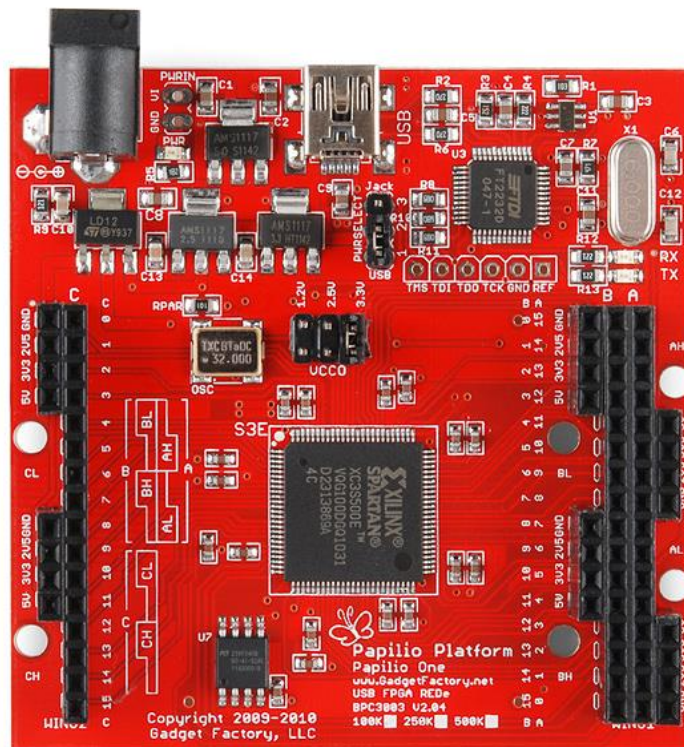


MEMORIA DE PRÁCTICA DE ZPUINO



Nombre: Francisco Núñez Cid

Correo: francisco.nunez.cid@gmail.com

Curso: 2016 – 2017

ÍNDICE

• Introducción	3
• Objetivos	3
• Fundamentos teóricos	
○ ¿Qué es ZPUino?	4
○ ¿Qué es Papilio?	4
○ Arquitectura de la placa	4
○ Programación	5
○ Utilización del software	6
○ Hardware utilizado	6
• Ejercicios	7
• Conclusión	24
• Glosario	24
• Bibliografía	25

Introducción

Durante el desarrollo de todas las prácticas de laboratorio del primer bloque, de la parte de **ZPUino**, he ido tomando una serie de notas de lo que he ido realizando, los problemas que me he ido encontrado a la hora de hacer los diferentes ejercicios, las soluciones de cada uno de ellos y los diferentes resultados que voy concluyendo.

Objetivos

Los principales propósitos de las prácticas de ZPUino son los siguientes:

- Conocer las características de una placa de entrada/salida Papilio y utilizarla en comunicación con un PC.
- Conocer una serie de componentes básicos de hardware típico de aplicaciones de sistemas empotrados.
- Instalar y configurar el software suministrado por el fabricante (DESIGN LAB), que posibilita la gestión de los distintos recursos de la tarjeta.
- Conocer que se puede hacer desde DESIGN LAB sobre las placas Papilios: diseñar circuitos a nivel de captura de esquemáticos e implementarlos en la FPGA, cargar el SoC ZPUino, desarrollar Sketches, etc.
- Conocer la estructura de un programa en Papilio Design Lab, su lenguaje de programación y el manejo del puerto serie a través de un programa en python.
- Conocer mecanismos de activación de señales externas de la placa. Obtener habilidades en el desarrollo, la carga y ejecución de programas, así como en la verificación del funcionamiento de algunos ejercicios.

Fundamentos teóricos

- ¿Qué es ZPUino?

Es un SoC implementado en HDL (principalmente VHDL), basado en un procesador ZPU de 32 bits, que funciona a 100 Mhz con una biblioteca de periféricos Wishbone.

- ¿Qué es Papilio?

Es una plataforma de desarrollo de hardware basada en FPGA en el Xilinx Spartan 3E FPGA, que permite tanto diseñar el hardware de la FPGA como desarrollar Sketches (programas) que se ejecutarán en el microcontrolador empleado (ZPUino/Arduino).

- **Arquitectura de la placa**

La placa Papilio ONE se muestra en la siguiente imagen marcando los elementos más usados.

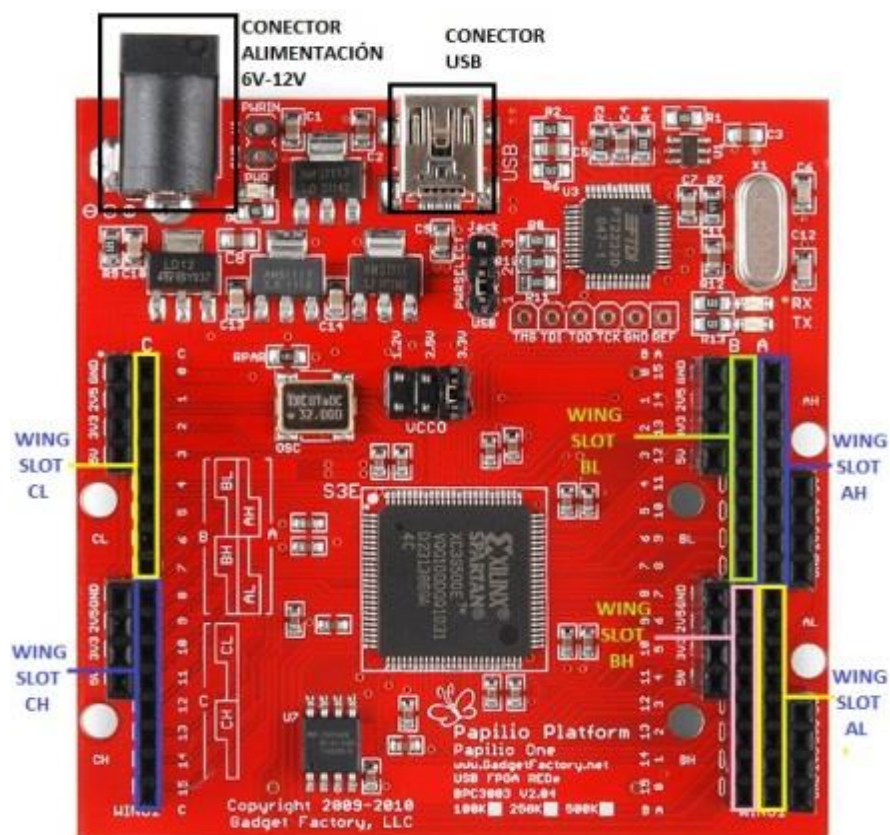


Imagen 1

Esta placa posee una FPGA de la marca Xilinx Spartan 3E y toda la circuitería de soporte que incluye, reguladores de tensión, un puerto USB doble canal para comunicación serie y JTAG, que permite programar el microcontrolador desde cualquier PC de manera cómoda y también hacer pruebas de comunicación con el propio chip y un USB conector Jack de entrada.

Dispone de 48 pines (WING A, B, C) que pueden configurarse como entrada o salida y a los que puede conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de entre 0 y 5 V, y 4 pines de alimentación: GND, 2.5V, 3.3V, 5V.

También dispone de un oscilador 32 Mhz que puede ser utilizada por DCM de Xilinx para generar cualquier velocidad de reloj requerida y de una 4M SPI Flash.

- **Programación**

Un programa en Papilio Design Lab es igual que en Arduino, consta de 3 partes:

1. Declaración de variables.

2. Función de inicialización Setup ().

Se ejecuta una única vez y es empleado para configurar un determinado pin como entrada o salida, configurar la comunicación serie, etc.

3. Función Loop ().

Aquí se encuentra el programa que controlará la respuesta de la placa Papilio, y esto lo hará infinitas veces hasta que se desconecte la alimentación de la placa.

Un ejemplo de la estructura sería:

```
// Declaración de Variables
int x ;

void setup () {
//Código inicialización microcontrolador,
velocidad serial, pines de I/O....
}

void loop () {
// Código del programa
}
```

- **Utilización del Software**

1. Descarga e instalación de Papilio Design Lab (<http://10.1.15.78/~bellido>).
2. Conexión de la placa Papilio al PC a través del cable USB.
3. Abrir una ventana de símbolos y escribir: **"Ubuntu-setup.sh"**.
4. Escribir luego: **"sudo apt-get install default-jre"**.
5. Abrir el programa Papilio Design Lab.
6. Seleccionamos la placa que estamos utilizando. Herramientas → Placa.
7. Seleccionamos el puerto al que tenemos conectada la placa. Herramientas → Puerto.

Los pasos 5 – 7 se hacen siempre cada vez que se inicia el software de Papilio.

- **Hardware utilizado**

1. Placa Papilio.
2. Cable USB.
3. Resistencias.
4. Diodos led.
5. Cables para realizar las conexiones.
6. Switch.

Ejercicios

1) Diseñar circuitos básicos sobre FPGAs.

En este primer ejercicio hay que hacer que desde Design Lab diseñemos el circuito de un inversor para implementarlo en la FPGA de la placa Papilio, y en el que se utilizará un switch y un led para comprobar su funcionamiento.

Yo he realizado este ejercicio de la siguiente manera:

1. Abrimos Papilio Design Lab, y se pulsa sobre **“Nuevo Circuito FPGA”**. Esto abrirá una nueva ventana en blanco.



Imagen 2

2. Pulsamos sobre **“Editar Circuito”**. Te saldrá una ventana para guardar el proyecto, en el que lo he llamado “Inversor”. Después de esto, se abrirá Xilinx ISE.

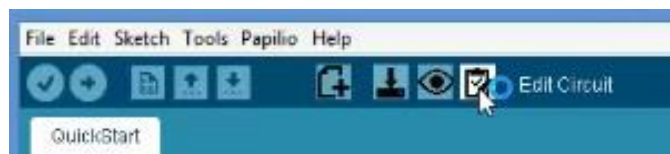


Imagen 3

3. Una vez abierto Xilinx ISE, hacer doble click en **“Papilio_One_500K”**, y debe aparecer un circuito en blanco, en el que se va a realizar nuestro diseño.

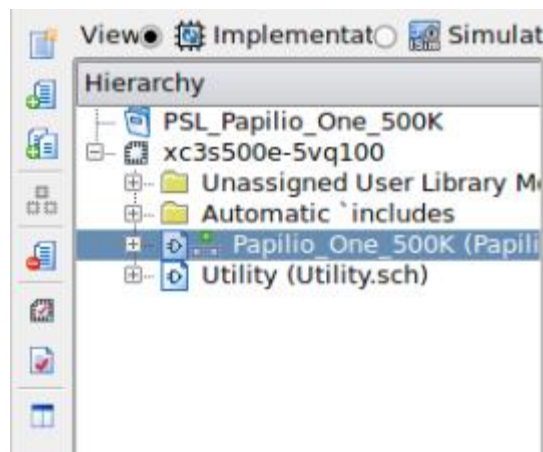


Imagen 4

4. Se realiza el diseño del inversor, y le cambio el nombre de la entrada a “WING_ALO” y la salida a “WING_CLO”.

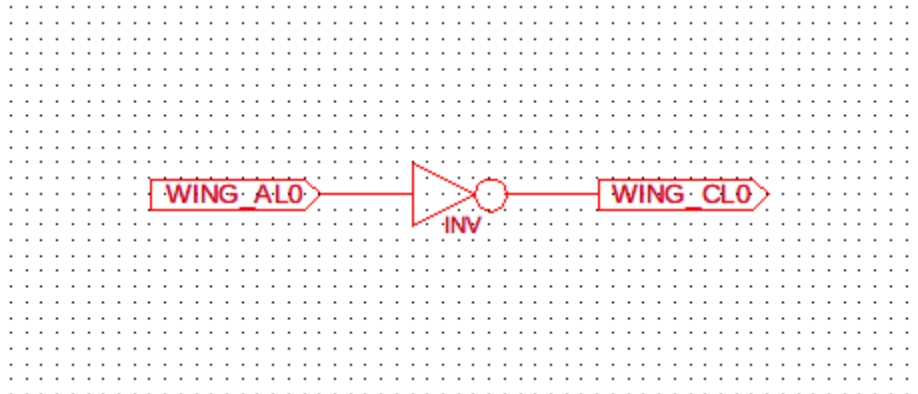


Imagen 5

5. Una vez realizado el diseño del inverso, se va a comprobar si hay errores sintácticos. Se pulsa en el cuadro de abajo en “**Generate Programming File**”, y esperar a que no haya errores y se genere el programa sin problemas.

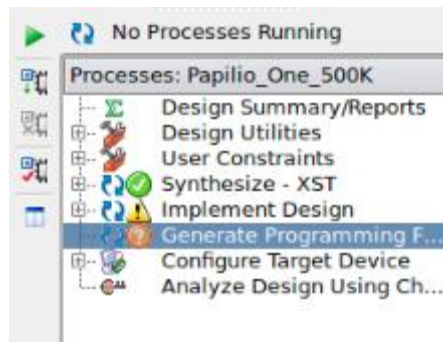


Imagen 6

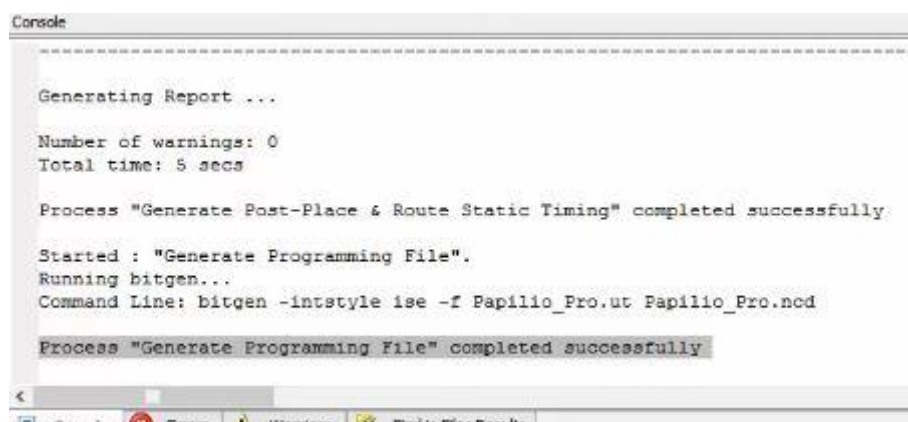


Imagen 7

6. Ahora, nos vamos de nuevo a Papilio DesignLab, y se pulsa sobre “Cargar Circuito”, y esperamos a que se cargue (Imagen 9).



Imagen 8

```

Done burning bitfile.
/home/practicas/Escritorio/DesignLab-1.0.7/hardware/tools/papilio/papilio_loader
Programming to SPI Flash
Using built-in device list
Programming a Papilio One 500K
Using built-in device list
JTAG chainpos: 0 Device IDCODE = 0x41c22093      Desc: XC3S500E

Uploading "bscan_spi_xc3s500e.bit". Done.
Programming time 178.6 ms

Programming External Flash Memory with "/home/practicas/DesignLab/Inversor/circuit/500K/papilio_one_500k.bit".
Found SST Flash (Pages=2048, Page Size=264 bytes, 4325376 bits).
Erasing   :
Ok
Verifying :
.....Pass
Programming :
.....Finished Programming
Ok
Verifying :
.....Pass
Done.
SPI execution time 15657.6 ms
USB transactions: Write 2883 read 2166 retries 367
Using built-in device list
JTAG chainpos: 0 Device IDCODE = 0x41c22093      Desc: XC3S500E

ISC_Done      = 0
ISC_Enabled   = 0
House Cleaning = 1
DONE          = 0

```

Imagen 9

7. Para cargar correctamente el Bitfile hay que hacer una modificación en el nombre del fichero que genera ISE:
- Dentro de la carpeta “Inversor”, que es el nombre que se le ha puesto al proyecto, hay que: borrar **papilio_one_500k.bit**.
 - Renombrar: **Papilio_One_500K.bit** a **papilio_one_500k.bit**.

8. Ahora, se pasa a realizar el circuito a la placa de Papilio, en que se va a utilizar un switch y un led.

El switch que tiene 3 conexiones, se va a conectar a: cable negro (GND), cable rojo (Vcc) y cable amarillo (WING_AL0).

La pata del Led va WING_CL0, y el cable negro va a GND.

El circuito montado se observa en la siguiente imagen.

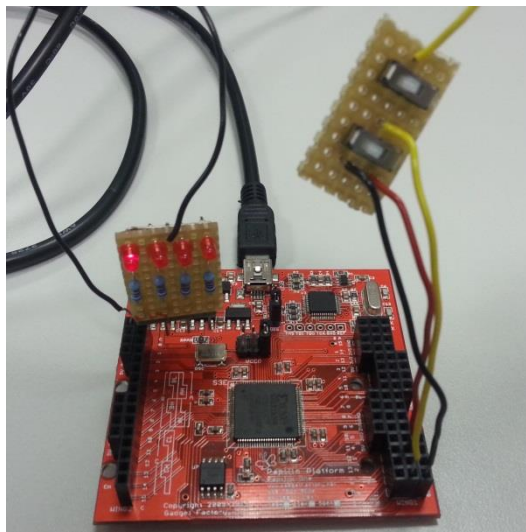


Imagen 10

Nota: Este primer ejercicio ha sido una primera toma de contacto con el entorno de programación y con la placa de Papilio. Ha sido bastante fácil este ejercicio, porque ha estado muy guiado y se explica muy bien en un tutorial como ir realizándolo. Además, no ha habido que realizar nada de programación, porque solo era diseñar un circuito sobre la FPGA.

2) Cargar SoC ZPUino y desarrollar Sketches (programas).

En este segundo ejercicio hay que hacer que a través del puerto serie encendamos un led (pulsando H) o se apague (pulsando L) con un programa Python.

Yo he realizado este ejercicio de la siguiente manera:

1. He empezado con el montaje del circuito, que he reutilizado el del ejercicio anterior, donde tengo conectado el led al WING_CL0. En este ejercicio, lo he llamado "32", que es otra forma de llamarlo.

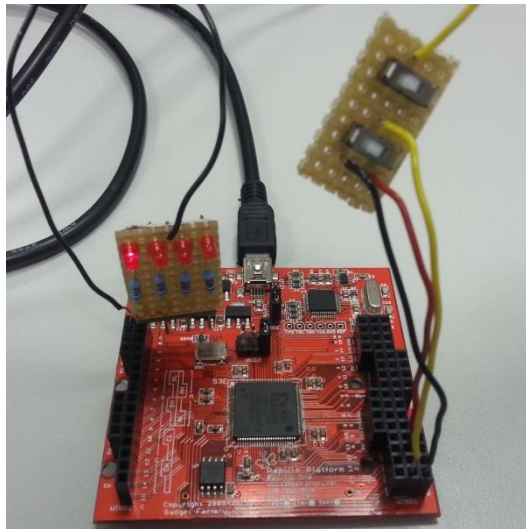


Imagen 11

2. He abierto Papilio Design Lab y he seleccionado la placa y puerto a utilizar.
3. Realizo el código que hay que mandarle a la placa Papilio (la explicación está en el código).

```
#define circuit ZPUino_Vanilla

int ledPin = 32; //Me conecto al Pin 32 = WING_CL0

void setup(){

  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);

}

void loop () {
  if (Serial.available()) { //Si está disponible
    char c = Serial.read(); //Guardamos la lectura en una
                           //variable char
  }
}
```

```

    if (c == 'H') { //Si es una 'H', enciendo el LED
        digitalWrite(ledPin, HIGH);
    } else if (c == 'L') { //Si es una 'L', apago el LED
        digitalWrite(ledPin, LOW);
    }
}
}

```

4. Pulso sobre **“Cargar Circuito”**, y esperamos a que se cargue.



Imagen 12

5. Una vez cargado el circuito en la FPGA, se pulsa sobre verificar el código y subirlo a la placa de Papilio.



Imagen 13

6. Voy a programar en Python el código que permitirá mandarle comandos a la placa Papilio, que lo he llamado **“Led.py”**, pero antes de escribir el código hay que instalar la librería **“python-serial”** en una ventana de símbolos para que no nos de error a la hora de ejecutarlo.

```
sudo apt-get install python-serial
```

El código sería:

```

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

print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando a la placa
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

arduino.close() #Finalizamos la comunicación.

```

El código es muy fácil, primero se conecta al puerto de la placa Papilio, y después mientras sea True, va leyendo los valores que se le van introduciendo al puerto serie, si es H (se enciende) y si es L (se apaga).

7. En la ventana de comandos ejecutamos la instrucción: **cd Escritorio** que es donde tengo el archivo **Led.py**.
8. Por último, ejecuto la instrucción **python Led.py** para ejecutar el programa y debería de funcionar. Al escribir “H” en la ventana de símbolos el led se debería de encender y si pulsamos “L” el led se debería de apagar.

Nota: En este segundo ejercicio la parte de hardware no he tenido ningún problema porque he reutilizado el montaje del ejercicio 1. Para la parte de software tampoco he tenido ningún problema, porque lo que ha habido que hacer es la comunicación puerto serie que se hace como en los ejercicios de Arduino.

3) Convertir la placa Papilio en un analizador lógico.

En este tercer ejercicio lo que hay que hacer es que se cogerá una placa Arduino en el que se le cargará el ejemplo: “**Faded**”, y se deberá ver la señal PWM con el analizador lógico que debe estar funcionando en la placa Papilio.

Yo he realizado este ejercicio de la siguiente manera:

1. Abrimos Papilio Design Lab y selecciono la placa y el puerto a utilizar.
2. Pulso sobre el botón “**Analizador Lógico**”.



Imagen 14

3. Al pulsar sobre el analizador lógico sale el siguiente mensaje (que hace que se cargue un analizador lógico en la placa), y se pulsa en **"YES"**.

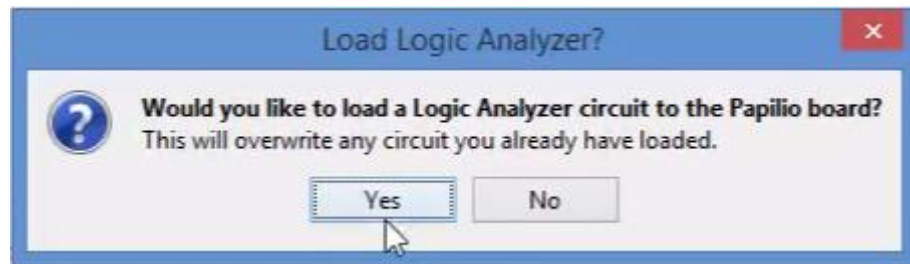


Imagen 15

4. Después, sale otro mensaje sobre la conexión de los pines, y se pulsa en **"OK"**.

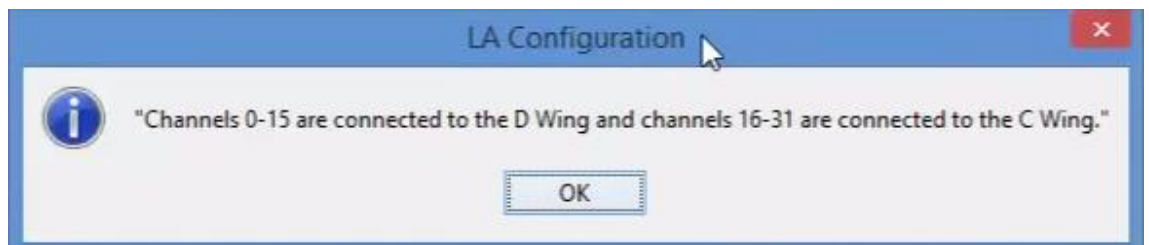


Imagen 16

5. Se debe de abrir el cliente de analizador lógico, y se pulsa sobre el botón marcado, para cambiarle las características.

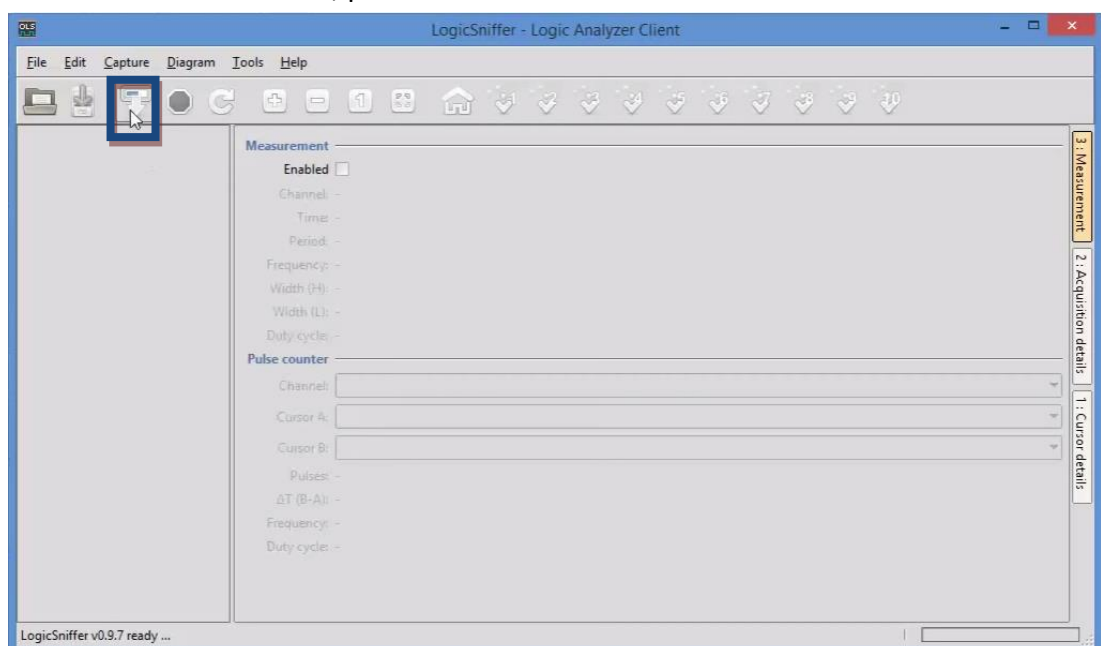


Imagen 17

6. Se abre una nueva ventana, y en “**Connection**”, cambiamos el **Analyzer port** y se le pone el puerto de la placa Papilio, cambiamos el **Port Speed** a 115200bps, y en **Device type** elegimos la placa Papilio One 500K.

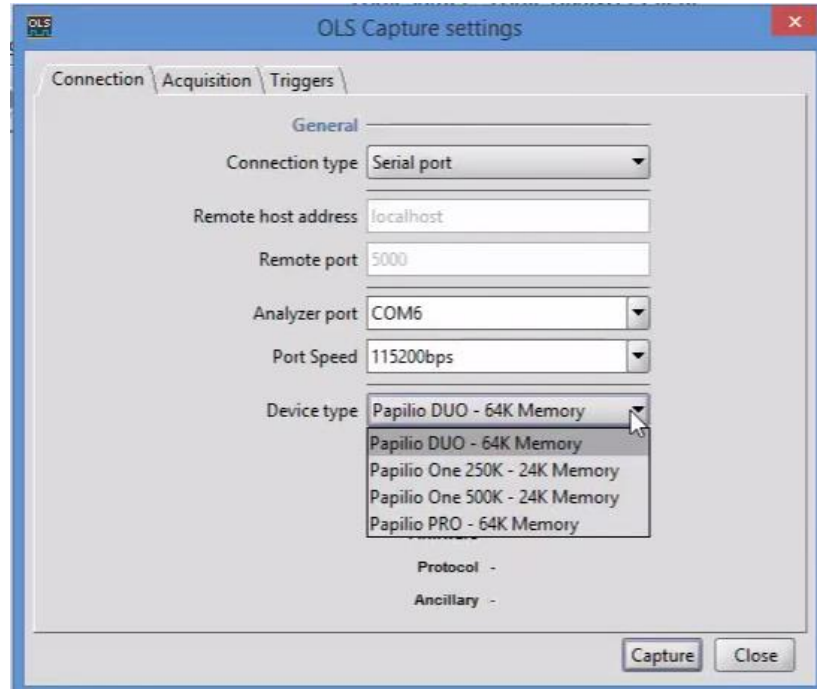


Imagen 18

7. Ahora, se pulsa sobre “**Acquisition**”, cambiamos el valor de **Sampling Rate** a 20.000 KHz y los **Channel Groups** pulsamos los 4.

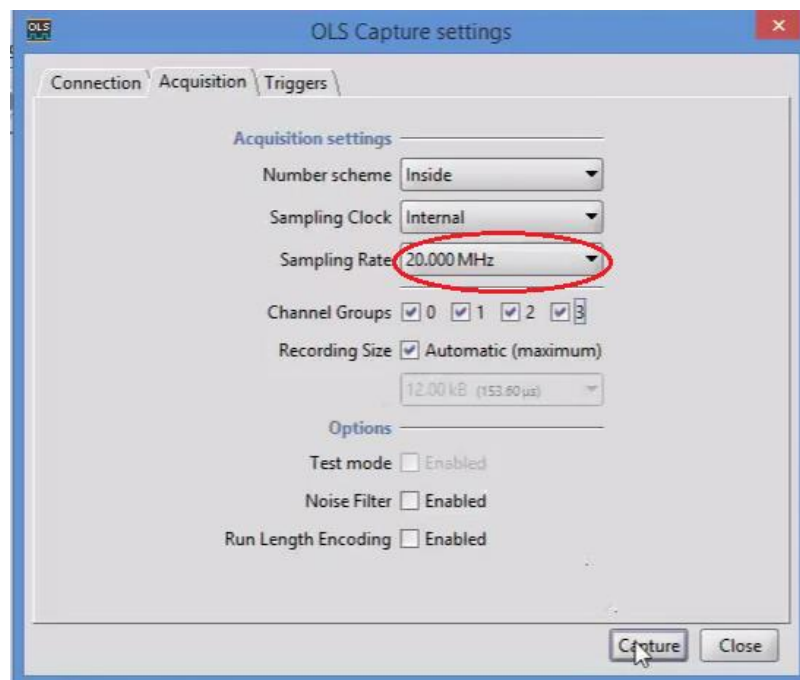


Imagen 19

8. Una vez realizado todos estos pasos anteriores se nos debería de mostrar el analizador lógico capturado (ahora mismo sin nada).

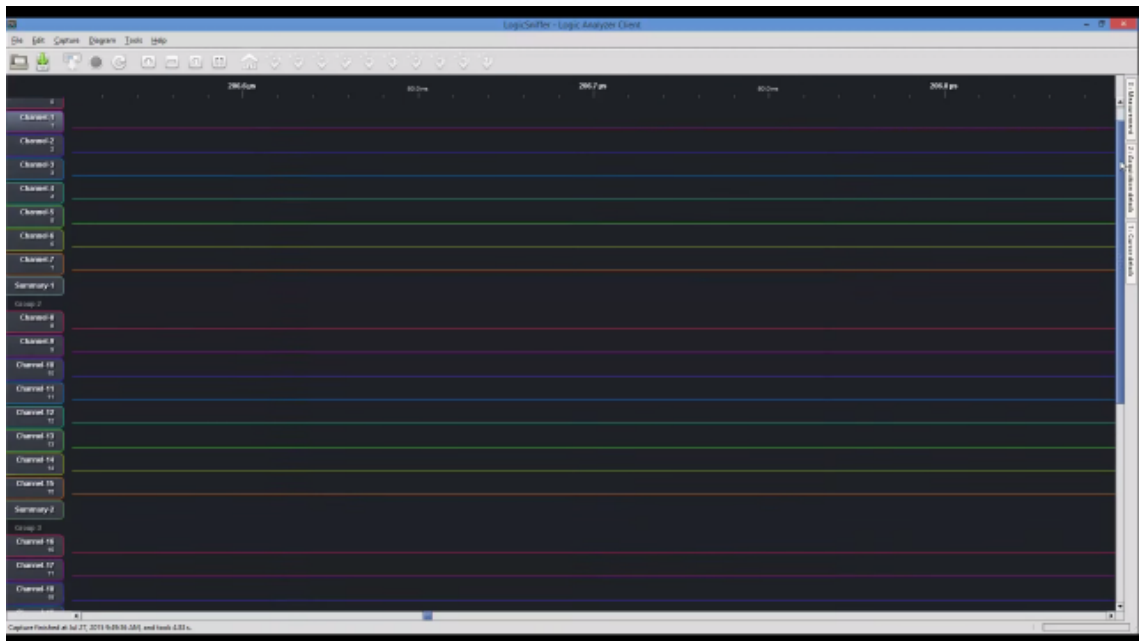


Imagen 20

9. Ahora, monto el circuito: conecto los GND de ambas placas (Arduino y Papilio) y el Pin Digital 9 (de la placa Arduino) con el pin WING_AL6 (de la placa Papilio).

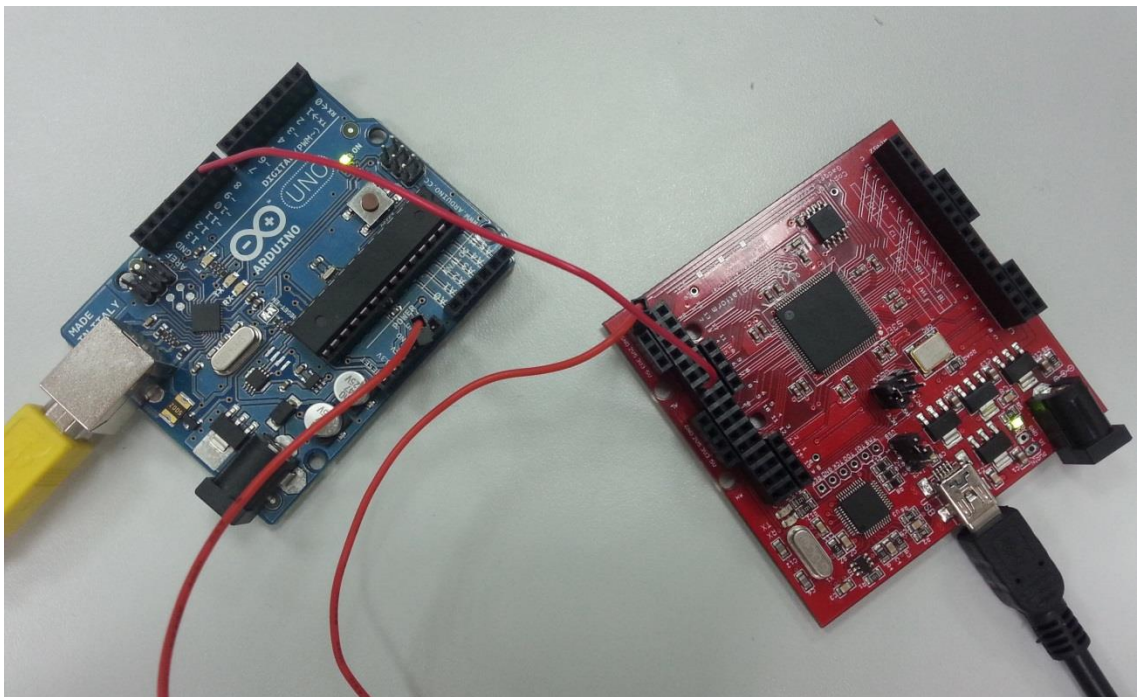


Imagen 21

10. Una vez realizado el circuito, en el programa Arduino cargamos el programa **"Faded"** y volvemos a capturar en el analizador lógico las ondas para ver ahora el PWM de Arduino.
11. Por último, si no se observase bien la onda en el **Channel 6**, se puede variar el **Sampling Rate** para verla mejor.

Nota: En este tercer ejercicio la parte de hardware no he tenido ningún problema porque solo ha sido conectar un par de cables entre ambas placas (Arduino y Papilio, para la comunicación de analizador lógico), y la parte de software ha sido utilizar el ejemplo de **"Faded"**. He tenido un problema a la hora de observar la onda y era porque el **Sampling Rate** estaba a un valor muy alto, pero lo he solucionado bajándolo y poniéndolo 20KHz. Además, ha sido bastante fácil este ejercicio, porque ha estado muy guiado y se explica muy bien en un tutorial como ir realizándolo.

4) Rediseñando el SoC añadiendo periféricos.

En este cuarto ejercicio lo que hay que hacer es desarrollar algún ejemplo básico de como añadir un periférico al SoC de ZPUino, sintetizarlo, generar el bit file, programarlo en la FPGA y utilizarlo desde un Sketch.

Yo he realizado este ejercicio de la siguiente manera:

1. Abrimos Papilio Design Lab y selecciono la placa y el puerto a utilizar.
2. Pulso sobre el botón **"Nuevo proyecto SoC ZPUino"**.

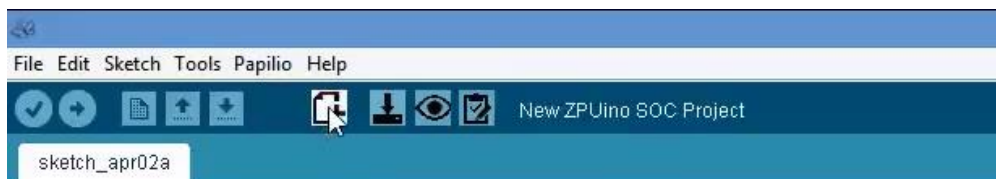


Imagen 22

3. Pulso sobre el botón **"Editar circuito"**. Después de esto, se abrirá Xilinx ISE.

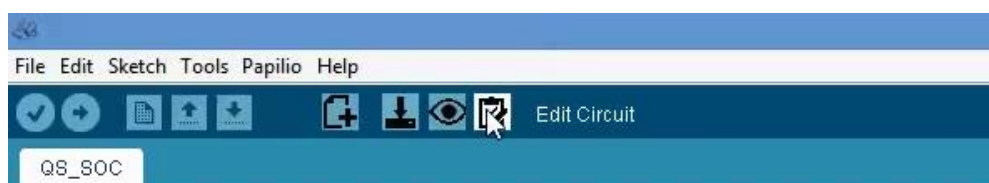


Imagen 23

4. Una vez abierto Xilinx ISE, hacer doble click en **"Papilio_One_500K"**, y debe aparecer un circuito, sobre el que se va a realizar nuestro diseño.

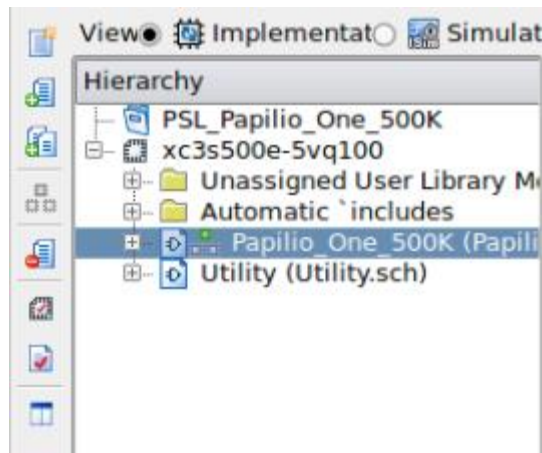


Imagen 24

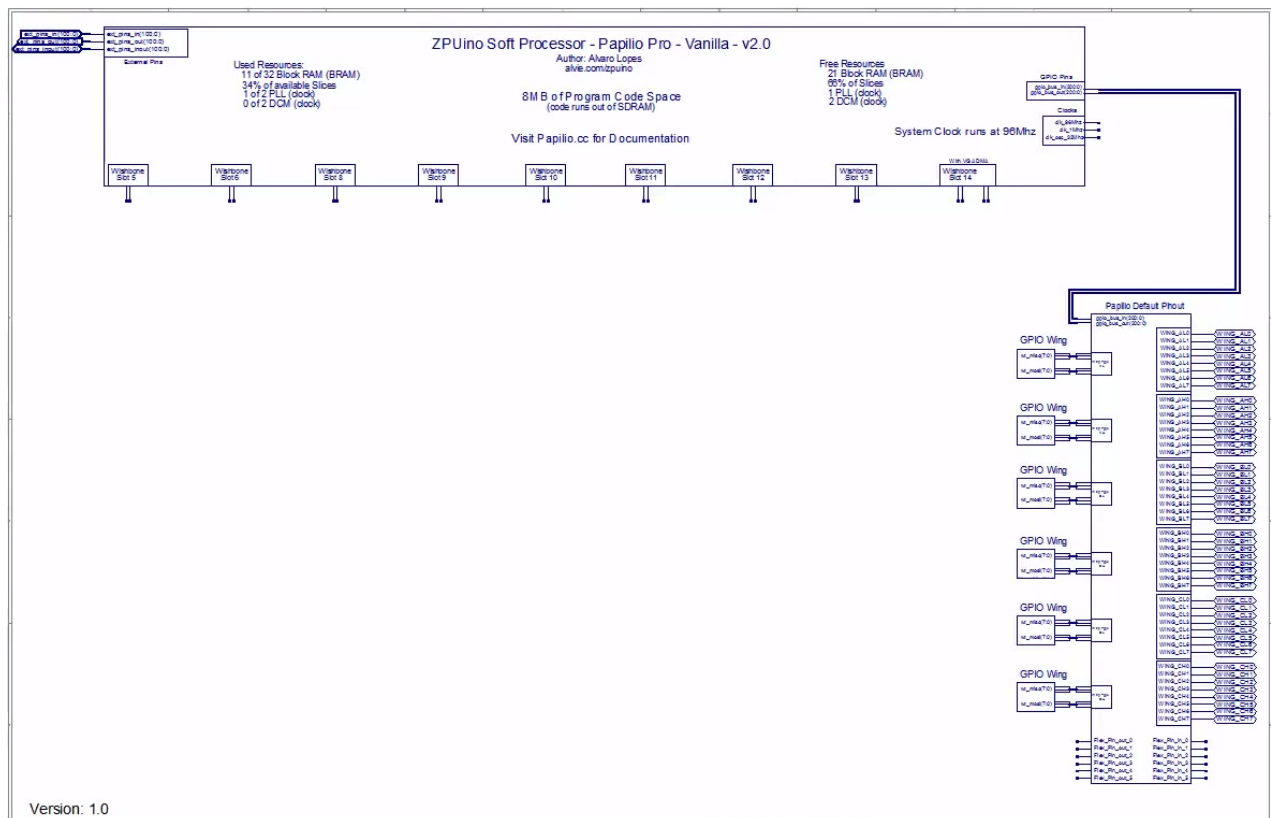


Imagen 25

5. Se realiza el diseño de la UART, y le cambio el nombre de la entrada a “WING_BL0” (rx) y la salida a “WING_BL1” (tx).

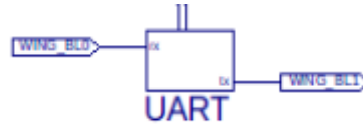


Imagen 26

6. Una vez realizado el diseño de la UART, lo añado al “Wishbone Slot 5” del ZPUino Soft Processor.

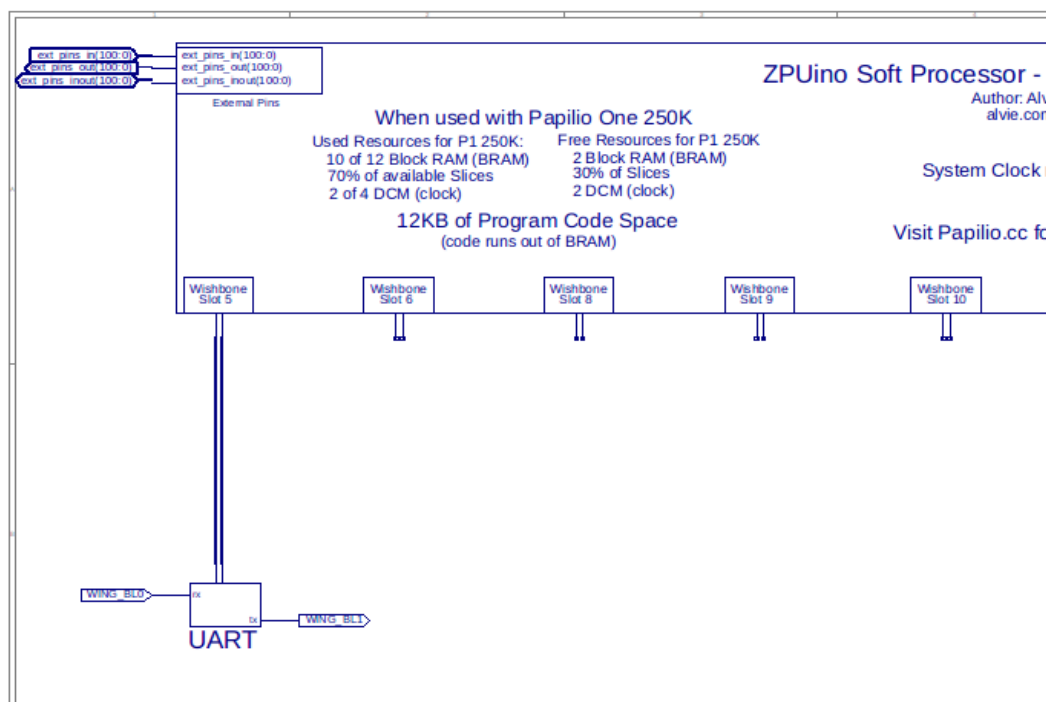


Imagen 27

7. Una vez realizado el diseño de la UART, se va a comprobar si hay errores sintácticos. Se pulsa en el cuadro de abajo en “**Generate Programming File**”, y esperar a que no haya errores y se genere el programa sin problemas.

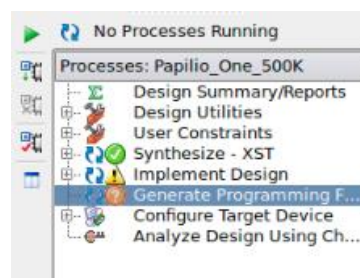


Imagen 28

8. Ahora, nos vamos de nuevo a Papilio DesignLab, y se pulsa sobre “**Cargar Circuito**”, y esperamos a que se cargue (Imagen 30).



Imagen 29

```

Done burning bitfile.
/home/practicass/Escritorio/DesignLab-1.0.7/hardware/tools/papilio/papilio_loader
Programming to SPI Flash
Using built-in device list
Programming a Papilio One 500K
Using built-in device list
JTAG chainpos: 0 Device IDCODE = 0x41c22093      Desc: XC3S500E

Uploading "bscan_spi_xc3s500e.bit". Done.
Programming time 178.6 ms

Programming External Flash Memory with "/home/practicass/DesignLab/Inversor/circuit/500K/papilio_one_500k.bit".
Found SST Flash (Pages=2048, Page Size=264 bytes, 4325376 bits).
Erasing :
Ok
Verifying :
.....Pass
Programming :
.....Finished Programming
Ok
Verifying :
.....Pass
Done.
SPI execution time 15657.6 ms
USB transactions: Write 2883 read 2166 retries 367
Using built-in device list
JTAG chainpos: 0 Device IDCODE = 0x41c22093      Desc: XC3S500E

ISC_Done      = 0
ISC_Enabled   = 0
House Cleaning = 1
DONE          = 0

```

Imagen 30

9. Para cargar correctamente el Bitfile hay que hacer una modificación en el nombre del fichero que genera ISE:

- Dentro de la carpeta “**UART**”, que es el nombre que se le ha puesto al proyecto, hay que: borrar **papilio_one_500k.bit**.
- Renombrar: **Papilio_One_500K.bit** a **papilio_one_500k.bit**.

10. Ahora paso a escribir el código en Design Lab.

Primero, me creo la variable “**mySerial1**” de tipo **HardwareSerial** y la conecta al **WishboneSlot 5**. También, la variable “**led**” y lo conecta al **WING_CL0 = 32**.

En el **loop()**, si se pulsa “**H**” (enciende el led) o si pulsa “**L**” (se apaga).

El código es:

```
HardwareSerial mySerial1(WishboneSlot(5));

int led = 32;

void setup() {
  pinMode (led, OUTPUT);
  mySerial1.begin(9600);
}

void loop() {
  if(mySerial1.available()) {
    char c = mySerial.read();
    if(c == 'H') {
      digitalWrite (led, HIGH);
    } else if (c == 'L') {
      digitalWrite (led, LOW);
    }
  }
}
```

11. Ahora, paso a realizar el montaje del circuito.

Primero, conecto el led al pin 32 o al WING_CLO (es lo mismo) y el cable negro al GND.

Luego, el cable TTL-RS232-USB tiene 3 cables (amarillo, naranja y negro).

El cable negro va conectado a GND.

El cable amarillo que es RX va conectado a WING_BL1 (que es TX).

El cable naranja que es TX va conectado a WING_BLO (que es RX).

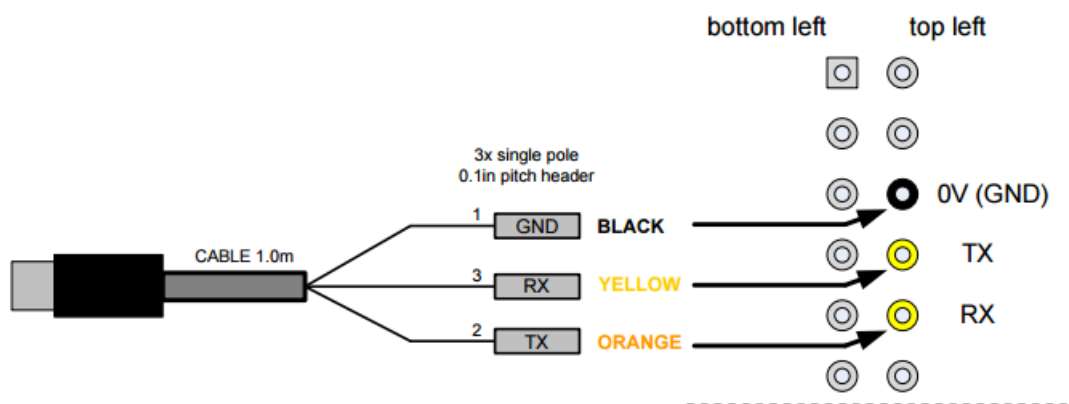


Imagen 31

El montaje del circuito quedaría así:

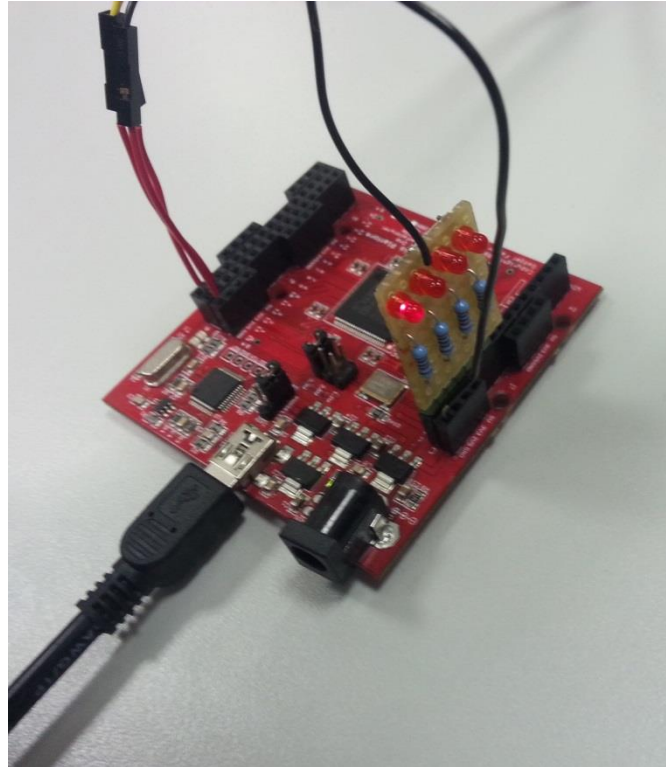


Imagen 32

12. Una vez que se ha montado el circuito y se ha escrito el programa en Design Lab. Pulso el botón **“verificar”** para ver si hay errores de programación y luego pulso el botón **“subir”** para subirlo a la placa Papilio.



Imagen 33

13. Una vez montado el circuito paso a programar en Python el código que permitirá mandarle comandos a la placa Papilio, que lo he llamado **“UART.py”**, pero antes de escribir el código hay que instalar la librería **“python-serial”** en una ventana de símbolos para que no nos de error a la hora de ejecutarlo.

```
sudo apt-get install python-serial
```

El código sería:

```
import serial  
arduino = serial.Serial('/dev/ttyUSB0', 9600)
```

```
print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando a la placa
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

arduino.close() #Finalizamos la comunicación.
```

El código es muy fácil, primero se conecta al puerto de la placa Papilio, y después mientras sea True, va leyendo los valores que se le van introduciendo al puerto serie, si es H (se enciende) y si es L (se apaga).

14. En la ventana de comandos ejecutamos la instrucción: **cd Escritorio** que es donde tengo el archivo **UART.py**.
15. Por último, ejecuto la instrucción **python UART.py** para ejecutar el programa y debería de funcionar. Al escribir “H” en la ventana de símbolos el led se debería de encender y si pulsamos “L” el led se debería de apagar.

Nota: En este cuarto ejercicio la parte de hardware no he tenido ningún problema en realizarla porque solo ha sido conectar un led y el cable TTL-RS232-USB con sus 3 conexiones, aunque al principio me equivoque al conectar el RX en donde el TX, pero lo solucioné al final. En la parte de software tampoco he tenido problemas, porque ha sido códigos como los de los ejercicios anteriores. Además este ejercicio ha sido fácil de realizar porque he seguido un tutorial, que viene paso a paso explicado todo muy bien.

Conclusión

Con la realización de estas dos prácticas de ZPUino, he aprendido a usar la placa y el entorno de programación (era muy parecido al de Arduino).

Algunas cosas que me han parecido interesantes en las prácticas son: la comunicación vía puerto serie, que ha habido que hacerlo a través de un programa en Python. También el diseño de circuitos sobre FPGA, que se hace en el programa Xilinx ISE y el ejercicio de convertir la placa Papilio en un analizador lógico.

Glosario

- **FPGA:** dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado.
- **GND (toma de tierra):** se emplea en las instalaciones eléctricas para llevar a tierra cualquier derivación indebida de la corriente eléctrica a los elementos que puedan estar en contacto, ya sea directa o indirectamente, con los usuarios de aparatos de uso normal, por un fallo del aislamiento de los conductores activos, evitando el paso de corriente al posible usuario.
- **HDL:** es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales.
- **Led:** es un diodo que emite luz.
- **Microcontrolador:** es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.
- **Reguladores de tensión:** es un dispositivo electrónico diseñado para mantener un nivel de tensión constante.
- **Resistencia:** oposición que tienen los electrones al moverse a través de un conductor.
- **SoC:** circuito integrado que incorpora gran parte de los componentes de un ordenador o cualquier otro sistema informático o electrónico.

- **Switch:** es un dispositivo que permite desviar o interrumpir el curso de la corriente eléctrica.
- **Wishbone:** es un bus para interconexión de periféricos que permite que las partes de un circuito integrado se comuniquen entre sí.

Bibliografía

- **Características de la placa**

- <http://papilio.cc/>
- <http://papilio.cc/index.php?n=Papilio.Hardware>
- <http://forum.gadgetfactory.net/index.php?/page/articles.html?p=1>
- <http://forum.gadgetfactory.net/index.php?/files/file/236-papilio-designlab-ide/>

- **Instalación del software**

- <https://coria.dte.us.es/~bellido/>
- <http://gadgetfactory.net/learn/2015/01/13/designlab-installation-guide-linux/>

- **Ejercicio 1**

- <http://gadgetfactory.net/learn/2015/05/03/designlab-make-a-simple-fpga-circuit-2/>

- **Ejercicio 2**

- <http://gadgetfactory.net/learn/2015/04/03/designlab-using-the-ide-for-the-first-time/>

- **Ejercicio 3**

- <http://gadgetfactory.net/learn/2015/07/30/designlab-using-papilio-as-stand-alone-logic-analyzer/>

- **Ejercicio 4**

- <http://gadgetfactory.net/learn/2015/05/15/designlab-make-a-custom-zpuino-system-on-chip-2/>
- http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TL-232R_RPi.pdf