Hybrid.ino

#include <hid.h>

#include <hiduniversal.h>

#include <usbhub.h>

#include "hidjoystickrptparser.h"

USB Usb;

USBHub Hub(&Usb);

HIDUniversal Hid(&Usb);

JoystickEvents JoyEvents;

JoystickReportParser Joy(&JoyEvents);

const int Leftin1 = 25;

const int Leftin2 = 23;

const int Leftpwm = 2;

const int Rightin1 = 45;

const int Rightin2 = 43 ;

const int Rightpwm = 8;

const int Fleftin1 = 37;

const int Fleftin2 = 35;

const int Fleftpwm = 5;

const int Frightin1 = 33;

const int Frightin2 = 31;

const int Frightpwm = 4;

const int Piston = 41;//A15 /\*41\*/;

void setup() {

Serial.begin(115200);

while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection

Serial.println("Start");

if (Usb.Init() == -1)

Serial.println("OSC did not start.");

delay(200);

if (!Hid.SetReportParser(0, &Joy))

ErrorMessage<uint8\_t > (PSTR("SetReportParser"), 1);

pinMode(Leftin1, OUTPUT);

pinMode(Leftin2, OUTPUT);

pinMode(Leftpwm, OUTPUT);

pinMode(Rightin1, OUTPUT);

pinMode(Rightin2, OUTPUT);

pinMode(Rightpwm, OUTPUT);

pinMode(Fleftin1, OUTPUT);

pinMode(Fleftin2, OUTPUT);

pinMode(Fleftpwm, OUTPUT);

pinMode(Frightin1, OUTPUT);

pinMode(Frightin2, OUTPUT);

pinMode(Frightpwm, OUTPUT);

pinMode(Piston, OUTPUT);

analogWrite(Piston, 255);

}

void loop() {

Usb.Task();

}

hidjoystickrptparser.h

const int leftin1 = 25;

const int leftin2 = 23;

const int leftpwm = 2;

const int rightin1 = 45;

const int rightin2 = 43 ;

const int rightpwm = 8;

const int fleftin1 = 37;

const int fleftin2 = 35;

const int fleftpwm = 5;

const int frightin1 = 33;

const int frightin2 = 31;

const int frightpwm = 4;

int piston = 41;

//#define piston A15

const int lpwm = 255, rpwm = 255, subtract = 155;

int x, y;

bool run = 0;

const int less = 20, more = 20;

int set = 0;

//void slow(void);

#include "hidjoystickrptparser.h"

JoystickReportParser::JoystickReportParser(JoystickEvents \*evt) :

joyEvents(evt),

oldHat(0xDE),

oldButtons(0) {

for (uint8\_t i = 0; i < RPT\_GEMEPAD\_LEN; i++)

oldPad[i] = 0xD;

}

void slow() {}

void JoystickReportParser::Parse(HID \*hid, bool is\_rpt\_id, uint8\_t len, uint8\_t \*buf) {

bool match = true;

// Checking if there are changes in report since the method was last called

for (uint8\_t i = 0; i < RPT\_GEMEPAD\_LEN; i++)

if (buf[i] != oldPad[i]) {

match = false;

break;

}

// Calling Game Pad event handler

if (!match && joyEvents) {

joyEvents->OnGamePadChanged((const GamePadEventData\*)buf);

for (uint8\_t i = 0; i < RPT\_GEMEPAD\_LEN; i++) oldPad[i] = buf[i];

}

uint8\_t hat = (buf[5] & 0xF);

// Calling Hat Switch event handler

if (hat != oldHat && joyEvents) {

joyEvents->OnHatSwitch(hat);

oldHat = hat;

}

uint16\_t buttons = (0x0000 | buf[6]);

buttons <<= 4;

buttons |= (buf[5] >> 4);

uint16\_t changes = (buttons ^ oldButtons);

// Calling Button Event Handler for every button changed

if (changes) {

for (uint8\_t i = 0; i < 0x0C; i++) {

uint16\_t mask = (0x0001 << i);

if (((mask & changes) > 0) && joyEvents)

if ((buttons & mask) > 0)

joyEvents->OnButtonDn(i + 1);

else

joyEvents->OnButtonUp(i + 1);

}

oldButtons = buttons;

}

}

void JoystickEvents::OnGamePadChanged(const GamePadEventData \*evt) {

/\* Serial.print("X1: ");

PrintHex<uint8\_t > (evt->X, 0x80);

Serial.print("\tY1: ");

PrintHex<uint8\_t > (evt->Y, 0x80);

Serial.print("\tX2: ");

PrintHex<uint8\_t > (evt->Z1, 0x80);

Serial.print("\tY2: ");

PrintHex<uint8\_t > (evt->Z2, 0x80);

Serial.print("\tRz: ");

PrintHex<uint8\_t > (evt->Rz, 0x80);

Serial.println("");

\*/

/\* if (evt->X == 127 && evt->Y == 127) {

run = 0;

}

else if (evt->X == 80 && evt->Y == 80) {

run = 1;

}

\*/

// if (run == 0) {

// Non analog code

if (evt->Y < 127 && evt->X == 127) { // FORWARD

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->Y > 127 && evt->X == 127) { // reverse run

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->X > 127 && evt->Y == 127) { // right side

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->X < 127 && evt->Y == 127) { // left side

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->X < 127 && evt->Y < 127) { // right diagonal back

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

}

if (evt->X > 127 && evt->Y < 127) { // right diagonal

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->X < 127 && evt->Y < 127) { // left diagonal

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

}

if (evt->X < 127 && evt->Y > 127) { // left diagonal back

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (evt->X == 127 && evt->Y == 127) { // to stop

/\* digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, HIGH);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, HIGH);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, HIGH);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, HIGH);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

\*/

for (int i = lpwm; i >= 0; i = i - less) {

analogWrite(leftpwm, i);

analogWrite(rightpwm, i);

analogWrite(fleftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

digitalWrite(leftin1, LOW);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, LOW);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, LOW);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, LOW);

digitalWrite(frightin2, LOW);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

}

// Non Analog Code end

// }

/\* if (run == 1) {

// Analog Code

x = evt->X; y = evt->Y;

Serial.println(x); Serial.println(y);

if (x > 75 && x < 85 && y > 75 && y < 85) { // to stop

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, HIGH);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, HIGH);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, HIGH);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, HIGH);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

}

// if (x > 0x80 y 0x00) { // forward

if(y < 0x80 && y > 0x00){

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(leftpwm, lpwm);

delay(17); // Delay to go straight

analogWrite(frightpwm, rpwm);

delay(33);

analogWrite(rightpwm, rpwm);

}

// if (x == 0x80 && y == 0xFF) { // reverse

if(y > 0x80 && y < 0xFF){

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(leftpwm, lpwm);

delay(17); // Delay to go straight

analogWrite(frightpwm, rpwm);

delay(33);

analogWrite(rightpwm, rpwm);

}

// if (x == 0xFF && y == 0x80) { // right

if(x < 0xFF && x > 0x80){

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(frightpwm, rpwm);

analogWrite(leftpwm, lpwm);

analogWrite(rightpwm, rpwm);

}

// if (x == 0x00 && y == 0x80) { // left

if(x < 0x80 && x > 0x00){

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(frightpwm, rpwm);

analogWrite(leftpwm, lpwm);

analogWrite(rightpwm, rpwm);

}

/\* if (x == 0xFF && y == 0xFF) { // right diagonal back

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

analogWrite(leftpwm, lpwm);

analogWrite(frightpwm, rpwm);

}

if (x == 0xFF && y == 0x00) { // right diagonal

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(rightpwm, rpwm);

}

if (x == 0x00 && y == 0x00) { // left diagonal

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

analogWrite(leftpwm, lpwm);

analogWrite(frightpwm, rpwm);

}

if (x == 0x00 && y == 0xFF) { // left diagonal back

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

analogWrite(fleftpwm, lpwm);

analogWrite(rightpwm, rpwm);

}

// Analog Code end

\*/

// }

}

void JoystickEvents::OnHatSwitch(uint8\_t hat) {

/\* Serial.print("Hat Switch: ");

PrintHex<uint8\_t > (hat, 0x80);

Serial.println("");

\*/

if (hat == 00) { // forward

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 04) { // reverse

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

// delay(17); // Delay to go straight

analogWrite(frightpwm, i);

// delay(33);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 02) { // right side

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 06) { // left

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 03) { // right diagonal back

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

}

if (hat == 01) { // right diagonal

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 07) { // left diagonal

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

}

if (hat == 05) { // left diagonal back

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (hat == 0x0F) { // to stop

/\* digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, HIGH);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, HIGH);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, HIGH);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, HIGH);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

\*/

for (int i = lpwm; i >= 0; i = i - less) {

analogWrite(leftpwm, i);

analogWrite(rightpwm, i);

analogWrite(fleftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

digitalWrite(leftin1, LOW);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, LOW);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, LOW);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, LOW);

digitalWrite(frightin2, LOW);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

}

}

void JoystickEvents::OnButtonUp(uint8\_t but\_id) {

/\* Serial.print("Up: ");

Serial.println(but\_id, DEC);

\*/

if(but\_id == 9){

set = 0;

}

if ( but\_id == 2 || but\_id == 4 || but\_id == 7 || but\_id == 8) { // To stop

/\* digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, HIGH);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, HIGH);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, HIGH);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, HIGH);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

\*/

for (int i = lpwm; i >= 0; i = i - less) {

analogWrite(leftpwm, i);

analogWrite(rightpwm, i);

analogWrite(fleftpwm, i);

analogWrite(frightpwm, i);

delay(10);

}

digitalWrite(leftin1, LOW);

digitalWrite(leftin2, LOW);

digitalWrite(rightin1, LOW);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin1, LOW);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin1, LOW);

digitalWrite(frightin2, LOW);

analogWrite(leftpwm, 0);

analogWrite(rightpwm, 0);

analogWrite(fleftpwm, 0);

analogWrite(frightpwm, 0);

}

}

void JoystickEvents::OnButtonDn(uint8\_t but\_id) {

/\* Serial.print("Dn: ");

Serial.println(but\_id, DEC);

\*/

if (but\_id == 7) { // right rotate

digitalWrite(leftin2, HIGH);

digitalWrite(leftin1, LOW);

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(fleftin2, HIGH);

digitalWrite(fleftin1, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= (lpwm - subtract); i = i + (2 \* more)) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (but\_id == 8) { // left rotate

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(rightin2, HIGH);

digitalWrite(rightin1, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

digitalWrite(frightin2, HIGH);

digitalWrite(frightin1, LOW);

for (int i = 0; i <= (lpwm - subtract); i = i + (2 \* more)) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

analogWrite(frightpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (but\_id == 2) { // right turn

digitalWrite(leftin1, HIGH);

digitalWrite(leftin2, LOW);

digitalWrite(fleftin1, HIGH);

digitalWrite(fleftin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(fleftpwm, i);

analogWrite(leftpwm, i);

delay(10);

}

}

if (but\_id == 4) { // left turn

digitalWrite(rightin1, HIGH);

digitalWrite(rightin2, LOW);

digitalWrite(frightin1, HIGH);

digitalWrite(frightin2, LOW);

for (int i = 0; i <= lpwm; i = i + more) {

analogWrite(frightpwm, i);

analogWrite(rightpwm, i);

delay(10);

}

}

if (but\_id == 9) {

if (set == 0) {

Serial.println("Runned");

// delay(1000);

set = 1;

analogWrite(piston, 0);

delay(/\*3000\*/3500);

analogWrite(piston, 255);

}

}

}

Hidjoystickparser.h

#if !defined(\_\_HIDJOYSTICKRPTPARSER\_H\_\_)

#define \_\_HIDJOYSTICKRPTPARSER\_H\_\_

#include <hid.h>

struct GamePadEventData {

uint8\_t X, Y, Z1, Z2, Rz;

};

class JoystickEvents {

public:

virtual void OnGamePadChanged(const GamePadEventData \*evt);

virtual void OnHatSwitch(uint8\_t hat);

virtual void OnButtonUp(uint8\_t but\_id);

virtual void OnButtonDn(uint8\_t but\_id);

// void slow();

};

#define RPT\_GEMEPAD\_LEN 5

class JoystickReportParser : public HIDReportParser {

JoystickEvents \*joyEvents;

uint8\_t oldPad[RPT\_GEMEPAD\_LEN];

uint8\_t oldHat;

uint16\_t oldButtons;

public:

JoystickReportParser(JoystickEvents \*evt);

void slow();

virtual void Parse(HID \*hid, bool is\_rpt\_id, uint8\_t len, uint8\_t \*buf);

};

#endif // \_\_HIDJOYSTICKRPTPARSER\_H\_\_