrupt(true);

// Set the update rate

GPS.sendCommand(PMTK\_SET\_NMEA\_UPDATE\_1HZ); // 1 Hz update rate

delay(1000);

}

void updateNixie(int new\_rpm) {

if(new\_rpm > last\_rpm) {

if(new\_rpm > last\_rpm + MAX\_DELTA) {

last\_rpm += MAX\_DELTA;

} else {

last\_rpm = new\_rpm;

}

} else {

if(new\_rpm < last\_rpm - MAX\_DELTA) {

last\_rpm -= MAX\_DELTA;

} else {

last\_rpm = new\_rpm;

}

}

analogWrite(nixie, last\_rpm);

}

SIGNAL(TIMER0\_COMPA\_vect) {

char c = GPS.read();

if (GPSECHO)

if (c) UDR0 = c;

}

void useInterrupt(boolean v) {

if (v) {

OCR0A = 0xAF;

TIMSK0 |= \_BV(OCIE0A);

usingInterrupt = true;

} else {

TIMSK0 &= ~\_BV(OCIE0A);

usingInterrupt = false;

}

}

uint32\_t timer = millis();

////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////--PROGRAM--////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

void loop() // run over and over again

{

if (! usingInterrupt) {

char c = GPS.read();

if (GPSECHO)

if (c) UDR0 = c;

}

if (GPS.newNMEAreceived()) {

if (!GPS.parse(GPS.lastNMEA()))

return;

}

if (timer > millis()) timer = millis();

if (millis() - timer > 250) {

timer = millis();

/////--FOR REV COUNTER--/////

//Update RPM every second

delay(250);

//Don't process interrupts during calculations

detachInterrupt(0);

//Note that this would be 60\*1000/(millis() - timeold)\*rpmcount if the interrupt

//happened once per revolution instead of twice. Other multiples could be used

//for multi-bladed propellers or fans

rpm = 30\*1000/(millis() - timeold)\*rpmcount;

timeold = millis();

rpmcount = 0;

nixieRPM = (rpm/50) + 30;

Serial.print(rpm);

Serial.println(" RPM");

updateNixie(nixieRPM);

//Restart the interrupt processing

attachInterrupt(0, rpm\_fun, FALLING);

speedkph = GPS.speed \* 1.852;

speedmph = GPS.speed \* 1.15078;

int sp = GPS.satellites;

int first = sp/10;

int second = sp-(first \* 10);

///////// If GPS Fix///////////////////////

if (GPS.fix = 1){

Serial.println("GPS Fix");

Serial.print("sat: ");

Serial.println(GPS.satellites);

//--FOR SPEEDOMETER--//

tube.setBackgroundColor(Black);

tube.setNumber(0,first);

tube.setNumber(1,second);

tube.display();

}

////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////--END--///////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////

}

}