/\* 10-9-6-5

 Pat's BT-IR Combo 4 Wheel Drive Finger On to Run, Off to Stop.

 This Combo BT-IR Sketch is controlled by an Android Phone, IR Remote M and Mini Remote.

 There was a conflict using PWM pin 11 when you combined both BT & IR, Library analogWrite with pin 11?

 Modified by Pat McMahon (V7) 18/5/2021, in1=10, in2=9, in3=6, in4=5, added Front pin 3, Middle extra pin 13, Back Lights pin 4, HORN pin 9.

 Adjusted the code plus for combinations of backleft, backright, frontleft, frontright. Main Code is Forward 2&4, Back1&3.

\*/

#include <IRremote.h>

const int ledBlue = 3;

const int ledRed = 4;

const int ledOrange = 13;

const int buzzer = 12;

const int RECV\_PIN = 2;

const int SEND\_PIN = 12;

#define USE\_NO\_CARRIER = 1;

int lastMiniIRCommand = 0;

IRrecv irrecv(RECV\_PIN,SEND\_PIN);

// connect motor controller pins to Arduino digital pins

// motor one

#define in1 10 //L298n Motor Driver pins.

#define in2 9

// motor two

#define in3 6

#define in4 5

const int delayTime = 300;

//#define LED 2

#define FRONTLIGHTS 3 //Blue LED's

#define BACKLIGHTS 4 //Red Led's

#define HORN 12 //Passive Buzzer

#define EXTRA 13 //Yellow Led's

int command; //Int to store app command state.

int Speed = 204; // 0 - 255.

int Speedsec;

int buttonState = 0;

int lastButtonState = 0;

int Turnradius = 0; //Set the radius of a turn, 0 - 255 Note:the robot will malfunction if this is higher than int Speed.

int brakeTime = 45;

int brkonoff = 1; //1 for the electronic braking system, 0 for normal.

void setup() {

 pinMode(in1, OUTPUT);

 pinMode(in2, OUTPUT);

 pinMode(in3, OUTPUT);

 pinMode(in4, OUTPUT);

 // pinMode(LED, OUTPUT); //Set the LED pin.

 pinMode(FRONTLIGHTS, OUTPUT);

 pinMode(BACKLIGHTS, OUTPUT);

 pinMode(HORN, OUTPUT);

 pinMode(EXTRA, OUTPUT);// MIDDLELIGHTS

 pinMode(LED\_BUILTIN, OUTPUT);

 Serial.begin(9600); //Set the baud rate to your Bluetooth module.

 // set all the motor control pins to outputs

 pinMode(ledBlue, OUTPUT);

 pinMode(ledRed, OUTPUT);

 pinMode(ledOrange, OUTPUT);

 pinMode(buzzer, OUTPUT);

 irrecv.enableIRIn(); // Start the receiver

}

void loop() {

 if (Serial.available() > 0) {

 command = Serial.read();

 Stop(); //Initialize with motors stoped.

 switch (command) {

 case 'F':

 forward();

 break;

 case 'B':

 back();

 break;

 case 'L':

 left();

 break;

 case 'R':

 right();

 break;

 case 'G':

 forwardleft();

 break;

 case 'I':

 forwardright();

 break;

 case 'H':

 backleft();

 break;

 case 'J':

 backright();

 break;

 case 'W':

 FrontLightsOn();

 break;

 case 'w':

 FrontLightsOff();

 break;

 case 'U':

 BackLightsOn();

 break;

 case 'u':

 BackLightsOff();

 break;

 case 'V':

 HornOn();

 break;

 case 'v':

 HornOff();

 break;

 case 'X':

 ExtraOn();

 break;

 case 'x':

 ExtraOff();

 break;

 Speed = 100;

 break;

 case '1':

 Speed = 140;

 break;

 case '2':

 Speed = 153;

 break;

 case '3':

 Speed = 165;

 break;

 case '4':

 Speed = 178;

 break;

 case '5':

 Speed = 191;

 break;

 case '6':

 Speed = 204;

 break;

 case '7':

 Speed = 216;

 break;

 case '8':

 Speed = 229;

 break;

 case '9':

 Speed = 242;

 break;

 case 'q':

 Speed = 255;

 break;

 }

 Speedsec = Turnradius;

 if (brkonoff == 1) {

 brakeOn();

 } else {

 brakeOff();

 }

 }

 decode\_results results;

 if (irrecv.decode(&results)) // if there is an IR reading

 {

 Serial.println(results.value, HEX);

 switch (results.value)

 {

 //Remote M

 case 0x2F0:

 Serial.println("Forwards!");

 forwards();

 break;

 case 0xAF0:

 Serial.println("Backwards!");

 backwards();

 break;

 case 0x2D0:

 Serial.println("Lefty!");

 lefty();

 break;

 case 0xCD0:

 Serial.println("Righty!");

 righty();

 break;

 case 0x738:

 Serial.println("middleLeds!"); // Middle Orange Led's

 middleLeds();

 break;

 case 0xF38:

 Serial.println("frontLeds!"); // Front Blue Led's

 frontLeds();

 break;

 case 0x338:

 Serial.println("backLeds!"); //Back Red Led's

 backLeds();

 break;

 case 0xB38:

 Serial.println("horn!");

 horn();

 break;

 case 0xA70:

 Serial.println("Stop!");

 halt();

 break;

 //Mini NEC Remote

 case 0xFF18E7:

 Serial.println("Forwards!");

 forwards();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF4AB5:

 Serial.println("Backwards!");

 backwards();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF10EF:

 Serial.println("Lefty!");

 lefty();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF5AA5:

 Serial.println("Righty!");

 righty();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF38C7:

 Serial.println("Stop!");

 halt();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF629D:

 Serial.println("middleLeds!"); // Middle Orange Led's

 middleLeds();

 lastMiniIRCommand = results.value;

 break;

 case 0xFFA25D:

 Serial.println("frontLeds!"); // Front Blue Led's

 frontLeds();

 lastMiniIRCommand = results.value;

 break;

 case 0xFFE21D:

 Serial.println("backLeds!"); //Back Red Led's

 backLeds();

 lastMiniIRCommand = results.value;

 break;

 case 0xFF02FD:

 Serial.println("horn!");

 horn();

 lastMiniIRCommand = results.value;

 break;

 case 0xFFFFFFFF:

 Serial.println("RepeatLastCommand!");

 Serial.println(lastMiniIRCommand, HEX);

 switch (lastMiniIRCommand)

 {

 case 0xFF18E7:

 Serial.println("Repeat Forwards!");

 forwards();

 break;

 case 0xFF4AB5:

 Serial.println("Repeat Backwards!");

 backwards();

 break;

 case 0xFF10EF:

 Serial.println("Repeat Left!");

 left();

 break;

 case 0xFF5AA5:

 Serial.println("Repeat Right!");

 right();

 break;

 case 0xFF38C7:

 Serial.println("Repeat Stop!");

 halt();

 break;

 }

 }

 irrecv.resume(); // Receive the next value

 }

}

//BT Commands

void forward() {

 Serial.println("BT Forward");

 //forwards();

 analogWrite(in2, Speed);

 analogWrite(in4, Speed);

}

void back() {

 Serial.println("BT Back");

 //backwards();

 analogWrite(in1, Speed);

 analogWrite(in3, Speed);

}

void left() {

 Serial.println("BT Left");

 //lefty();

 analogWrite(in2, Speed);

 analogWrite(in3, Speed);

}

void right() {

 Serial.println("BT Right");

 //righty();

 analogWrite(in4, Speed);

 analogWrite(in1, Speed);

}

void forwardleft() {

 analogWrite(in4, Speedsec);

 analogWrite(in2, Speed);

}

void forwardright() {

 Serial.println("BT Forwardright");

 analogWrite(in4, Speed);

 analogWrite(in2, Speedsec);

}

void backright() {

 Serial.println("BT Backright");

 analogWrite(in3, Speed);

 analogWrite(in1, Speedsec);

}

void backleft() {

 Serial.println("BT Backleft");

 analogWrite(in3, Speedsec);

 analogWrite(in1, Speed);

}

void FrontLightsOff() {

 digitalWrite(FRONTLIGHTS, LOW);

}

void FrontLightsOn() {

 digitalWrite(FRONTLIGHTS, HIGH);

}

void HornOff() {

 digitalWrite(HORN, LOW);

}

void HornOn() {

 digitalWrite(HORN, HIGH);

}

void ExtraOff() {

 digitalWrite(EXTRA, LOW);

}

void ExtraOn() {

 digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

 delay(500); // wait for a second

 digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

 delay(500); // wait for a second

}

void BackLightsOff() {

 digitalWrite(BACKLIGHTS, LOW);

}

void BackLightsOn() {

 digitalWrite(BACKLIGHTS, HIGH);

}

void Stop() {

 analogWrite(in1, 0);

 analogWrite(in2, 0);

 analogWrite(in3, 0);

 analogWrite(in4, 0);

}

// the loop function runs over and over again forever

void MIDDLELIGHTS() {

 digitalWrite(EXTRA, HIGH); // turn the LED on (HIGH is the voltage level)

 delay(1000); // wait for a second

 digitalWrite(EXTRA, LOW); // turn the LED off by making the voltage LOW

 delay(1000); // wait for a second

}

void brakeOn() {

 //Here's the future use: an electronic braking system!

 // read the pushbutton input pin:

 buttonState = command;

 // compare the buttonState to its previous state

 if (buttonState != lastButtonState) {

 // if the state has changed, increment the counter

 if (buttonState == 'S') {

 if (lastButtonState != buttonState) {

 digitalWrite(in1, HIGH);

 digitalWrite(in2, HIGH);

 digitalWrite(in3, HIGH);

 digitalWrite(in4, HIGH);

 delay(brakeTime);

 Stop();

 }

 }

 // save the current state as the last state,

 //for next time through the loop

 lastButtonState = buttonState;

 }

}

void brakeOff() {

}

//IR Commands

void forwards()

{

 // turn on motor A

 digitalWrite(in1, LOW);

 digitalWrite(in2, HIGH);

 digitalWrite(ledBlue, HIGH);

 // turn on motor B

 digitalWrite(in3, LOW);

 digitalWrite(in4, HIGH);

 digitalWrite(ledBlue, HIGH);

 delay(delayTime);

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void backwards()

{

 // turn on motor A

 digitalWrite(in1, HIGH);

 digitalWrite(in2, LOW);

 digitalWrite(ledRed, HIGH);

 // turn on motor B

 digitalWrite(in3, HIGH);

 digitalWrite(in4, LOW);

 digitalWrite(ledRed, HIGH);

 delay(delayTime);

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void righty()

{

 // turn on motor A

 digitalWrite(in1, HIGH);

 digitalWrite(in2, LOW);

 digitalWrite(ledOrange, HIGH);

 // turn on motor B

 digitalWrite(in3, LOW);

 digitalWrite(in4, HIGH);

 digitalWrite(ledOrange, HIGH);

 delay(delayTime);

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void lefty()

{

 // turn on motor A

 digitalWrite(in1, LOW);

 digitalWrite(in2, HIGH);

 digitalWrite(ledOrange, HIGH);

 // turn on motor B

 digitalWrite(in3, HIGH);

 digitalWrite(in4, LOW);

 digitalWrite(ledOrange, HIGH);

 delay(delayTime);

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void middleLeds()

{

 // turn on ledOrange

 digitalWrite(ledOrange, HIGH);

 delay(delayTime);

 // turn off ledOrange

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void frontLeds()

{

 // turn on ledBlue

 digitalWrite(ledBlue, HIGH);

 delay(delayTime);

 // turn off ledBlue

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void backLeds()

{

 // turn on ledRed

 digitalWrite(ledRed, HIGH);

 delay(delayTime);

 // turn off ledRed

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void horn()

{

 // turn on horn

 digitalWrite(buzzer, HIGH);

 delay(delayTime);

 // turn off horn

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);

}

void halt()

{

 // stop both motors

 digitalWrite(in1, LOW);

 digitalWrite(in2, LOW);

 digitalWrite(in3, LOW);

 digitalWrite(in4, LOW);

 digitalWrite(ledBlue, LOW);

 digitalWrite(ledRed, LOW);

 digitalWrite(ledOrange, LOW);

 digitalWrite(buzzer, LOW);;

}