

LDH - Memoria de prácticas

José Antonio Torrecilla Fuentes

Sevilla, 23 de Enero de 2017

Introducción	2
Diseño	2
Fabricación	2
Ensamblaje	2
Incidencias	3
Test	3
Código Arduino	3
Código Visual Studio	4

Introducción

Se redacta en el presente documento una memoria de la práctica realizada en el segundo bloque de la asignatura Laboratorio de Desarrollo Hardware. En esta práctica se ha podido elegir entre crear un shield para la placa de desarrollo Arduino o hacerlo, además, compatible con la placa de desarrollo Papilio; en este caso he optado por hacerlo compatible únicamente con Arduino.

En concreto se ha creado un shield que cuenta con 4 pulsadores y un joystick, siguiendo las especificaciones que se indicaban en la práctica. A lo largo del proceso, vamos a poder diferenciar cuatro etapas claramente diferenciadas: Diseño, fabricación, ensamblaje y test.

Diseño

Para el diseño se ha podido elegir entre dos famosas herramientas para este fin: Eagle y KiCAD. Ambas herramientas son muy potentes en mi opinión, sin embargo, he decidido usar KiCAD por el simple hecho de ser una herramienta de código abierto y totalmente gratuita. La etapa de diseño es un proceso que tiene también varias fases.

Esquemático

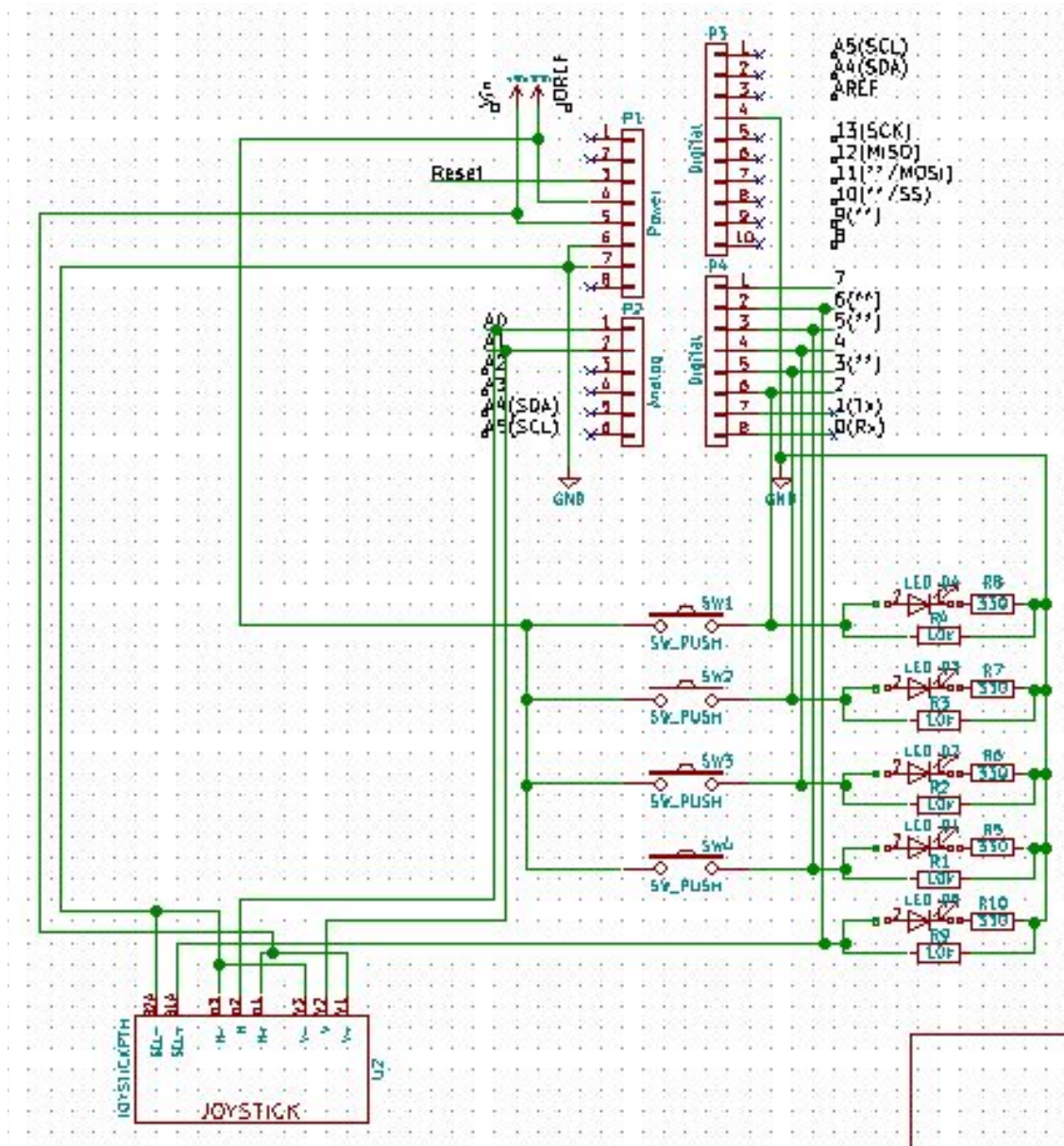
En KiCAD el editor del esquemático del circuito se llama "Eeschema". En este editor vamos a poder añadir todos los componentes que incluyen las amplias librerías que trae por defecto la última versión de KiCAD. Así que vamos añadiendo todos los componentes que se indica en la especificación, a la vez que comprobamos que todos están a nuestra disposición. Esta es una lista de los nombres de los componentes usados en el esquemático de KiCAD:

Descripción	Nombre
Conector de 6 pines	CONN_01X06
Conector de 8 pines	CONN_01X08
Conector de 10 pines	CONN_01X10
Diodo LED	LED
Resistencia	R
Pulsador	SW_PUSH
Tierra	GND

Como nos damos cuenta de que no tenemos ningún joystick, tenemos dos opciones: O bien, podemos diseñarlo, tanto el símbolo esquemático como después, la huella; o bien podemos incluir una nueva librería que incluya este componente. Siempre que podamos, optamos por la segunda opción, más fiable y rápida.

El fabricante del joystick (SparkFun) pone a nuestra disposición una librería para Eagle, que el usuario ["The J" del foro de SparkFun](#) se ha molestado en compatibilizar para KiCAD y lo ha compartido a la comunidad, y que desde aquí agradecemos. Una vez descargados los ficheros, podemos distinguir la librería de símbolos esquemáticos por tener formato ".lib" y la agregamos desde el editor de librerías.

Ahora ya podemos usar el símbolo del joystick llamado "JOYSTICKPTH". Ya solo tenemos que realizar las conexiones oportunas tal y como se indica en las especificaciones. Además podemos asignar valores a las distintas resistencias, condensadores, etc. para un posible capa de serigrafía con información relevante. Mi esquema ha quedado así:



Asignación de huellas

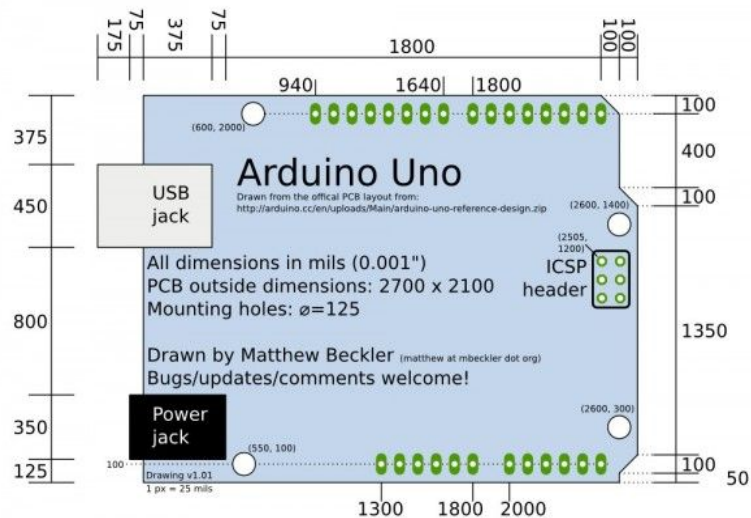
En KiCAD la asignación de huellas a los componentes se hace en “CvPCB”. Aquí tenemos que añadir la librería de huellas de SparkFun para asignar su huella al joystick. Para ello, en “Librerías de huellas”, añadimos una librería manualmente, escribiendo la ruta donde se encuentra el archivo, que distinguiremos por tener formato “.mod” y eligiendo “Legacy” en el tipo de complemento. Ya solo tenemos que asignar a cada componente, su huella correspondiente, esta es la asignación que yo he usado:

LED	LED-3MM
CONN_01X06	Socket_Strip_Straight_1x06
CONN_01X08	Socket_Strip_Straight_1x08
CONN_01X10	Socket_Strip_Straight_1x10
R	R_1206
SW_PUSH	SW_PUSH-12mm
JOYSTICKPTH	SparkFun-JOYSTICK

Una vez realizada la asignación, guardamos, cerramos CvPCB y generamos el listado de redes, permitiendo que KiCAD de una enumeración automática a todos los componentes.

PCB

En KiCAD el editor de diseño de PCB se llama “Pcbnew”. Lo primero que debemos hacer es leer el listado de redes para que aparezcan todos las huellas de los componentes. El segundo paso, seleccionando previamente la capa de corte, es dibujar el contorno de la placa y los agujeros para tornillería. El tercer paso, colocar los conectores de Arduino respetando las medidas para que cumpla la función de shield. Yo he usado esta imagen de referencia:



Una vez tenemos los límites geométricos de nuestra placa, podemos empezar a colocar las huellas sobre esta. Antes de comenzar a dibujar pistas, debemos elegir la capa de cobre “bottom” que en KiCAD se llama “B.Cu (PgDn)” y configurar el ancho de pista y la distancia de separación entre pistas. Hecho esto, empezamos a conectar los pads; para esto recomiendo usar el lienzo OpenGL pulsando F11, ya que es mucho más cómodo. Podemos añadir texto sobre el cobre con la herramienta adecuada.

Cuando todas las pistas están conectadas, realizamos una comprobación de reglas de diseño y si todo está correcto, vamos a añadir una zona de relleno de cobre de alivio térmico, que comprenderá toda la superficie de la placa. Por último generamos los ficheros Gerber: Elegimos las capas “B.Cu” y “Edge.Cuts” y generamos por separado el archivo de taladros.

Fabricación

Una vez finalizada la fase de diseño y generados los archivos Gerber, hemos procedido a la fabricación de la placa. Para ello hemos usado una placa virgen con sustrato de baquelita de 1.6 milímetros de grosor. El traslado del patrón de circuito a sustrato ha sido mediante la técnica de fresado, haciendo uso de una máquina tipo plotter que graba sobre la placa el dibujo deseado empleando fresas que eliminan el cobre de la misma. La máquina usada es de la marca LPKF. No se ha aplicado ningún tipo de máscara de soldadura, ni capa de información de componentes o “silk-screen”.

Dado que nuestra PCB es una “single-sided”, es decir, tiene una sola capa de interconexión de elementos, necesitaremos únicamente 3 archivos: La capa de cobre trasera que será nuestra cara de soldadura (solder side), la capa de corte y la capa de taladros.

Ensamblaje

Es el momento de ensamblar los distintos componentes sobre nuestra placa. Los primeros componentes a soldar serán las resistencias SMD y lo haremos de una en una. Como estos componentes son muy pequeños, la dificultad es considerablemente mayor que con respecto al resto de componentes y usaremos todos los recursos que puedan ayudarnos a obtener el mejor resultado.

Comenzaremos colocando la placa en el soporte con pinzas tipo cocodrilo, comprobando que tenemos a mano todo lo que vayamos a necesitar: la esponja (húmeda si es necesario), el estaño, el polímetro, el flux... Precalentamos el soldador a 350 °C y colocamos la primera resistencia sobre la huella correspondiente con la ayuda de unas pinzas y usamos una de las pinzas tipo cocodrilo del soporte para que realice presión para que esta no se mueva cuando el soldador entre en contacto (truco personal).

Una vez bien colocada, con el hilo de estaño en la mano con menos precisión (en mi caso la izquierda) y el soldador en la otra mano, pre-estañeamos la punta del soldador, la pasamos por la esponja y la ponemos en contacto con el pad y el terminal a soldar, esperando un breve instante para que ambos alcancen la temperatura deseada, entonces acercamos el hilo de estaño y fundimos una cantidad adecuada (en mi opinión, preferible una cantidad mínima, como la sal en la comida). El estaño debe repartirse de forma homogénea entre el pad y el terminal, si no es así, es recomendable usar flux para reducir la tensión superficial del estaño. Retiramos el soldador para que pad, terminal y estaño bajen, como un conjunto, de temperatura.

Comprobamos la conexión con el polímetro para dar por finalizado el proceso y repetirlo con el resto de terminales y componentes. Si no conduce, es posible que hayamos añadido demasiado poco estaño, en ese caso repetimos el proceso sobre el mismo terminal. Hay que prestar especial atención a la orientación de los componentes en el momento de su colocación.

Con los componentes “through-hole” o de agujero pasante, la colocación es mucho más sencilla, ya que podemos doblar un poco los terminales para que no se muevan. La soldadura de estos componentes requerirá mayor tiempo para calentar el pad y el terminal, dado que estos son más grandes y hay mayor cantidad de metal que calentar. A medida que soldamos estos componentes, podemos ir retirando el terminal sobrante cortándolos con unos alicates, para que no estorben en las siguientes soldaduras.

Una vez finalizada la placa, antes de apagar el soldador, volver a pre-estañar la punta y pasarla por la esponja para que quede solo una fina capa de estaño. De este modo protegemos la punta y mejoramos su conservación y en definitiva, la vida de esta.

Incidencias

Al soldar los diodos LED, algunos se han sobrecalentado y no funcionan. Habría que haber bajado la temperatura del soldador y precalentar los pad y los terminales durante menos tiempo.

Test

Para poner a prueba el shield se ha realizado un programa en C# para Windows que abre una comunicación serie con el Arduino, recibe las señales generadas por el joystick y los botones y las usa para seleccionar un color y mostrar la inclinación de los ejes del joystick.

Código Arduino

```
int joyPin1 = 0;
int joyPin2 = 1;
int buttona = 5;
int buttonb = 4;
int buttonc = 3;
int buttond = 2;
int value1 = 0;
int value2 = 0;
int vbuttona = 0;
int vbuttonb = 0;
int vbuttonc = 0;
int vbuttond = 0;

void setup() {
  Serial.begin(115200);
}

void loop() {
  value1 = analogRead(joyPin1);
  delay(100);
  value2 = analogRead(joyPin2);
  delay(100);

  vbuttona = digitalRead(buttona);
  vbuttonb = digitalRead(buttonb);
  vbuttonc = digitalRead(buttonc);
  vbuttond = digitalRead(buttond);

  int boton = 0;
  if(vbuttona) boton = 1;
  else if(vbuttonb) boton = 2;
```



```

else if(vbuttonc) boton = 3;
else if(vbuttond) boton = 4;
else boton = 0;

Serial.write(0x7E);
Serial.write(5);
Serial.write(value1 & 0xff);
Serial.write(value1 >> 8);
Serial.write(value2 & 0xff);
Serial.write(value2 >> 8);
Serial.write(boton);
Serial.write(0x0D);
delay(100);
}

```

Código Visual Studio

```

// Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace LDH_Practica_JoseAntonioTorrecillaFuentes
{
    public partial class Form1 : Form
    {
        public int i;
        public Form1()
        {
            InitializeComponent();
            i = 0;
            // Abrir el puerto serie al inicio.
            serialPort1.Open();
        }

        private void serialPort1_DataReceived(object sender,
        System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            int b;
            while (serialPort1.BytesToRead > 0)
            {
                b = serialPort1.ReadByte();
            }
        }
    }
}

```

```

        // Procesar el dato recibido
        rxByteStateMachine(b);
    }
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // Cerrar el puerto serie al cerrar el formulario.
    serialPort1.Close();
}

public int rxStatus, rxLen, rxIdx;
public int [] rxBuff;
public void rxByteStateMachine(int rxByte)
{
    switch (rxStatus)
    {
        case 0: // Inicio del paquete
            if (rxByte == 0x7E)
            { //Si se recibe el inicio de la trama
                rxStatus = 1; // Estado de longitud
            }
            break;
        case 1: // Longitud
            rxLen = rxByte;
            rxIdx = 0;
            rxBuff = new int[rxLen];
            rxStatus = 2;
            break;
        case 2: // Recepción de datos
            rxBuff[rxIdx] = rxByte;
            rxIdx++;
            if (rxIdx == rxLen) rxStatus = 3;
            break;
        case 3: // Fin del paquete
            if (rxByte == 0x0D)
            { // Si se recibe el fin de trama
                // Decodificar el paquete
                ParsePacket(rxBuff);
            }
            rxStatus = 0; // Vuelta al estado inicial
            break;
        default:
            break;
    }
}

int AnalogJoyX, AnalogJoyY, DigitalJoy;
public void ParsePacket(int[] Packet)
{
    // Decodificación del eje X
    AnalogJoyX = (Packet[1] << 8) | Packet[0]; // Cambiar endianness en el micro para

```

que sea igual enunciado

```
AnalogJoyX = 1024 - AnalogJoyX;  
// Descodificación del eje Y  
AnalogJoyY = (Packet[3] << 8) | Packet[2];  
AnalogJoyY = 1024 - AnalogJoyY;  
//Descodificación Joy Digital  
DigitalJoy = Packet[4];
```

```
this.Invalidate();  
}
```

```
private void Form1_Paint(object sender, PaintEventArgs e)  
{  
    textBox1.Text = "Eje X: " + AnalogJoyY + "; Eje Y: " + AnalogJoyX + "; Botón: " +  
    DigitalJoy;
```

```
    progressBar2.Value = AnalogJoyX;  
    progressBar1.Value = AnalogJoyY;
```

```
    switch (DigitalJoy)
```

```
{  
    case 0: panel1.BackColor = Color.Red;  
    break;  
    case 1: panel1.BackColor = Color.Blue;  
    break;  
    case 2: panel1.BackColor = Color.Orange;  
    break;  
    case 3: panel1.BackColor = Color.Yellow;  
    break;  
    case 4: panel1.BackColor = Color.Purple;  
    break;  
    case 0: panel1.BackColor = Color.Red;  
    break;  
}
```

```
}
```