

Universidad de Sevilla

# Memorias Practicas

Laboratorio de Desarrollo de Hardware

Francisco Javier Solís Franco  
30-11-2016

## Índice

<b>Memoria 2: Plataforma FPGA Papilio(ZPuino).</b> .....	2
<b>Objetivos</b> .....	2
<b>Introducción</b> .....	2
<b>Desarrollo de la practica</b> .....	3
<b>Diseño de circuito básico en la FPGA.</b> .....	3
<b>Cargar Soc ZPUino y cargar sketches.</b> .....	4
<b>La placa PAPILIO como Analizador lógico.</b> .....	5
<b>Rediseñando el SoC. Añadiendo periféricos.</b> .....	8
<b>Conclusión</b> .....	9

# Memoria 2: Plataforma FPGA Papilio(ZPuino).

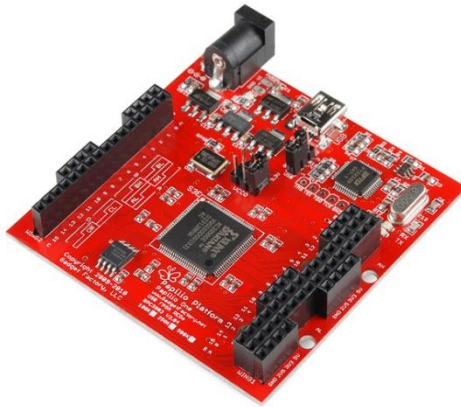
## Objetivos

Los objetivos de las practicas con esta plataforma es conocer la plataforma PAPILIO, la instalación y configuración del entorno de desarrollo DESIGN LAB y conocer lo que podemos hacer con este entorno y las placas papilio como:

- Diseñar circuitos a nivel esquemático e implementarlos en la FPGA.
- Cargar Soc's (ZPuino en nuestro caso).
- Desarrollo de Sketches para ZPuino.
- Realizar un analizador lógico con la placa PAPILIO.
- Añadir periféricos al Soc ZPuino y desarrollar sketches que los usen.

## Introducción

La plataforma PAPILIO son placas de desarrollo hardware abiertas que se basan en FPGAs de XILINX, en nuestro caso usaremos la **PAPILIO One** que cuenta con la FPGA **SPARTAN3E 500**. Que cuenta con 4Mbit de memoria SPI Flash, dos canales USB para JTAG y comunicaciones serie y 48 I/O lines, entre otras cosas.



Las WINGS son placas de expansión como las shield de Arduino pero adaptadas para a los conectores de expansión de la papilio.

ZPuino es un Soc (System-on-a-chip) implementado con VHDL que se basa en el microprocesador ZPU de Zylins que cuenta con las especificaciones y el diseño abiertas, y al tratarse de un diseño soft-core podemos añadir nuevos periféricos al Soc según nuestras necesidades.

El entorno de desarrollo nos permite trabajar como en Arduino ya que se basa en el mismo IDE pero adaptado para ser compatible con ZPuino, que también nos permite modificar el Soc a nivel hardware para añadir nuevos periféricos.

## Desarrollo de la practica

Comenzamos por instalar el entorno, el cual conseguiremos a través de <http://10.1.15.78/~bellido> y seguiremos la guía de instalación de en Linux de [gadgetfactory.net](http://gadgetfactory.net).

Y al igual que con Arduino comenzaremos por ejemplos básicos e iremos ampliando el conocimiento. Lo primero que haremos será cargar el esquema de un circuito en la FPGA.

Diseño de circuito básico en la FPGA.

Seguiremos el tutorial: <http://gadgetfactory.net/learn/2015/05/03/designlab-make-a-simple-fpga-circuit-2/>, que seguiremos para diseñar el siguiente circuito:



Para comprobar el funcionamiento conectaremos sobre la PAPILO un switch y un led, lo correspondiente al circuito diseñado con el siguiente resultado:



Vídeo del funcionamiento: <https://youtu.be/v8r6XWtY9IA>

Comprobamos como con el switch conmutamos entre vcc y gnd y pasamos por el inversor diseñado mediante el led.

### Cargar Soc ZPUino y cargar sketches.

Desarrollaremos el tutorial “Open QuickStart Sketch” a través del siguiente enlace:

<http://gadgetfactory.net/learn/2015/04/03/designlab-using-the-ide-for-the-first-time/> , que consisten en un tutorial que nos muestra como cargar el SoC ZPUino en la FPGA y ejecutar sketches.

Una vez tengamos cargado el SoC ZPUino en la FPGA procederemos a cargar el sketch “papilio\_quickstart” el cual modificaremos para hacer que el control de un led sea a través del PC por el puerto serie.

### Código papilio:

```
#define circuit ZPUino_Vanilla

int ledPin=32 ;

void setup(){
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
}

void loop(){
    if(Serial.available()){
        char c = Serial.read();
        if(c == 'H'){
            digitalWrite(ledPin, HIGH);
        }else if(c == 'L'){
            digitalWrite(ledPin, LOW);
        }
    }
}
```

### Código Python:

```
import serial

papilio = serial.Serial('/dev/ttyUSB1', 9600)

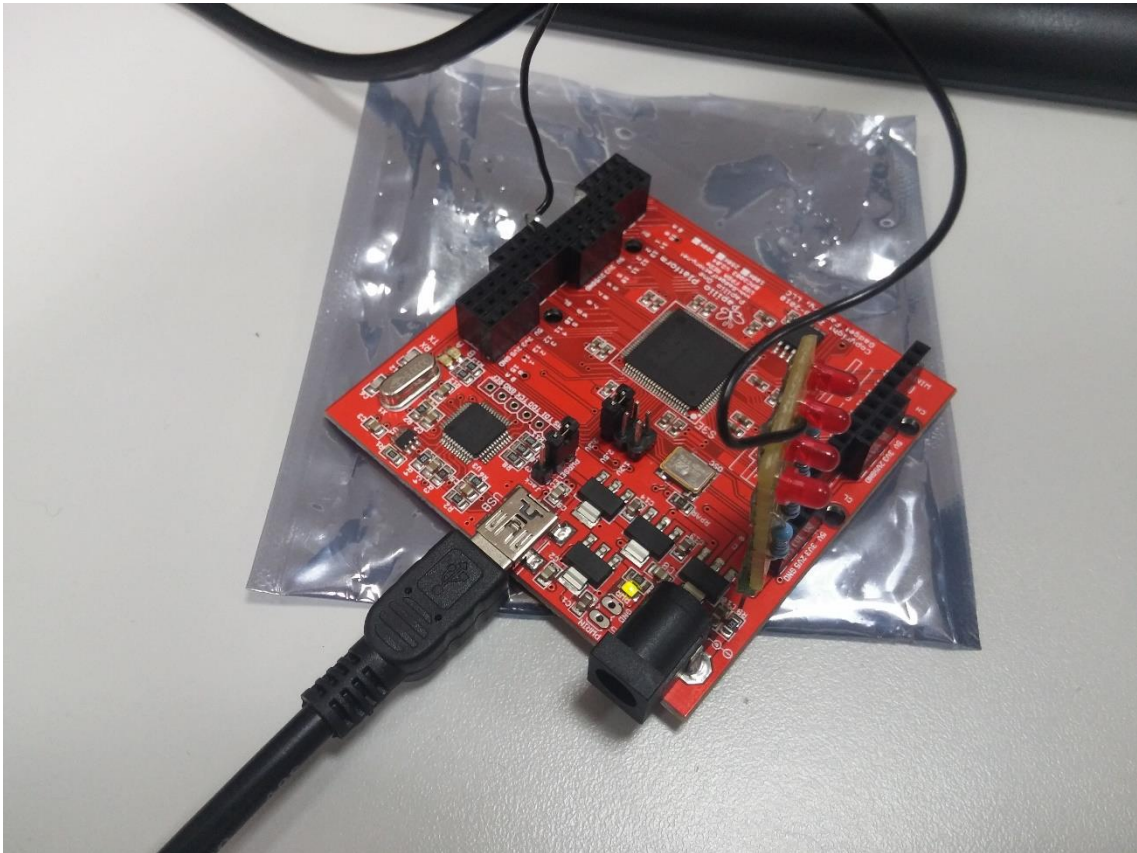
print("Starting!")

while True:
    comando = raw_input('introduce un comando: ') #input
    papilio.write(comando) #manda el comando hacia papilio
    if comando == 'H':
        print('LED encendido')
    elif comando == 'L':
        print('LED apagado')

papilio.close()
```

Comprobamos el resultado con el siguiente esquema:

- Led conectado a GND y al pin 32 de papilio, que recibirá los valores alto o bajo según se le indique por el puerto serie desde el programa en Python.



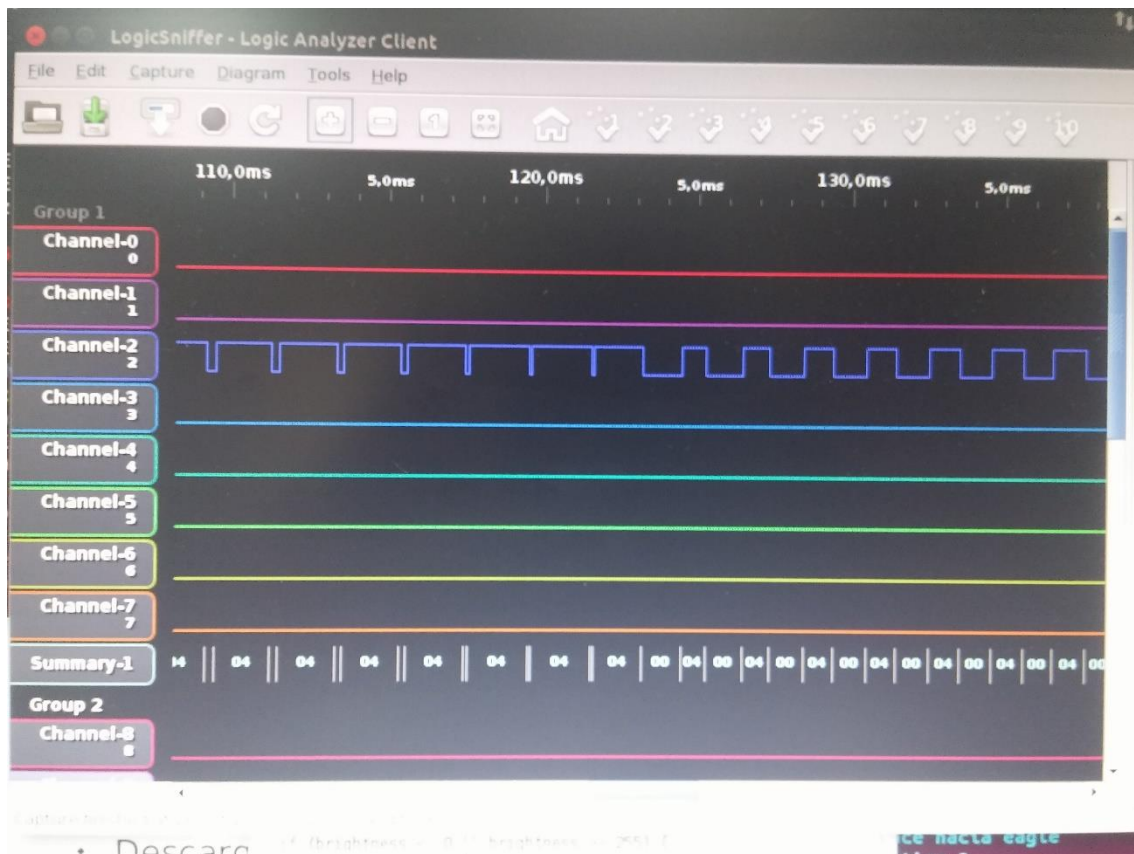
Vídeo del funcionamiento: <https://youtu.be/PVPdtZXywlw>

#### La placa PAPILIO como Analizador lógico.

Nuevamente seguiremos un tutorial para cargar el analizador lógico en la FPGA, que será este: <http://gadgetfactory.net/learn/2015/07/30/designlab-using-papilio-as-stand-alone-logic-analyzer/>.

Tendremos que configurar el analizador con algunos parámetros como la frecuencia de muestreo que será de unos 22MHz e indicar la placa que se está usando.

Tras esto cargaremos el sketch fade (en arduino), el cual encontraremos en el IDE y comenzaremos a capturar, obteniendo este resultado:



### Código arduino:

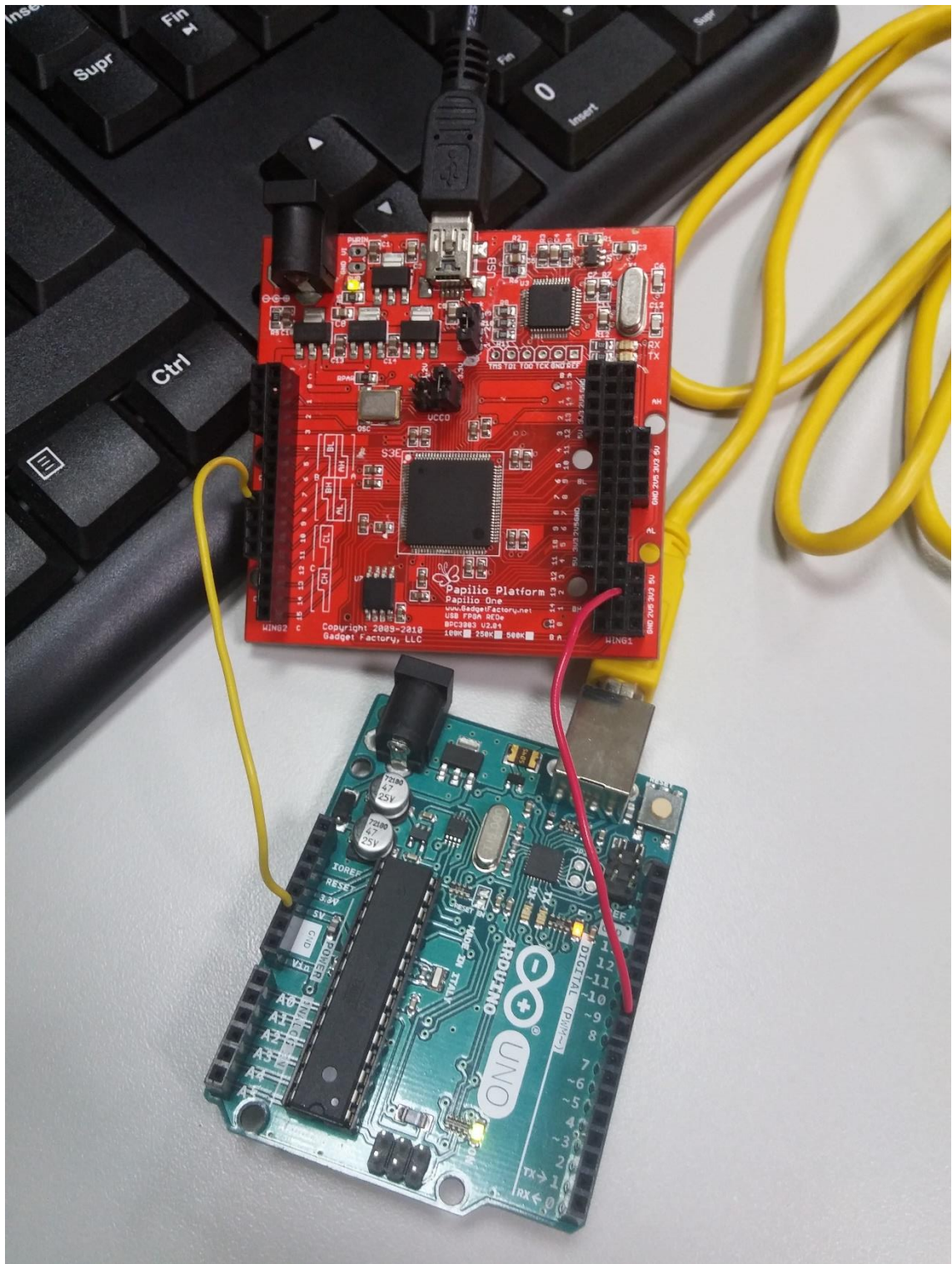
```
int led = 9;
int brightness = 0;
int fadeAmount = 5;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  analogWrite(led, brightness);

  brightness = brightness + fadeAmount;
  if(brightness <=0 || brightness >=255){
    fadeAmount = -fadeAmount;
  }
  delay(2);
}
```







## Rediseñando el SoC. Añadiendo periféricos.

Esta vez trataremos de añadir periféricos al SoC realizando los procedimientos de sintetizarlo, cargar el bit file, programarlo en la FPGA y probarlo con un sketch, seguiremos el tutorial “Creating a new ZPUino SoC project”: <http://gadgetfactory.net/learn/2015/05/15/designlab-make-a-custom-zpuino-system-on-chip-2/>

Lo que haremos será modificar una UART con unos pines concretos para usarlos como puerto serie, los cuales usaremos con un cable TTL-RS232-USB y cargaremos el sketch usado en el ejercicio del puerto serie anterior (aunque modificado ligeramente) para comprobar su funcionamiento.

### Código Papilio:

```
HardwareSerial mySerial(WishboneSlot(5));

int led = 32;

void setup(){
    pinMode(led, OUTPUT);
    mySerial.begin(9600);
}

void loop(){
    if(mySerial.available()){
        char c = mySerial.read();
        if(c == 'H'){
            digitalWrite(led, HIGH);
        }else if(){
            digitalWrite(led, LOW);
        }
    }
}
```

### Código Python:

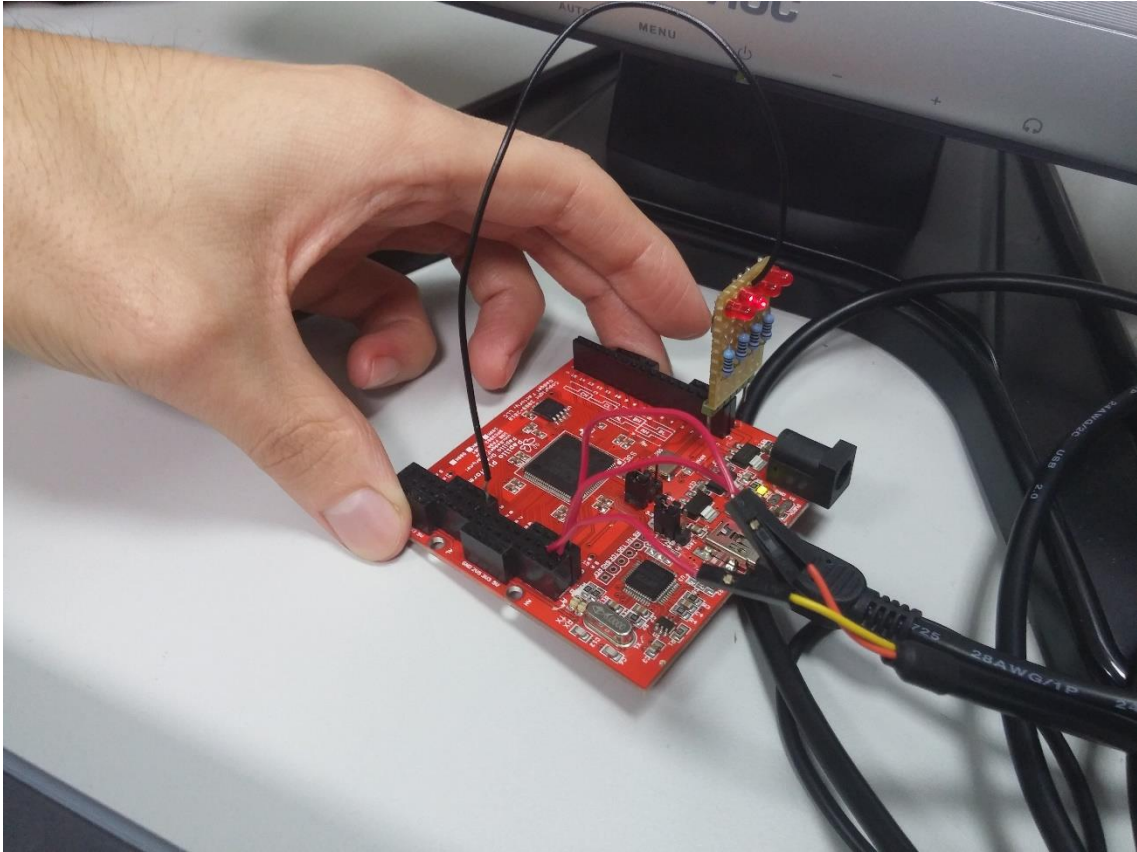
```
import serial

papilio = serial.Serial('/dev/ttyUSB2', 9600)

print('Starting!')

while True:
    comando = raw_input('Introduce un comando: ') #input
    papilio.write(comando) #mandamos el comando leído por el puerto
    serie

papilio.close() #finalizamos la comunicación
```



Vídeo del funcionamiento: <https://youtu.be/9DA4wxCv5i8>

## Conclusión

Tras conocer la plataforma PAPILO, vemos como tenemos ante nosotros una plataforma que nos permite casi cualquier cosa, desde cargar esquemáticos hasta realizar nuestro propio puerto serie, al igual que hacer analizadores lógicos.

Con estas prácticas vemos cómo usar las FPGA para poder diseñar el hardware que podamos necesitar en cada momento de una manera fácil y rápida.