Fernando Cuadra Silva 23/06/2017

Arquitectura de Microprocesadores

Joystick

Índice

Estado del Arte

Una palanca de mando o joystick (del inglés joy, alegría, y stick, palo) es un dispositivo de control de dos o tres ejes que se usa desde una computadora o videoconsola hasta un transbordador espacial, los nuevos aviones de transporte como el Airbus A320 y los nuevos diseños de aviones de caza, aunque en el caso de los Airbus se llama sidestick, pasando por grúas de carga y porta contenedores, también existen nuevos tractores y máquinas pesadas, que tienen funciones especiales controlados por computadora.

Se suele diferenciar entre joysticks digitales (que leen cuatro interruptores encendido/apagado en cruceta situada en la base más sus combinaciones y los botones de acción) y los analógicos (que usan potenciómetros para leer continuamente el estado de cada eje, y además de botones de acción pueden incorporar controles deslizantes), siendo estos últimos más precisos.

Los joystick se utilizaban originalmente para controlar los alerones y el plano de profundidad de una aeronave experimental. El invento parece deberse al piloto francés de principios del siglo XX Robert Dudau (radio manele), aunque el primer uso del término es de 1910 cuando el piloto Robert Loraine se refiere a joy-stick como sinónimo de cloche, el término francés utilizado mayormente en la época.

El primer joystick eléctrico de dos ejes probablemente fue inventado en 1944 en Alemania, durante la Segunda Guerra Mundial. Se desarrolló para controlar la bomba guiada Henschel Hs 293, lanzada desde un avión bombardero. El joystick era utilizado por el operador del arma, sentado en el avión bombardero para dirigir el misil hacia su blanco por control de radio. El joystick constaba de interruptores encendido/apagado en lugar de sensores analógicos, por lo que se le podría considerar el primer joystick digital. La señal se transmitía al misil mediante un cable fino, que se desenrollaba cuando el misil era lanzado.

Esta idea fue aprovechada por los científicos del Heeresversuchsanstalt en Peenemünde para el desarrollo de la bomba voladora. Una parte del equipo del programa alemán de cohetes desarrollaba el misil Wasserfall, sucesor del cohete V-2, el primer misil tierra-aire diseñado para derribar aviones de combate enemigos.

El equipo de desarrollo del Wasserfall modificó el sistema de control para convertir la señal eléctrica a señales de radio que se transmitían al misil, eliminando la necesidad del cable.

Los primeros joystick de máquina recreativa de salón, o máquina arcade, eran joysticks digitales porque el estándar de conexión de las placas de circuitos de estas máquinas mayoritariamente usado, llamado Jamma, que conecta a los diferentes periféricos de la carcasa (monitor, botonera, ranura para monedas...) solo detecta pulsaciones abierto/cerrado, con unos interruptores normales instalados en la base de la palanca, por lo cual los joystick para juegos deben ser de este tipo. Al evolucionar las recreativas a la par

que los ordenadores y videoconsolas, comenzaron a aparecer controles de tipo analógico y luego los digitales.

Joystick con un sólo botón, de los años 1980.

Las primeras consolas Pong usaban potenciómetros pero la videoconsola Atari 2600 estableció lo que sería el estándar mayoritariamente usado (con variaciones) de joystick digital de dos ejes más un botón de fuego, combinado con una pareja de potenciómetros (para usar con paddles/mouse/trackball).

Por el otro lado, el Apple II introdujo el joystick analógico con conector de 9 pines, que fue adoptado por el IBM PC pero con conector de 15 pines. Sólo unos pocos ordenadores de 8 bits utilizaron su propia variación del joystick analógico. Por su parte Nintendo introdujo con su videoconsola Nintendo Entertainment System la primera interfaz propietaria con señales multiplexadas, que con variaciones propietarias en cada consola y generación ha sido el sistema utilizado en las videoconsolas hasta la aparición del USB.

Apple utilizó la interfaz Apple Desktop Bus como método de conexión de joysticks analógicos y digitales y gamepads en los Apple Macintosh (notablemente caros por las pocas cantidades fabricadas), hasta que la aparición del iMac marcó el paso a USB.

Descripción del circuito

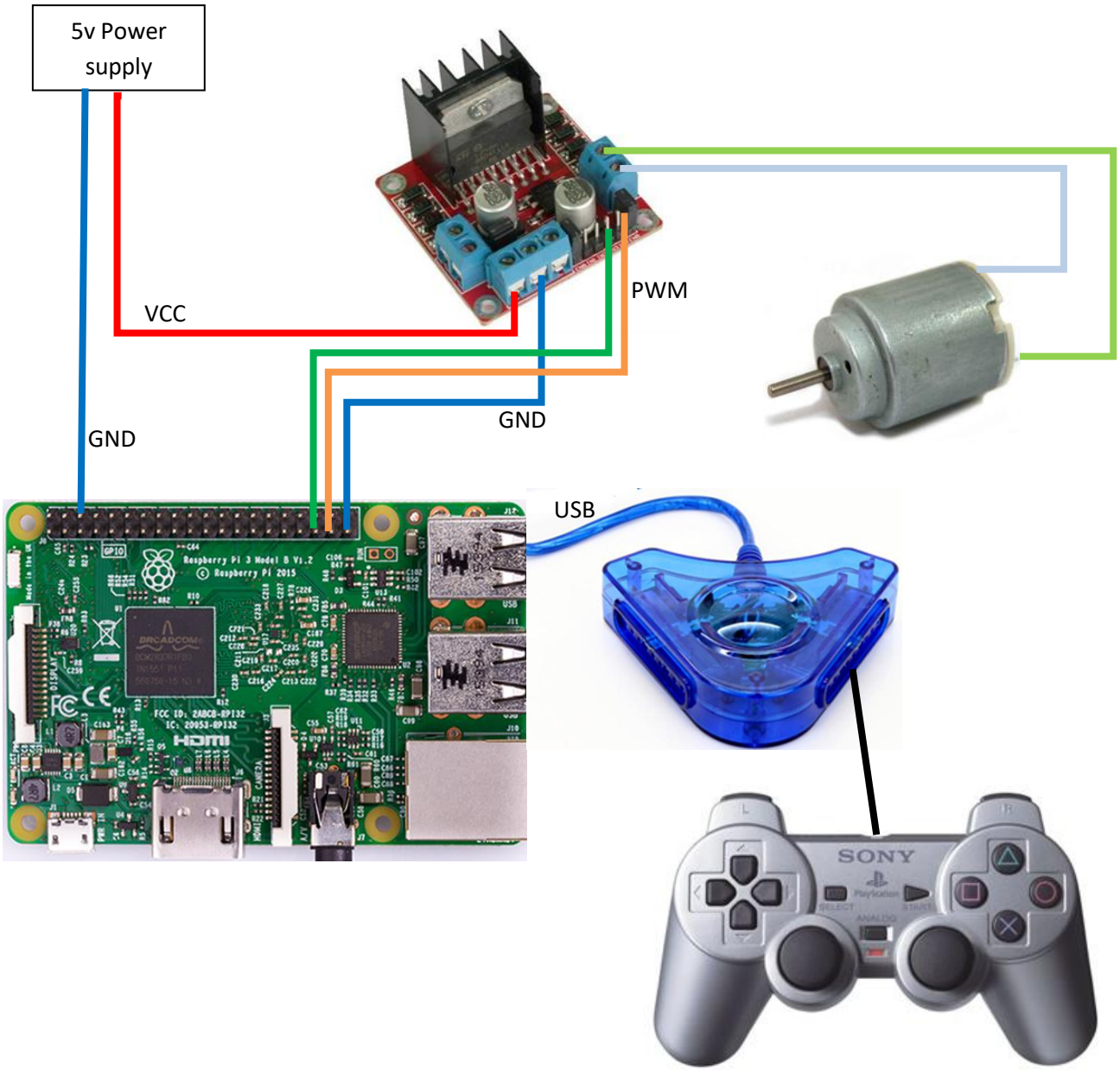
El circuito sirve de ejemplo de uso de un joystick y algunas posibles aplicaciones que se le puede dar. El circuito consta de 1 entrada (Joystick USB) y 2 salidas PWM.

Los valores del Joystick de entrada se leen mediante el uso del archivo: “/dev/input/js0”. En este archivo se va publicando los eventos que genera el control. Después de leer el archivo, se interpreta y se traduce a comandos como pueden ser un botón pulsado ó un movimiento de los ejes x y.

Al mover el joystick analógico izquierdo se mueve el cursor de la raspberry pi a la posición indicada. También al presionar las flechas de dirección el cursor se mueve a esa nueva posición. Al presionar el botón “X” se dará un clic en la posición actual del cursor. Si se presiona el botón “cuadrado” se dará un doble clic en la posición actual del cursor. Si se presiona el botón “start” el cursor se moverá automáticamente a la posición del botón de inicio y le dará clic automáticamente, abriendo el menú del botón de inicio. Si se presiona el botón “triángulo” la raspberry pi se pone en espera de una conexión blue tooth. A conectarse con un dispositivo movil se puede controlar el cursor remotamente. Si se presiona el botón “select” Se saldrá del programa.

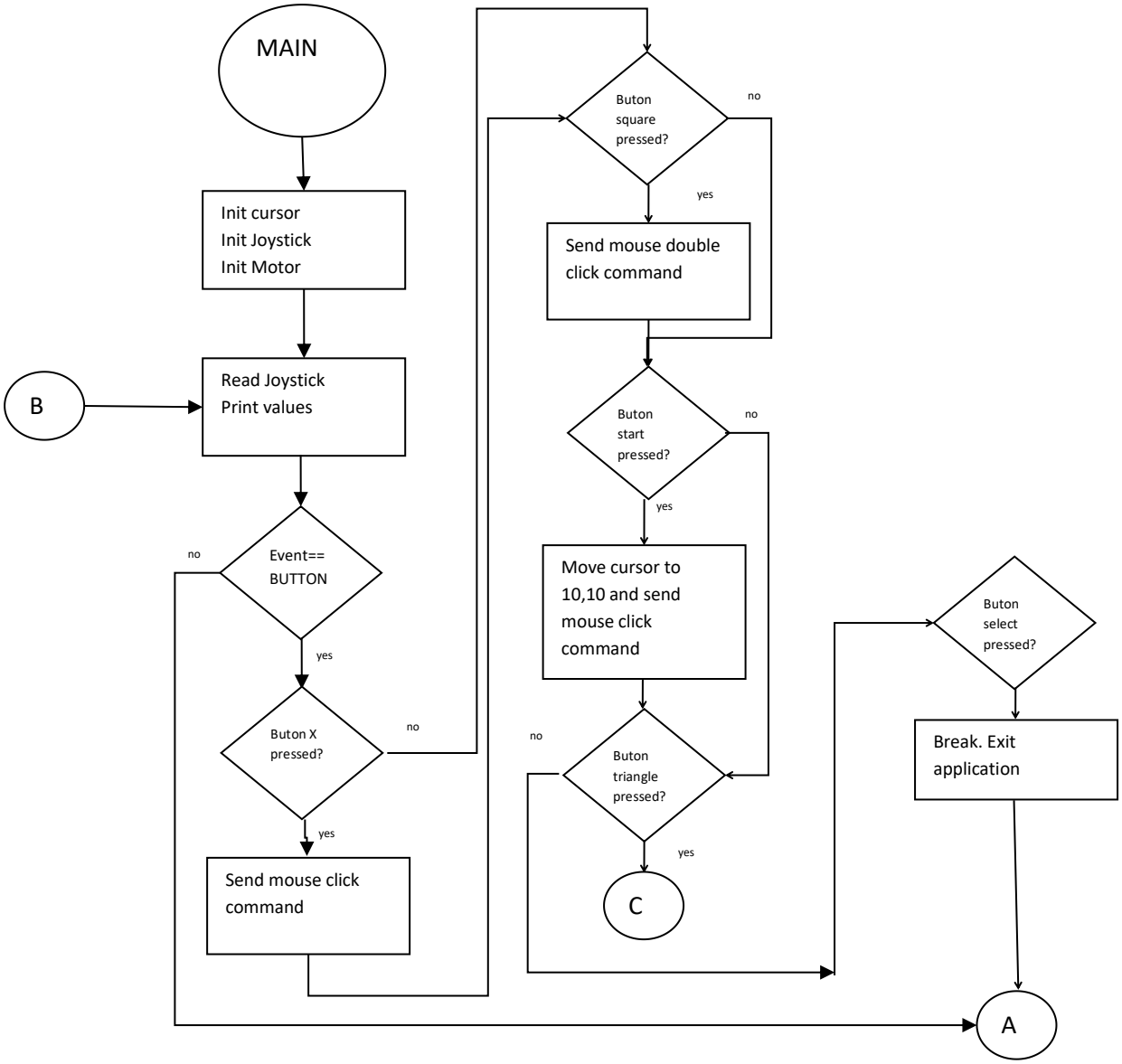
La raspberry pi tiene conectado un puente H y un motor de corriente directa. Al mover el joystick analógico derecho a la izquierda o a la derecha, se manda PWM a los pines GPIO para que el motor de corriente directa gire hacia un sentido ó hacia el otro con la velocidad de acuerdo a la posición de la palanca.

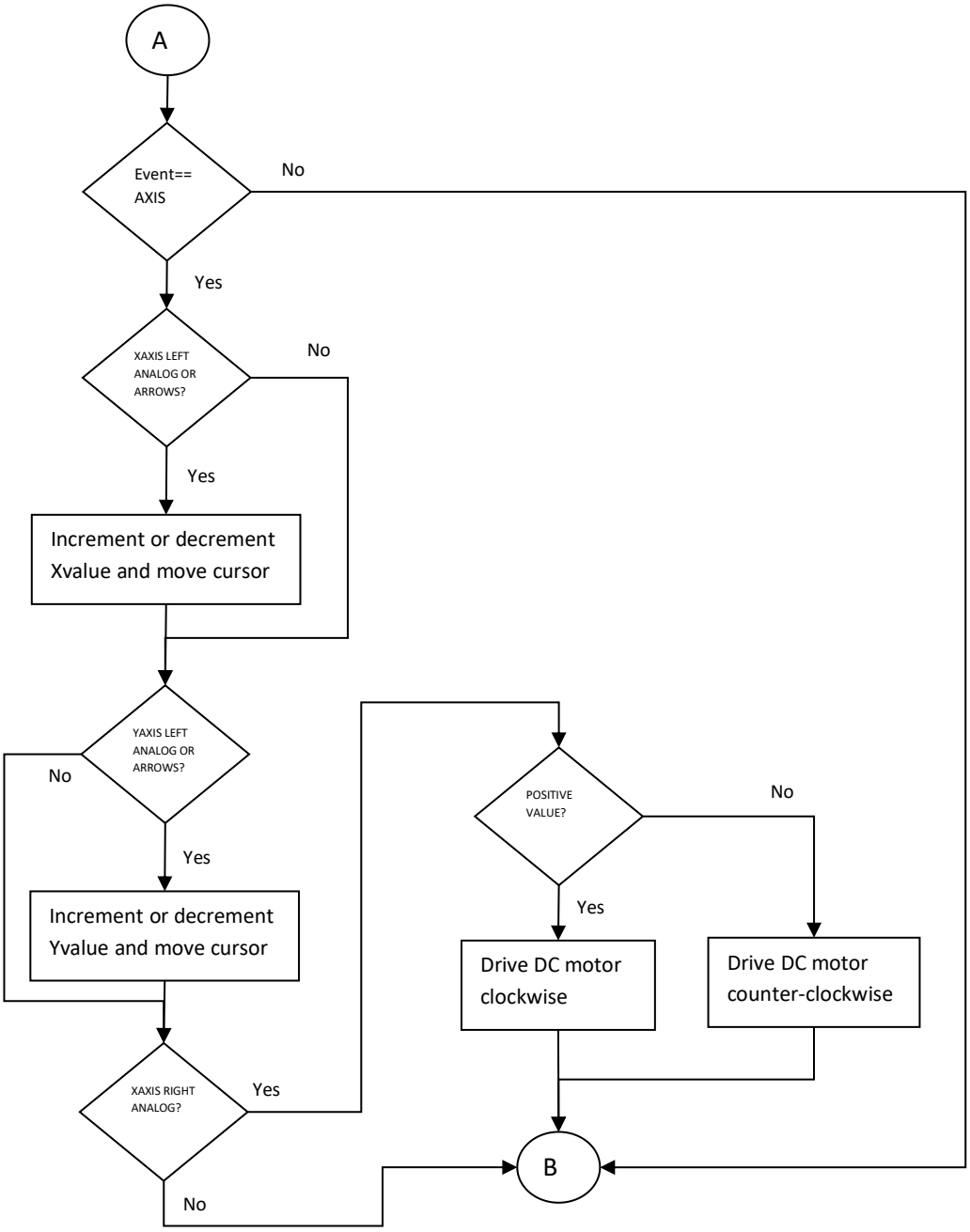
Diagrama de bloques del circuito

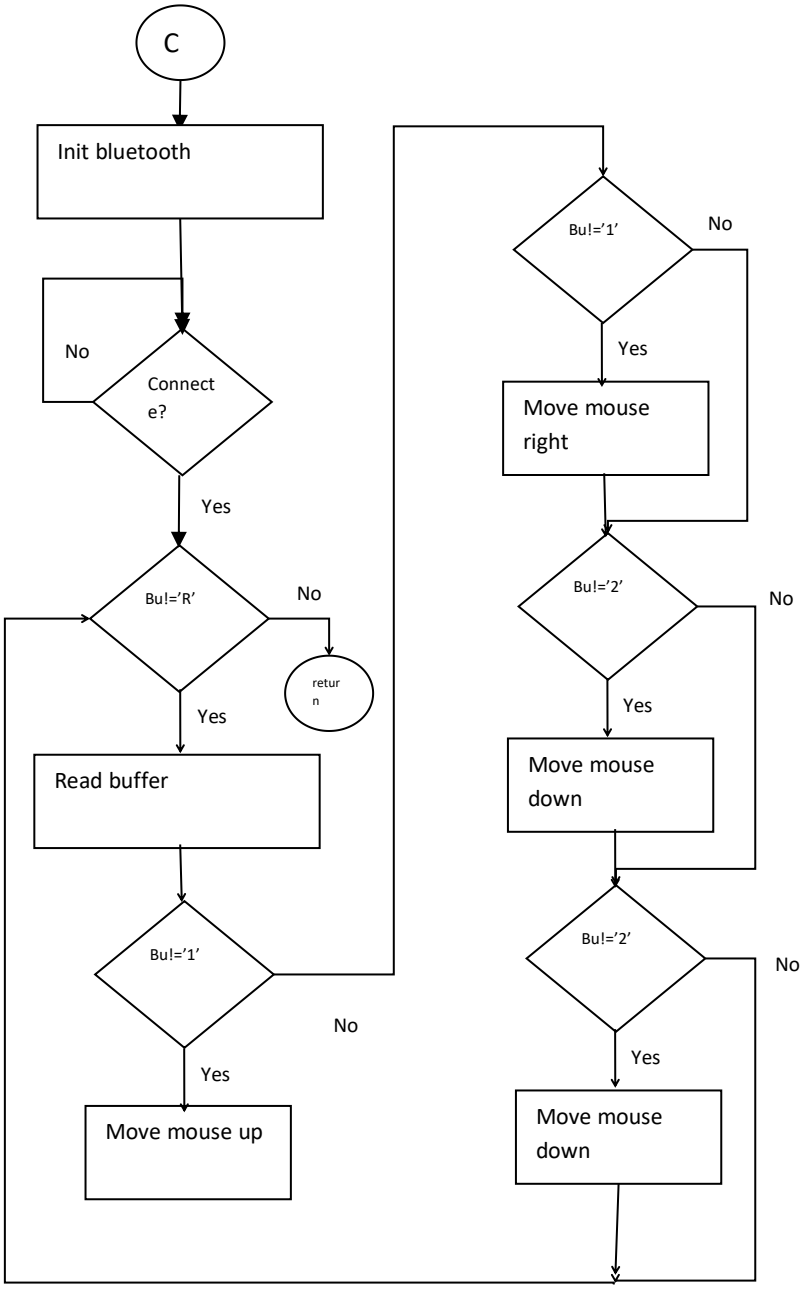


Se creó una librería para manipular el cursor “cursorClass.h” y se creó otra librería para leer los valores del joystick “joystickClass.h”. También se está utilizando la librería pigpio.h para generar los PWMs para mover el motor de corriente directa.

El siguiente es un diagrama de flujo de la aplicación:

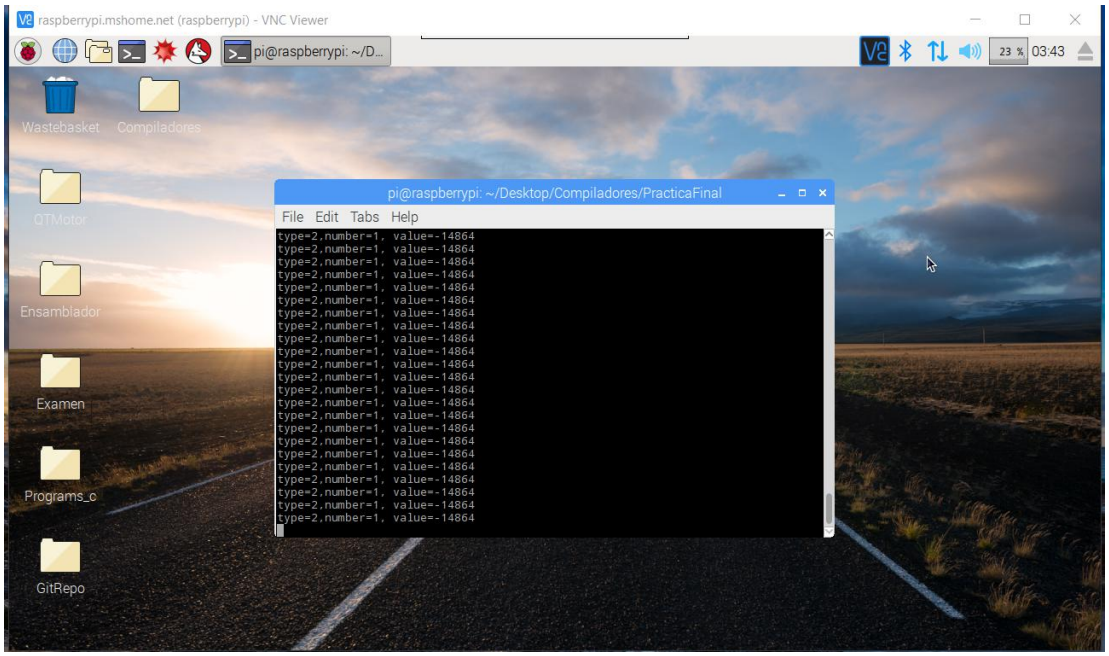



A
Event== AXIS
No
Yes
XAXIS LEFT ANALOG OR ARROWS?
No
Yes
Increment or decrement Xvalue and move cursor
YAXIS LEFT ANALOG OR ARROWS?
No
Yes
Increment or decrement Yvalue and move cursor
XAXIS RIGHT ANALOG?
Yes
No
POSITIVE VALUE?
No
Yes
Drive DC motor clockwise
Drive DC motor counter-clockwise
B

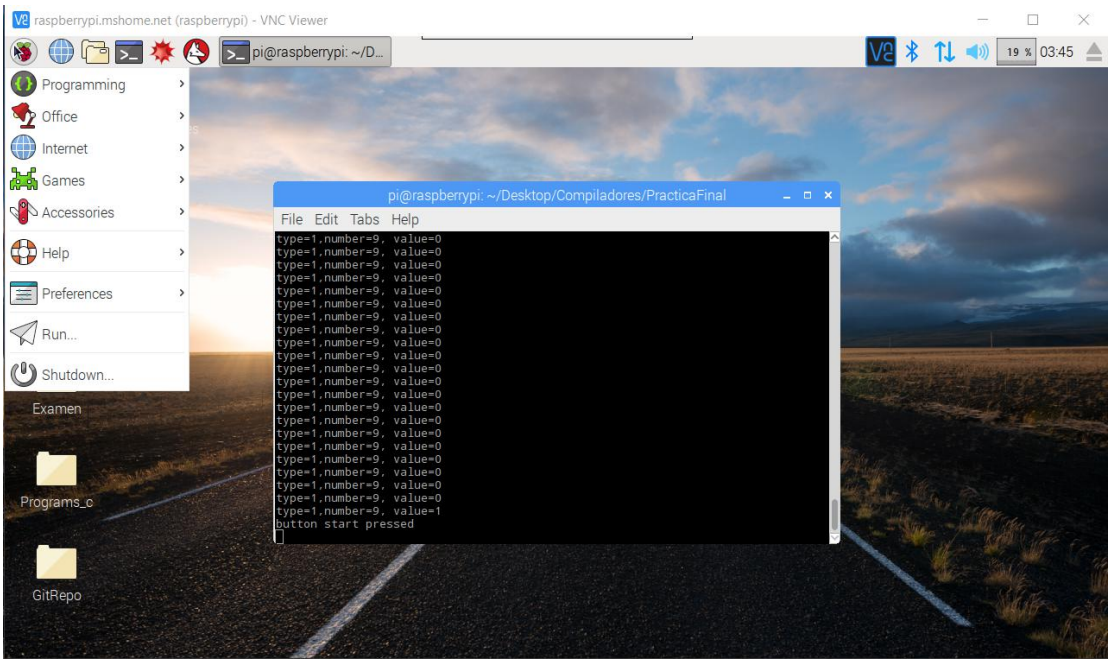

C
Init bluetooth
No
Connect e?
Yes
Bu!='R'
No
retur n
Yes
Read buffer
Bu!='1'
No
Yes
Move mouse up
Bu!='1'
No
Yes
Move mouse right
Bu!='2'
No
Yes
Move mouse down
Bu!='2'
No
Yes
Move mouse down

Resultados

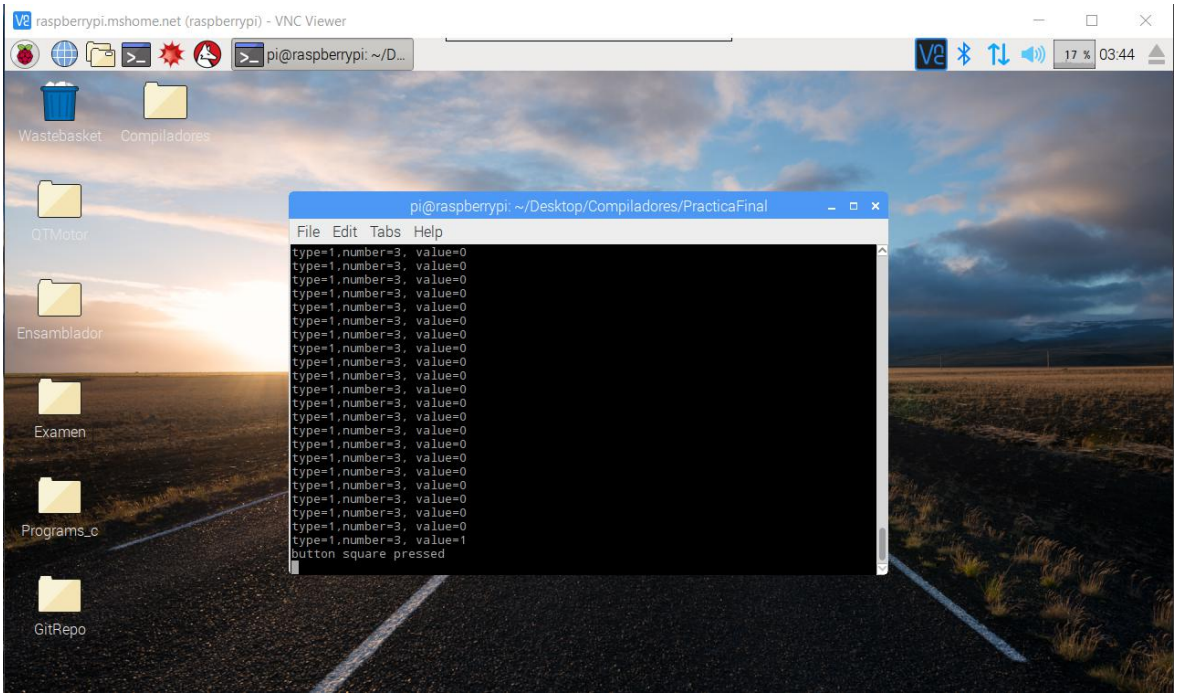
Al mover el joystick analógico derecho el motor gira en la dirección adecuada y a la velocidad correcta.



Al mover el joystick analógico derecho ó las flechas, el cursor se mueve a la posición correcta.



Al presionar el botón “start” el cursor se coloca correctamente en la posición (10,10) y da clic automáticamente para abrir el menú.



Al presionar el botón cuadrado, x, start, select se muestra en la pantalla qué botón se presionó.

Apéndice 1. Código

main (main_app.cpp):

```
/*
 * before compile visit:
      http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick
 *    this has to be done before using this code:
 *    sudo apt-get install joystick
 *  sudo jstest /dev/input/js0
 *  ls -l /dev/input/js0
 *  sudo chmod a+rw /dev/input/js0
 *
 */

#include "drivers/cursor/cursorClass.h"
#include "hal/joystick/joystickClass.h"
#include <time.h>
#include <pigpio.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
#include <stdio.h>
#include <unistd.h>


#define GPIOIZQ 26
#define GPIODER 19
#define PWMFREQ 60

cursorClass* MousePtr;

using namespace std;
int i=100;
js_event jsread;
JoystickClass *js;
int xtemp=100, ytemp=100;

void InitializeMotor(void);
void Bluetooth();


int main()
{
      unsigned int filter=0, xprevInc=0, yprevInc=0;
      unsigned char xbuttonprev=0,squarebuttonprev=0, startbuttonprev=0,
trianglebuttonprev=0;
      MousePtr=new cursorClass();
      js=new JoystickClass();
      InitializeMotor();
      printf("\nBegin\n");
      sleep(1);
```

```
      while(1)
      {
            jsread=js->EventRead();
            printf("type=%d,number=%d,
value=%d\n",jsread.type,jsread.number,jsread.value);    // value
            if ((jsread.type&JS_EVENT_BUTTON)== JS_EVENT_BUTTON)
            {
                  if(jsread.number==X_BUTTON)
                  {
                        if(jsread.value==1 && xbuttonprev==0)
                        {
                              printf("button x pressed\n");
                              MousePtr->Click();
                        }
                        xbuttonprev=jsread.value;
                  }
                  else if(jsread.number== SQUARE_BUTTON)
                  {
                        if(jsread.value==1 && squarebuttonprev==0)
                        {
                              printf("button square pressed\n");
                              MousePtr->DoubleClick();
                        }
                        squarebuttonprev=jsread.value;
                  }
                  else if(jsread.number== START_BUTTON)
                  {
                        if(jsread.value==1 && startbuttonprev==0)
                        {
                              printf("button start pressed\n");
                              MousePtr->MoveCursor(10,10);
                              MousePtr->Click();
                        }
                        startbuttonprev=jsread.value;
                  }
                  else if(jsread.number== TRIANGLE_BUTTON)
                  {
                        if(jsread.value==1 && trianglebuttonprev==0)
                        {
                              printf("button triangle pressed\nBluetooth\n");
                              Bluetooth();
                        }
                        trianglebuttonprev=jsread.value;
                  }
                  else if(jsread.number== SELECT_BUTTON && jsread.value==1)
                  {
                        printf("button select pressed\n");
                        break;      //exits from application
                  }

            }
            if((jsread.type&JS_EVENT_AXIS)== JS_EVENT_AXIS)
            {
                  if(jsread.number==XAXIS1 || jsread.number==XAXIS3)
                  {
                        if((filter++)%10==0)
                        {
```

```
                              xtemp=xtemp+(jsread.value/3276); //to get max increments
of 10 by 10 pixels
                              if(xtemp<0){xtemp=0;}
                              if(xtemp>MousePtr->maxwidth){xtemp=MousePtr->maxwidth;}
                              ytemp=ytemp+yprevInc;
                              if(ytemp<0){ytemp=0;}
                              if(ytemp>MousePtr->maxheight){ytemp=MousePtr->maxheight;}
                              MousePtr->MoveCursor(xtemp,ytemp);
                              xprevInc=jsread.value/3276;
                        }
                        if(jsread.value=0){xprevInc=0;}
                  }
                  if(jsread.number==YAXIS1 || jsread.number==YAXIS3)
                  {
                        if((filter++)%10==0)
                        {
                              ytemp=ytemp+(jsread.value/3276); //to get max increments
of 10 by 10 pixels
                              if(ytemp<0){ytemp=0;}
                              if(ytemp>MousePtr->maxheight){ytemp=MousePtr->maxheight;}
                              xtemp=xtemp+xprevInc;
                              if(xtemp<0){xtemp=0;}
                              if(xtemp>MousePtr->maxwidth){xtemp=MousePtr->maxwidth;}
                              MousePtr->MoveCursor(xtemp,ytemp);
                              yprevInc=jsread.value/3276;
                        }
                        if(jsread.value=0){yprevInc=0;}
                  }

                  if(jsread.number==XAXIS2 )//moving DC motor
                  {
                        if(jsread.value>0)
                        {
                              gpioPWM(GPIOIZQ,0);
                              gpioPWM(GPIODER,jsread.value);
                        }
                        else if (jsread.value<0)
                        {
                              gpioPWM(GPIOIZQ,(-1*jsread.value));
                              gpioPWM(GPIODER,0);
                        }
                  }
                  if(jsread.number==YAXIS1 || jsread.number==YAXIS3)
                  {
                        //do nothing
                  }

            }
            //sleep(1);
            usleep(1000); //wait 1ms
            fflush(stdout);
      }
      delete js;
      gpioTerminate();
      return 0;
}
```

```
void InitializeMotor(void)
{
	gpioInitialise();
	gpioSetPWMfrequency(GPIOIZQ,PWMFREQ);
	gpioSetPWMfrequency(GPIODER,PWMFREQ);
	gpioSetPWMrange(GPIOIZQ,32767);
	gpioSetPWMrange(GPIODER,32767);
	gpioPWM(GPIOIZQ,0);
	gpioPWM(GPIODER,0);
}


void Bluetooth()
{
	bdaddr_t my_bdaddress={0};
	struct sockaddr_rc loc_addr = { 0 }, rem_addr = { 0 };
	char buf[1024] = { 0 };
	int s, client, bytes_read;
	socklen_t opt = sizeof(rem_addr);

	// allocate socket
	s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

	// bind socket to port 1 of the first available
	// local bluetooth adapter
	loc_addr.rc_family = AF_BLUETOOTH;
	//loc_addr.rc_bdaddr = *BDADDR_ANY;
	loc_addr.rc_bdaddr = my_bdaddress;
	loc_addr.rc_channel = (uint8_t) 1;
	bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));

	// put socket into listening mode
	listen(s, 1);

	// accept one connection
	client = accept(s, (struct sockaddr *)&rem_addr, &opt);

	ba2str( &rem_addr.rc_bdaddr, buf );
	fprintf(stderr, "accepted connection from %s\n", buf);
	memset(buf, 0, sizeof(buf));

	// read data from the client
	while (!strstr(buf,"R"))
	{
		bytes_read = read(client, buf, sizeof(buf));
		if( bytes_read > 0 ) {
			printf("[%s]\n", buf);
			if(strchr(buf,'1'))
			{
				ytemp=ytemp-10;
				if(ytemp<0){ytemp=0;}
				if(ytemp>MousePtr->maxheight){ytemp=MousePtr->maxheight;}
				MousePtr->MoveCursor(xtemp,ytemp);
			}
			if(strchr(buf,'2'))
```

```
                {
                        xtemp=xtemp+10;
                        if(xtemp<0){xtemp=0;}
                        if(xtemp>MousePtr->maxwidth){xtemp=MousePtr->maxwidth;}
                        MousePtr->MoveCursor(xtemp,ytemp);
                }
                if(strchr(buf,'3'))
                {
                        ytemp=ytemp+10;
                        if(ytemp<0){ytemp=0;}
                        if(ytemp>MousePtr->maxheight){ytemp=MousePtr->maxheight;}
                        MousePtr->MoveCursor(xtemp,ytemp);
                }
                if(strchr(buf,'4'))
                {
                        xtemp=xtemp-10;
                        if(xtemp<0){xtemp=0;}
                        if(xtemp>MousePtr->maxwidth){xtemp=MousePtr->maxwidth;}
                        MousePtr->MoveCursor(xtemp,ytemp);
                }
        }


      }
      // close connection
      close(client);
      close(s);
}
```

cursorClass (cursorClass.cpp):

```
#include "cursorClass.h"
#include <unistd.h>
#include <string.h>
#include <stdio.h>

 cursorClass::cursorClass()
 {
       XWindowAttributes window_attributes_return;
       //Get system window
       dpy = XOpenDisplay(0);
       root_window = XRootWindow(dpy, 0);
       XSelectInput(dpy, root_window, KeyReleaseMask);

       XGetWindowAttributes(dpy, root_window, &window_attributes_return);
       maxwidth=window_attributes_return.width;
       maxheight=window_attributes_return.height;
       printf("max width %d, ",maxwidth);
       printf("max height %d. ",maxheight);
 }
 cursorClass::~cursorClass()
 {
       dpy=0;
```

```
	root_window=0;
}
void cursorClass::MoveCursor(int x,int y)
{
	if(dpy!=0 && root_window!=0)
	{
		XWarpPointer(dpy, None, root_window, 0, 0, 0, 0, x, y);
		XFlush(dpy);	//update cursor's position
	}
}
void cursorClass::Click()
{
	XEvent event;
	int button=1;

	if(dpy != 0)
	{
		memset(&event, 0x00, sizeof(event));
		event.type = ButtonPress;
		event.xbutton.button = button;
		event.xbutton.same_screen = True;
		XQueryPointer(dpy, root_window, &event.xbutton.root,
&event.xbutton.window, &event.xbutton.x_root, &event.xbutton.y_root, &event.xbutton.x,
&event.xbutton.y, &event.xbutton.state);
		event.xbutton.subwindow = event.xbutton.window;
		while(event.xbutton.subwindow)
		{
			event.xbutton.window = event.xbutton.subwindow;
			XQueryPointer(dpy, event.xbutton.window, &event.xbutton.root,
&event.xbutton.subwindow, &event.xbutton.x_root, &event.xbutton.y_root,
&event.xbutton.x, &event.xbutton.y, &event.xbutton.state);
		}
		XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
		XFlush(dpy);
		usleep(100000);
		event.type = ButtonRelease;
		event.xbutton.state = 0x100;
		XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
		XFlush(dpy);
	}
}
void cursorClass::DoubleClick()
{
	XEvent event;
	int button=1;

	if(dpy != 0)
	{
		memset(&event, 0x00, sizeof(event));
		event.type = ButtonPress;
		event.xbutton.button = button;
		event.xbutton.same_screen = True;
		XQueryPointer(dpy, root_window, &event.xbutton.root,
&event.xbutton.window, &event.xbutton.x_root, &event.xbutton.y_root, &event.xbutton.x,
&event.xbutton.y, &event.xbutton.state);
		event.xbutton.subwindow = event.xbutton.window;
		while(event.xbutton.subwindow)
```

```
            {
                  event.xbutton.window = event.xbutton.subwindow;
                  XQueryPointer(dpy, event.xbutton.window, &event.xbutton.root,
&event.xbutton.subwindow, &event.xbutton.x_root, &event.xbutton.y_root,
&event.xbutton.x, &event.xbutton.y, &event.xbutton.state);
            }
            XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
            XFlush(dpy);
            usleep(100000);
            event.type = ButtonRelease;
            event.xbutton.state = 0x100;
            XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
            XFlush(dpy);
            usleep(100000);
            event.type = ButtonPress;
            event.xbutton.state = 0x100;
            XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
            XFlush(dpy);
            usleep(100000);
            event.type = ButtonRelease;
            event.xbutton.state = 0x100;
            XSendEvent(dpy, event.xbutton.window, True, 0xfff, &event);
            XFlush(dpy);
      }
 }
```

cursorClass (cursorClass.h):

```
/*************************************************************************
 * cursor
 * ************************************************************************/
#ifndef CURSOR_H
    #define CURSOR_H


#include <X11/X.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>

class cursorClass{

public:
      cursorClass();
    ~cursorClass();
    void MoveCursor(int x,int y);
    void Click();
    void DoubleClick();
    int maxwidth;
    int maxheight;
private:
```

```
    Display *dpy;
    Window root_window;
};

#endif
```

joystickClass (joystickClass.cpp):

```
#include "joystickClass.h"


JoystickClass::JoystickClass()
{
      fd = open ("/dev/input/js0", O_RDONLY);
      ioctl (fd, JSIOCGAXES, &number_of_axes);
      printf("%d",number_of_axes);
      sleep(1);
      //FD_ZERO(&rfds);
    //FD_SET(fd, &rfds);
      //tv.tv_sec = 0;
    //tv.tv_usec = 10000; //wait 10ms per read
      //select(1, &rfds, NULL, NULL, &tv);
      fcntl(fd, F_SETFL, O_NONBLOCK );    /* use non-blocking mode */
}
JoystickClass::~JoystickClass()
{
      close (fd);
}

js_event JoystickClass::EventRead()
{
      read (fd, &e, sizeof(e));
      return e;
}
```

joystickClass (joystickClass.h):

```
#ifndef JOYSTICKCLASS_H
#define JOYSTICKCLASS_H

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <iostream>
#include <string>
#include <sstream>
```

```
#include <sys/ioctl.h>
#include <linux/joystick.h>

enum BUTTONNAMES{
      TRIANGLE_BUTTON=0,
      CIRCLE_BUTTON=1,
      X_BUTTON=2,
      SQUARE_BUTTON=3,
      L2_BUTTON=4,
      R2_BUTTON=5,
      L1_BUTTON=6,
      R1_BUTTON=7,
      SELECT_BUTTON=8,
      START_BUTTON=9,
      LEFTJOY_BUTTON=10,
      RIGTHJOY_BUTTON=11
};

enum Axis
{
      XAXIS1=0,
      YAXIS1=1,
      YAXIS2=2,
      XAXIS2=3,
      XAXIS3=4,
      YAXIS3=5
};



class JoystickClass
{
public:
      JoystickClass();
      ~JoystickClass();
      js_event EventRead();

      js_event e;
      char number_of_axes;
private:
    int fd;
    //fd_set rfds;
      struct timeval tv;
      static const short MIN_AXES_VALUE = -32768;
    static const short MAX_AXES_VALUE = 32767;
};

#endif
```

makefile(makefile):

```
build: pre compile asm linker run clean

pre:
	cpp main_app.cpp > main_app.i
	cpp drivers/cursor/cursorClass.cpp > cursorClass.i
	cpp hal/gpio/GPIOClass.cpp > GPIOClass.i
	cpp hal/joystick/joystickClass.cpp > joystickClass.i
compile:
	g++ -S main_app.i
	g++ -S -c cursorClass.i
	g++ -S -c GPIOClass.i
	g++ -S -c joystickClass.i

asm:
	as -o main_app.o main_app.s
	as -o cursorClass.o cursorClass.s
	as -o GPIOClass.o GPIOClass.s
	as -o joystickClass.o joystickClass.s
linker:
	g++ -lX11 -lpigpio -lrt -lpthread -Wall -lbluetooth -o Joystick.exe main_app.o
cursorClass.o GPIOClass.o joystickClass.o
run:
	./Joystick.exe
clean:
	rm *.i *.s *.o Joystick.exe
```

Referencias

https://es.wikipedia.org/wiki/Palanca_de_mando
http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick
https://www.kernel.org/doc/Documentation/input/joystick-api.txt