



Memoria Diseño PCB



Angel Lopez Santiago

Contenido

Objetivo	2
Materiales	2
Diseño de la PCB.....	3
Esquemático	3
Creación de la board.	4
Validación de diseño de la PCB	4
Ensamblado de componentes.....	5
Haciendo uso de la PCB.....	6
Empaquetando la información.....	6
Programando arduino	7
Programando NUCLEO	8
Procesando datos en la aplicación C#	9
Recepción de datos	9
Traducción de datos	10
Funcionamiento modo ratón	10
Modo Gamepad	11
Ventajas e inconvenientes de usar la aplicación C#.....	12

Objetivo

Nuestro objetivo es diseñar una PCB compatible con Arduino y Nucleo para funcionar como gamepad.

Materiales

Los materiales de los que disponemos son:

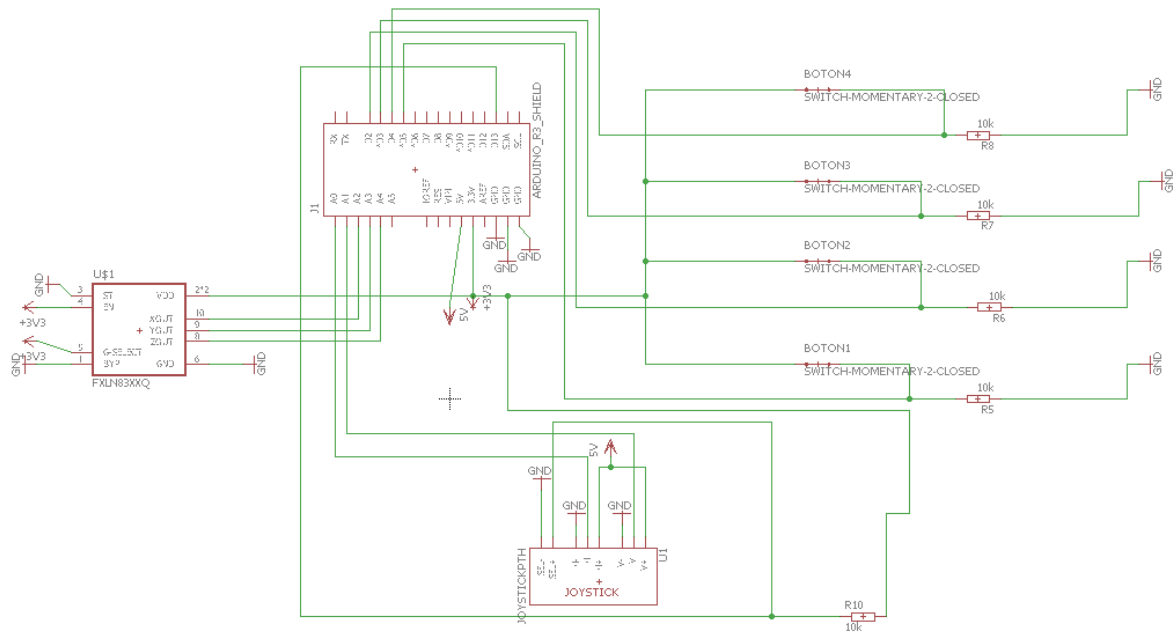
- 1 x Arduino UNO R3.
- 1 x Stm32 NUCLEO F401RE.
- 1 x Joystick Thumb Joystick Sparkfun.
- 4 x Pulsadores Momentary Pushbutton Switch - 12mm Square.
- 5 x Resistencias de 10K y 330oh: SMD 1206.
- 1 x Acelerometro 3 ejes [FXLN8372QR1](#) (Pulsa el hipervínculo para acceder a la data sheet).
- Nuestra PCB.
- Soldador.
- Estaño.
- Flux.
- Pasta de soldadura.

Diseño de la PCB

Ya que la distribución de pines de Arduino y NUCLEO son compatibles, sólo necesitaremos hacer un diseño.

Para hacer dicho diseño utilizaremos la herramienta de diseño de PCB EAGLE

Esquemático



Los 4 pulsadores irán conectados a los pines digitales 2-5 y el pulsador del joystick al pin 13.

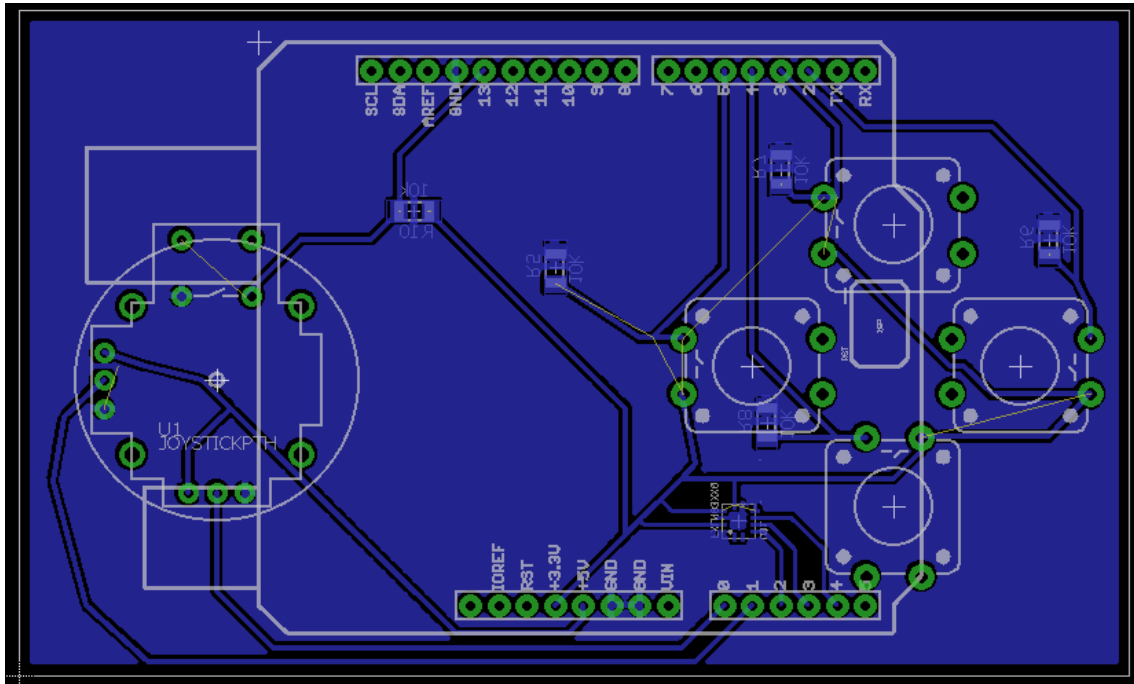
Como los pulsadores van conectados en PULLDOWN deberán ir acompañados de una resistencia, en este caso de 10k.

Las salidas de joystick irán conectados a las entradas a las entradas digitales A0 y A1 respectivamente y las salidas analógicas del acelerómetro a las entradas digitales A2, A3 y A4 respectivamente.

Creación de la board.

Para la creación de nuestra board hay que tener en cuenta estos detalles:

- Área máxima: 10cm x 6cm
- Es un diseño a una sola cara (solder side, bottom side)
- Tamaño del cable o wire a usar: 0,4064 mm (0,016inch). Excepto para los cables que conectan el chip acelerómetro. En ese caso usar un ancho de 0.3mm.
- Para trazar la pista es aconsejable crear un plano de tierra (gnd) con el cobre sobrante de la PCB.



Validación de diseño de la PCB

Procederemos a validar nuestro diseño en Eurocircuits.

Items ready for checkout

	PCB Visualizer®	Offer	Number	Type	Status	Item name	Service	Quantity	Delivery days	Unit price	Net price	Stencil top p
	✓ PCB Visualizer®		B0930412	PCB	Ready to checkout	ldh	NAKED proto	2	7 Working days	15.49	30.98	

PCB Visualizer® v1.4-24-170119

Board name ldh (B0930412) Data set: Customer data

©2016 - Eurocircuits n.v.

PCB Configurator PCB Checker

DRC - DFM information

DRC information DFM information

Layer Required Measured

Outer layer trackwidth (OL-TW) 0.250 mm 0.301 mm

Bottom copper 0.250 mm 0.301 mm

Outer layer isolation distance (OL-TT-TP-PP)

Fault view

Outer layer trackwidth (OL-TW) - Bottom copper

Current issue

Trackwidth

More information can be found here.

Board buildup

Top view

Total material thickness: 1.59 mm

Bird's Eye View

Det Summary

Service NAKED proto

Estimated shipment date 31-01-2017

Quantity 2 PCBs

Board surface / Order surface 0.60 dm² / 1.20 dm²

Prices

Gross* € 18.74

Single PCB € 37.49

Total boards € 4.45

Express transport € 41.94

Total

* The gross prices include 21.00% VAT.

Save changes

For more "Advanced options", switch to our standard PCB calculator offering "PCB proto" and "STANDARD pool" service options.

Advanced options

Alternatives

Customized matrix

2 PCBs	5 PCBs
7 working days	7 working days
Gross* € 18.74	Gross* € 9.35

Ensamblado de componentes

Para el ensamblado de componentes primero procederemos a soldar las resistencias SMD ya que para ello necesitaremos la mayor estabilidad posible. Para ello deberemos pre estañar uno de los path, colocar la resistencia, aplicar calor para que se suelde uno de los terminales sobre el path pre estañado y una vez hecho esto procederemos a soldar el otro terminal como si fuese un componente through hole.

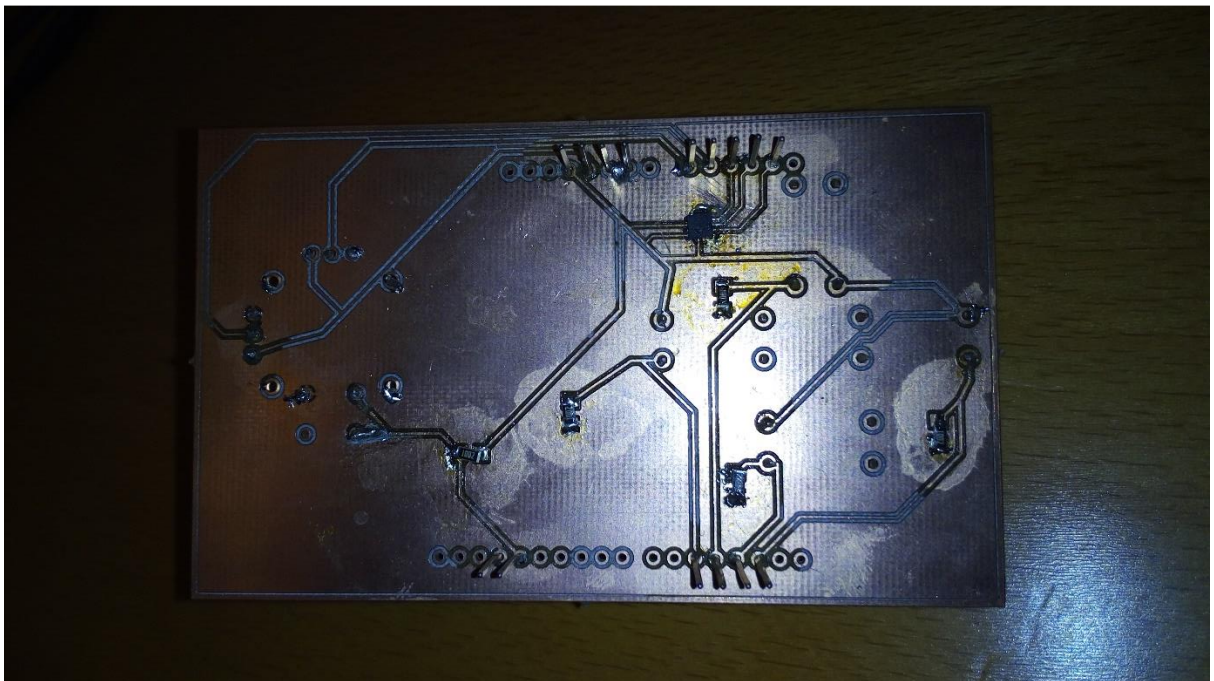
Una vez soldadas las resistencias procederemos a soldar los componentes through hole. Para ello colocamos el terminal en el path, aplicamos calor con el soldador y aplicamos estaño.

Hay que tener en cuenta que hay que comprobar el correcto ensamblado de los componentes haciendo uso de un potenciómetro para evitar el mal funcionamiento de nuestra pcb.

Para ensamblar el acelerómetro haremos uso de otra técnica. Para ello aplicamos pasta de soldadura en el path y aplicamos calor con un soldador de aire para pre estañar el path, una vez hecho esto, colocamos el acelerómetro y volvemos a aplicar calor para que se suelde.

Para más información de cómo soldar con pasta de soldadura pulse [aquí](#).

Este es el resultado final



Haciendo uso de la PCB

Como la placa arduino que dispongo es una placa clónica y no es compatible con la herramienta UNOJOY, he tenido que desarrollar un protocolo de mensajes por puerto serie entre la placa arduino y el PC y procesar los datos mediante una aplicación en C# haciendo uso de la biblioteca User32 de Windows para simular un ratón y teclado virtual.

Empaquetando la información

Off-set	Descripción	Dato
+0	Inicio de la trama	0x7E
+1	Longitud de la trama	0x0D
+2	Parte alta eje x joystick	
+3	Parte baja eje x joystick	
+4	Parte alta eje y joystick	
+5	Parte baja eje y joystick	
+6	Parte alta eje x acelerómetro	
+7	Parte baja eje x acelerómetro	
+8	Parte alta eje y acelerómetro	
+9	Parte baja eje y acelerómetro	
+10	Botón 1	
+11	Botón 2	
+12	Botón 3	
+13	Botón 4	
+14	Botón Joystick	
+15	Fin de la trama	0x0D

Programando arduino

Para programar la placa arduino dispongo de este código:

```
#include <Wire.h>

//Direccion I2C de la IMU

#define MPU 0x68

//MPU-6050 da los valores en enteros de 16 bits

int16_t AcX, AcY;

//Angulos

float Acc[2];

int xjoy=A0, yjoy=A1,xjoy2=A2, yjoy2=A3;

int boton1=2,boton2=3,boton3=4,boton4=5,botonjoy=6;

void setup() {

    Wire.begin();

    Wire.beginTransmission(MPU);

    Wire.write(0x6B);

    Wire.write(0);

    Wire.endTransmission(true);

    Serial.begin(9600);

    pinMode(boton1, INPUT);

    pinMode(boton2, INPUT);

    pinMode(boton3, INPUT);

    pinMode(boton4, INPUT);

    pinMode(botonjoy, INPUT);

}

int ejex,ejey;

void loop() {

    Wire.beginTransmission(MPU);

    Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX

    Wire.endTransmission(false);

    Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6 registros

    AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros

    AcY=Wire.read()<<8|Wire.read();

    ejex=analogRead(xjoy);
```



```

    ejey=analogRead(yjoy);
    Serial.write(0x7E); //Inicio de comunicación
    Serial.write(0x0D); //Tamaño del paquete de datos.
    Serial.write(ejex/256); //Parte alta del eje X del joystick
    Serial.write(ejex%256); //Parte baja de eje X del joystick
    Serial.write(ejey/256); //Parte alta de eje Y del joystick
    Serial.write(ejey%256); //Parte baja de eje Y del joystick
    Serial.write(AcX/256); //Parte alta de eje X del acelerometro
    Serial.write(AcX%256); //Parte baja de eje X del acelerometro
    Serial.write(AcY/256); //Parte alta de eje Y del acelerometro
    Serial.write(AcY%256); //Parte baja de eje Y del acelerometro
    Serial.write(digitalRead(boton1)); //Valor del boton 1
    Serial.write(digitalRead(boton2)); //Valor del boton 2
    Serial.write(digitalRead(boton3)); //Valor del boton 3
    Serial.write(digitalRead(boton4)); //Valor del boton 4
    Serial.write(digitalRead(botonjoy)); //Valor del boton del joystick
    Serial.write(0x0D); //fin del paquete
    delay(100);
}

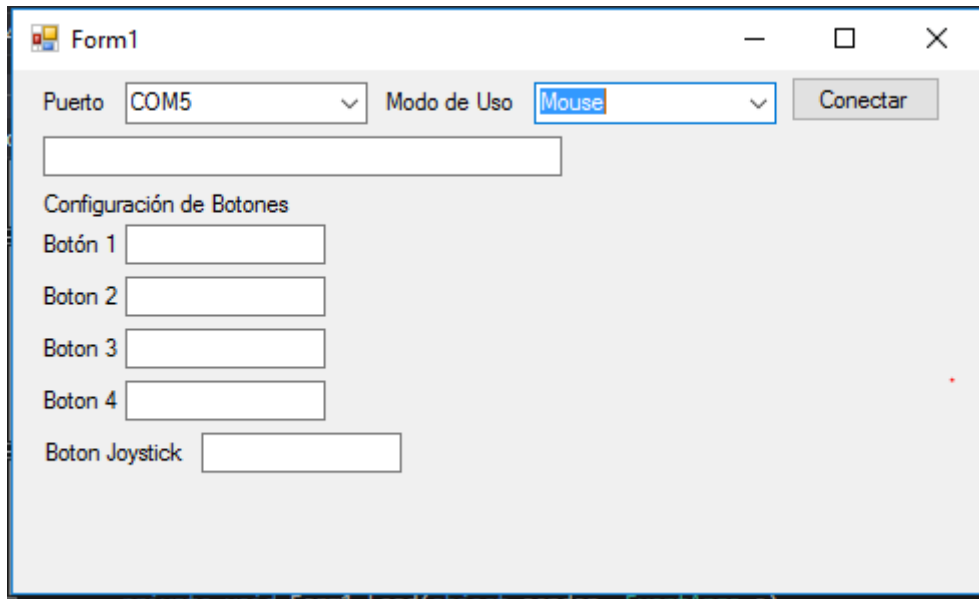
```

Programando NUCLEO

El código para NUCLEO es similar para arduino y se encuentra en el siguiente enlace de mi [repositorio de la plataforma de desarrollo mbed](#).

Procesando datos en la aplicación C#

Para procesar los datos enviado por Arduino o NUCLEO haremos uso de una aplicación en C#.



Ahora pasaré a explicar las funciones principales del programa.

Recepción de datos

```
public void RxByteStateMachine(int RxByte)
{
    switch (RxState)
    {
        case 0:
            if (RxByte == 0x7E)//Inicio de la trama
            {
                RxState = 1;
            }
            break;
        case 1:
            RxLen = RxByte; //tamaño de la trama
            RxIdx = 0;
            RxState = 2;
            RxBuff = new int[RxLen];
            break;
        case 2:
            RxBuff[RxIdx] = RxByte;//Se mete el dato dentro del buffer de
datos
            RxIdx++;
            if (RxIdx == RxLen)
            {
                RxState = 3;
            }
            break;
        case 3:
            if (RxByte == 0x0D)//Fin de la trama
            {
                ParsePacKett(RxBuff);//Funcion para traducir los datos
            }
            RxState = 0;
            break;
        default:
            break;
    }
}
```

```
}
```

Esta función es llamada cada vez que hay un dato disponible por el puerto serie.

Traducción de datos

```
public void ParsePacKett(int[] paquete)
{
    AnalogJoyX = paquete[0]*256+paquete[1];
    AnalogJoyY = paquete[2] * 256 + paquete[3];
    Boton1 = paquete[4];
    Boton2 = paquete[5];
    Boton3 = paquete[6];
    Boton4 = paquete[7];
    BotonJoy = paquete[8];
    this.Invalidate();
}
```

Funcionamiento modo ratón

```
private void ModoRaton()
{
    POINT p = new POINT();
    p.x = (500 - AnalogJoyX) / 50;
    p.y = (500 - AnalogJoyY) / 50;
    SetCursorPos((Cursor.Position.X + p.x), (Cursor.Position.Y +
p.y)); //Selecciona el puntero del sistema
    if (Boton4 == 1)
    {
        if (!m1) //Comprueba se está pulsado el botón izquierdo de nuestro
raton virtual
        {
            mouse_event(MOUSEEVENTF_LEFTDOWN, Cursor.Position.X,
Cursor.Position.Y, 0, 0); //Pulsa botón izquierdo del raton virtual
            m1 = true;
        }
    }
    else
    {
        if (m1)
        {
            mouse_event(MOUSEEVENTF_LEFTUP, Cursor.Position.X,
Cursor.Position.Y, 0, 0); //levanta el botón izquierdo del raton virtual
            m1 = false;
        }
    }
    if (Boton3 == 1)
    {
        if (!mr) //Comprueba si esta pulsado el botón derecho del raton
virtual
        {
            mouse_event(MOUSEEVENTF_RIGHTDOWN, Cursor.Position.X,
Cursor.Position.Y, 0, 0); //Pulsa el botón derecho del raton
            mr = true;
        }
    }
}
```

```

else
{
    if (mr)
    {
        mouse_event(MOUSEEVENTF_RIGHTUP, Cursor.Position.X,
Cursor.Position.Y, 0, 0); //Levanta el botón detecho del raton
        mr = false;
    }
}
}

```

Esta función es llamada constantemente

Modo Gamepad

```

private void ModoGamepad()
{
    if (AnalogJoyX < 200)
    {
        SendKeys.Send("d");
    }else
    {
        if (AnalogJoyX > 700)
        {
            SendKeys.Send("a");
        }
    }
    if (AnalogJoyY < 200)
    {
        SendKeys.Send("s");
    }
    else
    {
        if (AnalogJoyY > 700)
        {
            SendKeys.Send("w");
        }
    }
    if (Boton3 == 1)
    {
        SendKeys.Send(textBoton3.Text);
    }
    if (Boton4 == 1)
    {
        SendKeys.Send(textBoton4.Text);
    }
    if (Boton2 == 1)
    {
        SendKeys.Send(textBoton2.Text);
    }
    if (Boton1 == 1)
    {
        SendKeys.Send(textBoton1.Text);
    }
}

```

Ventajas e inconvenientes de usar la aplicación C#

El mayor inconveniente de usar esta aplicación es que para poder usar nuestro diseño como gamepad o raton necesitamos que la aplicación esté abierta, ya que al no disponer de Drivers HID nuestras placas no podemos usarlas de otra forma.

La gran ventaja que tiene es que es totalmente “usable” por cualquier placa de desarrollo que disponga de puerto serie.

Puedes descargar el proyecto [aquí](#).