

# LHD: Memoria Práctica

## ZPUINO

Alejandro González Sánchez

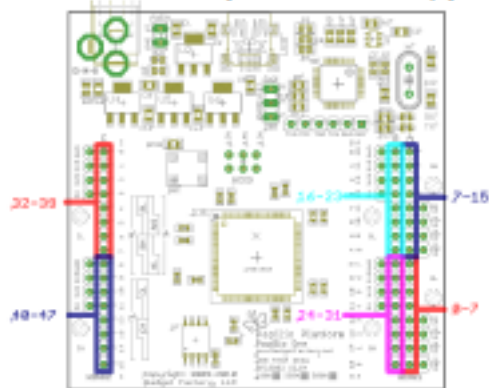
# INDICE

- 
- Objetivos ..... pág. 3
  - Actividades Realizadas ..... pág. 6
  - Opinión personal ..... pág. 13
  - Conclusiones ..... pág. 14

# OBJETIVOS

- El objetivo de esta práctica es relacionarnos con una plataforma de desarrollo basada en FPGA.
- Aprendemos a preparar el entorno de desarrollo del microprocesador ZPuino.
- Realización de ejemplos de programación sobre la plataforma Papilio/ZPuino.
- Desarrollar otros ejemplos de un control de diversos componentes hardware.

- Plataformas de desarrollo Papilio
- Son placas de desarrollo hardware abiertas basadas en FPGAs de XILINX:
  - Papilio one – SPARTAN3E (250 y 500)
  - Papilio Pro – SPARTAN6
  - Papilio duo – SPARTAN6 y atmega32U4
- WINGS: son placas de expansión adaptadas al conector de expansión de papilio
- Existen multitud de ejemplos de diseño para papilio
- ZPUino: Es un SoC implementado en VHDL (soft core) basado en el microprocesador ZPU de Zylins.
- Tanto las especificaciones del microprocesador como el diseño de ZPUino son abiertas
- Al ser diseño soft-core, se pueden añadir nuevos periféricos al SoC según las necesidades (diseño hardware en HDLs)
- Se puede trabajar con ZPUino de modo equivalente a Arduino ya que se ha adaptado el IDE (entorno de desarrollo software de arduino) para ser compatible con ZPUino. Como veremos, el entorno también permitira modificar el SoC a nivel hardware para añadir nuevos periféricos al mismo.
- Entorno de desarrollo para papilio/ZPUino:
  - Design Lab: (apariencia)
  - Como se puede ver es muy parecido al entorno de Arduino



La placa empleada en esta práctica es: Papilio one

– SPARTAN3E 500. Que tiene las siguientes características:

- Xilinx Spartan 6 LX9 FPGA
- 512KB or 2MB SRAM
- AVR ATmega32U4 chip that is a derivative of the Arduino Leonardo design.
- High efficiency LTC3419 Switching Voltage Regulator
- 64Mbit Macronix MX25L6445 SPI Flash
- 54 I/O pins arranged in an Arduino-Compatible Mega Form Factor
- High Speed USB port for programming and communication
- ZPUino Soft Processor



En la primera práctica de laboratorio se nos ha introducido el manejo y la programación de la placa papilio, led y switch, programación con pyhton, desde el pc hemos podido darle órdenes a la placa por el puerto serie.

# ACTIVIDADES REALIZADAS

- ZPUino: Es un SoC implementado en VHDL (soft core) basado en el microprocesador ZPU de Zylins. (

<http://opensource.zylin.com/zpu.htm>)

<http://papilio.cc/index.php?n=Papilio.ZPUinoIntroduction>

- Tanto las especificaciones del microprocesador como el diseño de ZPUino son abiertas

- Al ser diseño soft-core, se pueden añadir nuevos periféricos al SoC según las necesidades (diseño hardware en HDLs)

- Se puede trabajar con ZPUino de modo equivalente a Arduino ya que se ha adaptado el IDE (entorno de desarrollo software de arduino) para ser compatible con ZPUino.

Como veremos, el entorno también permitirá modificar el SoC a nivel hardware para añadir nuevos periféricos al mismo.

Entorno de desarrollo para papilio/ZPUino: → Design Lab:

También debemos instalar jre en Ubuntu

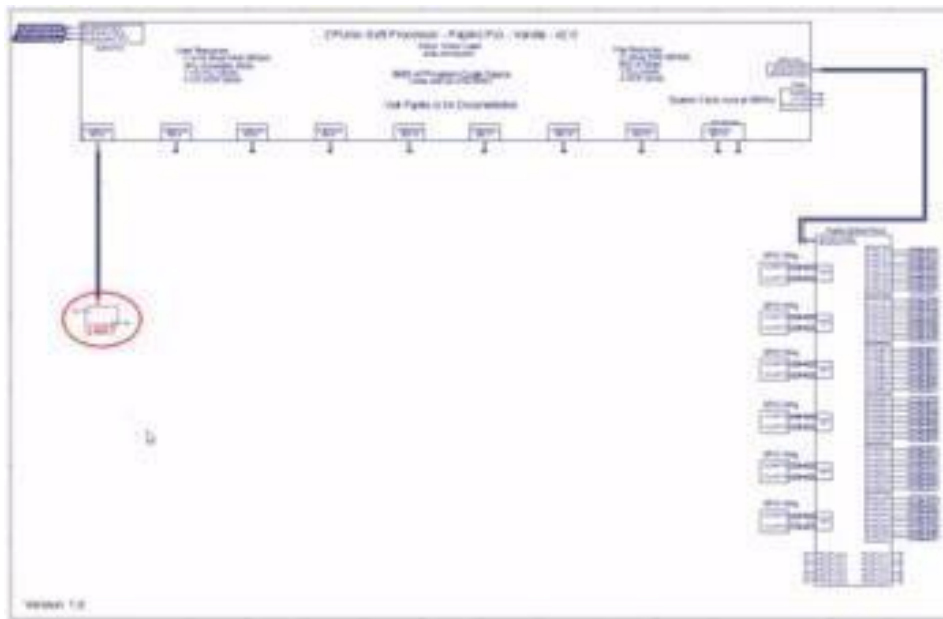
`sudo apt-get update`

`sudo apt-get install default-jre`

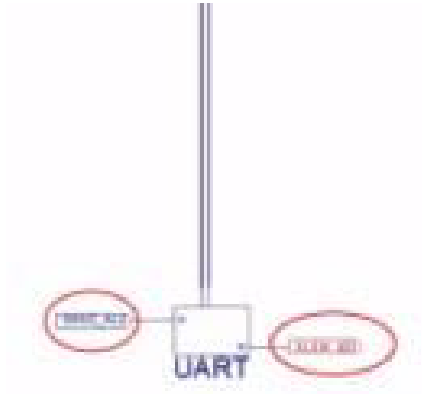
## ACTIVIDAD 1

En esta actividad vamos a editar el circuito, para ellos vamos a seguir este tutorial (<http://gadgetfactory.net/learn/2015/04/03/designlab-using-the-ide-for-the-first-time/>)

Una vez abierto el entorno de desarrollo presionamos el botón Edit Circuit,



Tenemos que hacer doble click sobre Papilio\_Pro.sch para poder abrir el editor, ahora debemos pulsar sobre el símbolo del conector de entrada y salida. Y procedemos a cambiarle los nombres. Tenemos que colocar WING\_AL0 y WING\_AL1, estos pines corresponden con los de la placa Papilio.



Una vez que tenemos todo esto hecho sintetizamos el circuito y generamos los archivos de programación.

Ahora antes de proceder a cargar el archivo en la placa, tenemos que cambiar el nombre de un archivo como nos muestra en la práctica, vamos a la carpeta donde tengamos el proyecto, entramos en circuit y después entramos en 500k. Ahora tenemos que renombrar los archivos papilio-one500k.

Carpeta: /circuit/500K/ -

1.- Borrar papilio\_one\_500k.bit -

2.- Renombrar Papilio\_One\_500K.bit a papilio\_one\_500k.bit

# FUNCIONAMIENTO CON LED, SWITCH y PYTHON

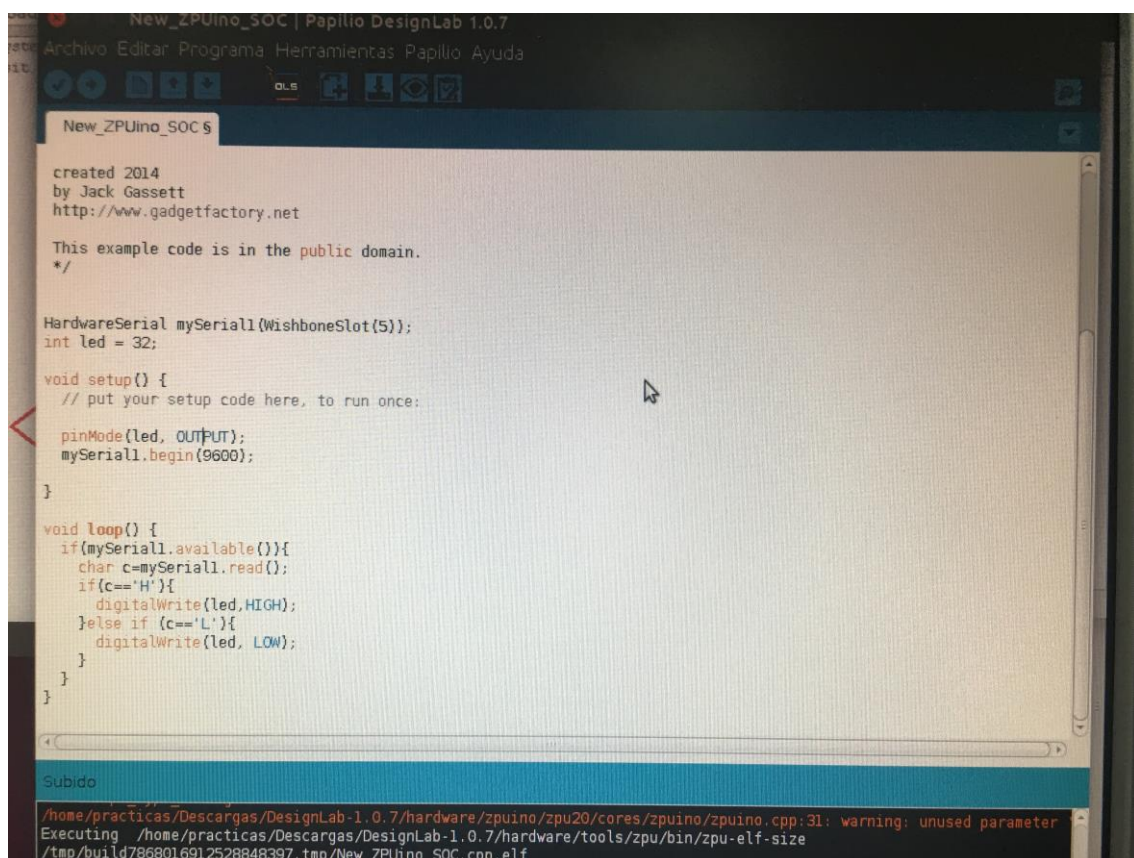
Analizar el código y conectar un LED y un Switch de la manera correcta para que el LED o parpadea o se queda encendido según conmutemos el Switch. Mostrar la salida serie (monitor serie)

Modificar el sketch para que en vez de parpadear el Led quede apagado y se añada a la línea del monitor serie la información correcta:

→ LED: Encendido → LED: Apagado

Para ello usamos este código python

*\*Modificar nuevamente para controlar el encendido apagado desde el PC via puerto serie*



```
New_ZPUino_SOC | Papilio DesignLab 1.0.7
Archivo Editar Programa Herramientas Papilio Ayuda

New_ZPUino_SOC $

created 2014
by Jack Gassett
http://www.gadgetfactory.net

This example code is in the public domain.
*/

HardwareSerial mySerial1(WishboneSlot(5));
int led = 32;

void setup() {
  // put your setup code here, to run once:
  pinMode(led, OUTPUT);
  mySerial1.begin(9600);
}

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

Subido

/home/practicas/Descargas/DesignLab-1.0.7/hardware/zpuino/zpu20/cores/zpuino/zpuino.cpp:31: warning: unused parameter
Executing /home/practicas/Descargas/DesignLab-1.0.7/hardware/tools/zpu/bin/zpu-elf-size
/tmp/build7868016912528848397.tmp/New_ZPUino_SOC.cpp.elf
```

Y en Arduino este es el código.



```
zpu.py x
import serial

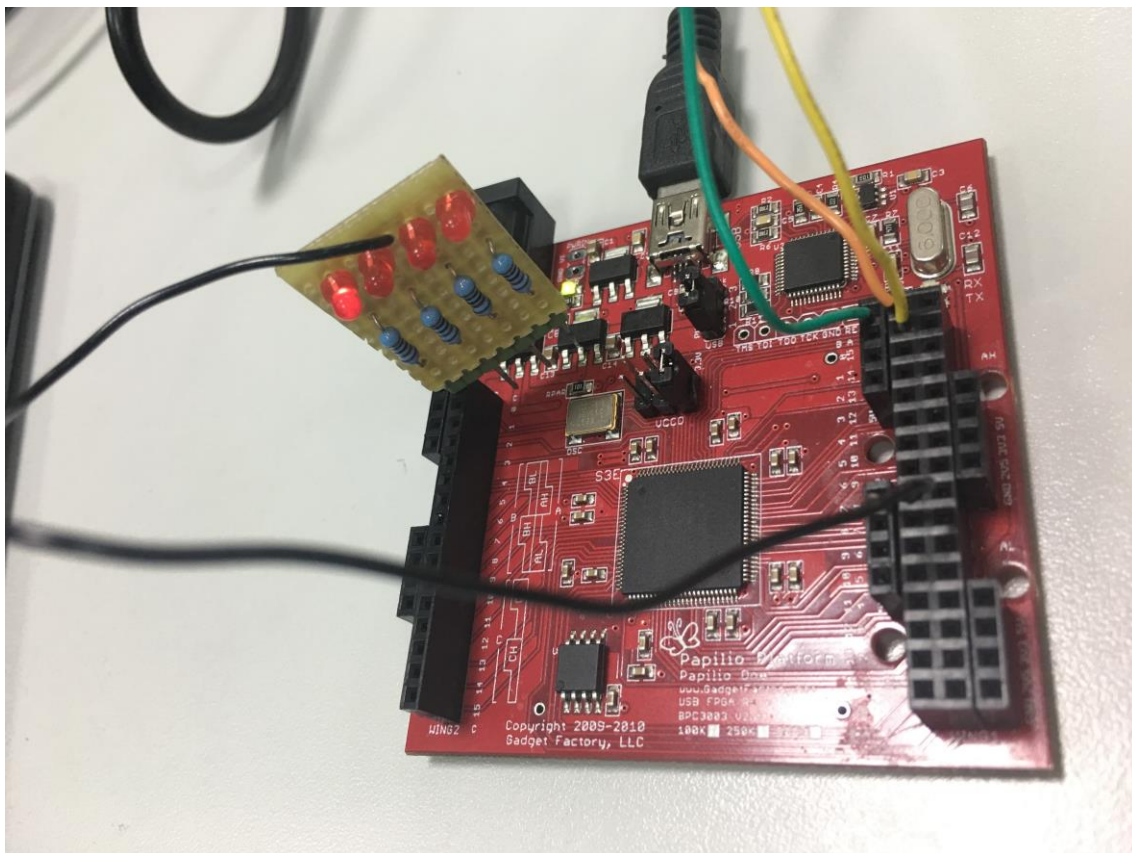
arduino = serial.Serial('/dev/ttyUSB0', 9600)

print("Starting!")

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

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

Python Anchura de la pestaña: 8 Ln 3, Col 38 INS



# USO DE PAPILIO EN ANALIZADOR LOGICO

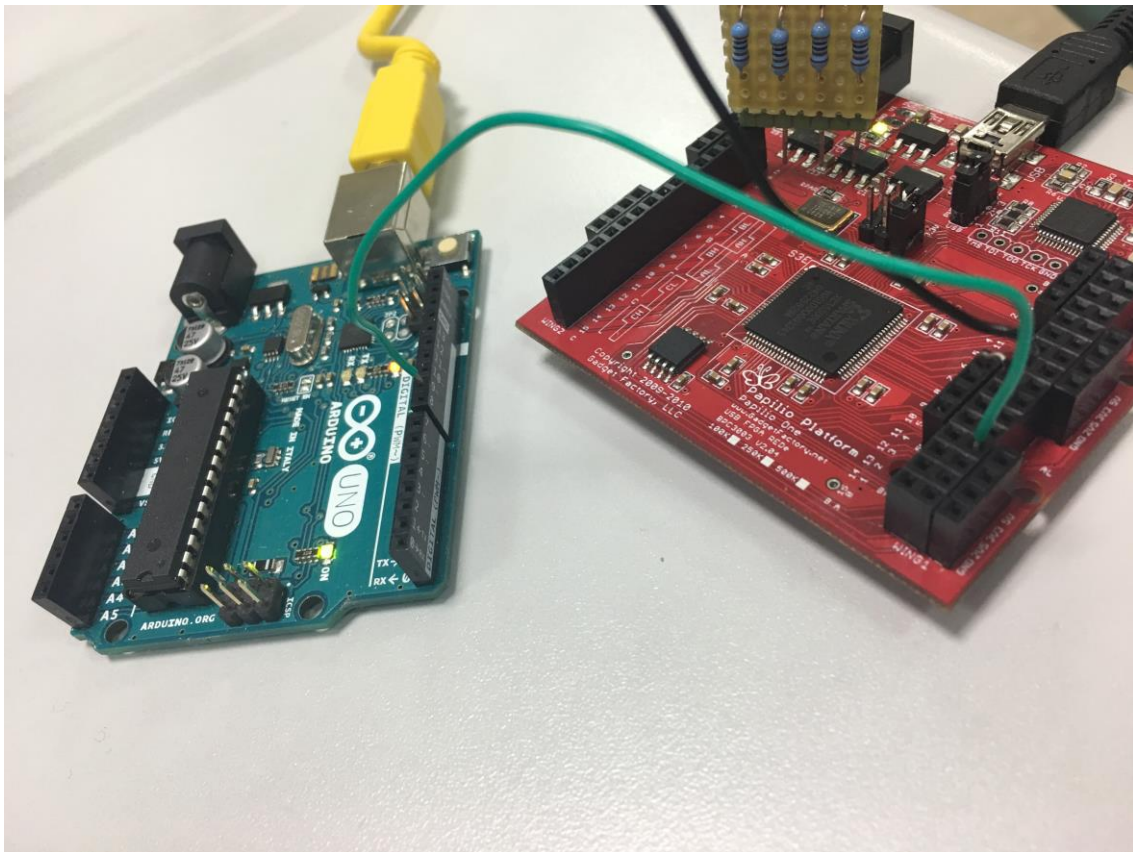
Para poder realizar esto tenemos que seguir un tutorial dado por el profesor.

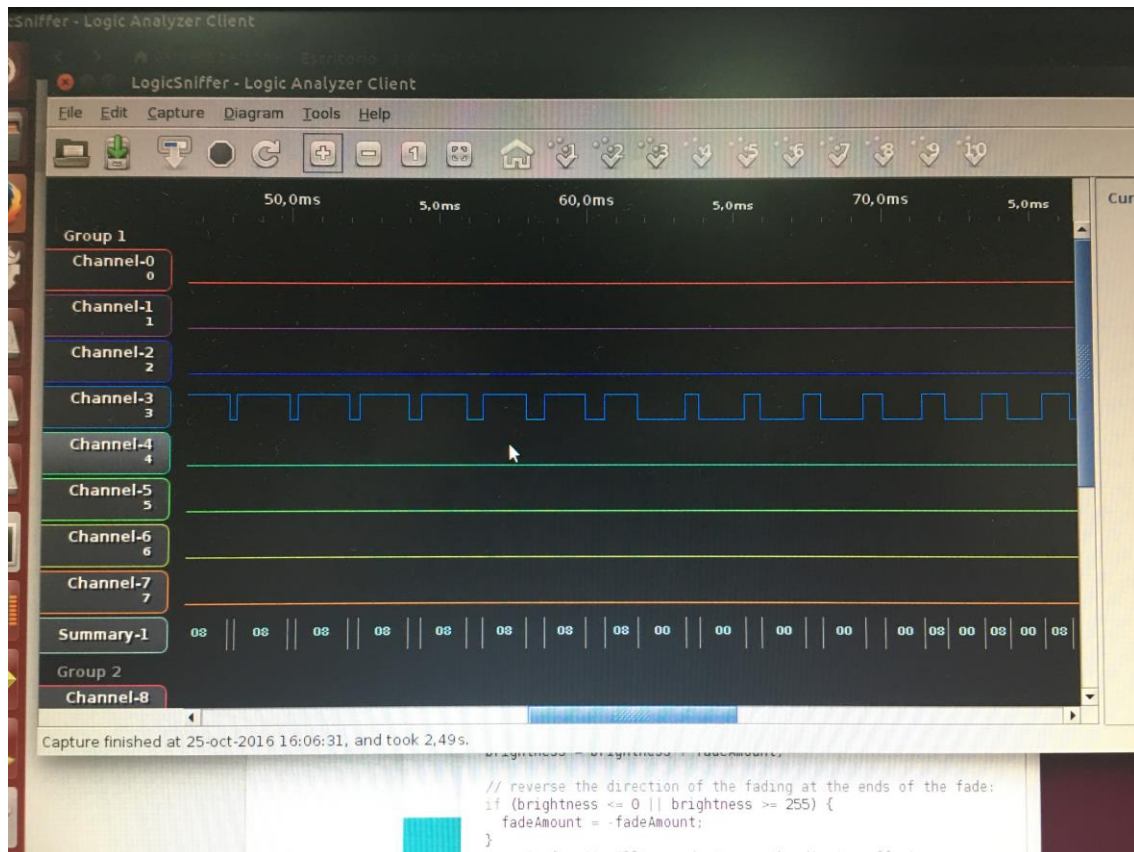
<http://gadgetfactory.net/learn/2015/07/30/designlab-using-papilio-as-stand-alone-logic-analyzer/>

- Descargar: <https://10.1.15.75/~bellido/ols-0.9.7.2-full.tar.gz>
- Extraer y ejecutar el script run.sh

→ \$ cd

→ \$ . run.sh





Vemos el analizador lógico la cual la señal esta generada en Arduino mediante PWM, por eso va cambiando el tamaño del pulso.



**Una vez que tengamos la UART disponible en los Pines WA0 y WA1 debemos conectarnos al PC con el cable TTL-RS232-USB**

**Vamos a seguir este tutorial.**

**[http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS\\_TTL-232R\\_RPi.pdf](http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_RPi.pdf)**

- Mostrar que funciona correctamente la UART añadida haciendo el siguiente ejemplo de SKETCH:
- Controlar encendido/apagado de un LED desde el PC a través de la nueva UART
- Este es el cable UART que vamos a usar en las prácticas.



### **Evaluación después de la actividad**

Tras la realización de la práctica, hemos aprendido a usar el entorno de desarrollo de arduino pero esta vez junto a papilio, como diseñar SOC y como poder usar la placa como analizador lógico. También hemos creado un nuevo método de conexión mediante el cable UART para poder comunicar la placa con otros dispositivos.

### **Incidencias**

- Al no tener conocimiento previo de la materia, alto coste de entendimiento de los socs, pero gracias al tutorial, se ha podido desarrollar entera y sin problemas.

### **Valoración personal**

Muy práctica e interesante pero demasiado rápido todo como para poder asimilar bien los conceptos, poder ir apuntando cosas o disfrutar más al hacerlo mas tranquilo.

# OPINION PERSONAL

Esta práctica me ha costado mucho más trabajo que las que realizamos anteriormente con Arduino, nos ha estado dando problemas a la hora de cambiar los archivos de nombre y cargarla, Arduino es mucho más sencillo en este aspecto porque empiezas a programar directamente, es verdad que esta placa es mucho más versátil, también tiene muchos más pines y eso hace que podamos conectar muchos más periféricos.

En general vuelvo a reiterarme como en ocasiones anteriores, que estas prácticas se deberían hacer en más asignaturas, son prácticas que tú te enfrentas a problemas reales de la vida, y puedes ver resultados tangibles, y eso hace que el alumno avance mucho más que solamente a base de teoría, y más en nuestra titulación.

# CONCLUSIONES

Siendo mucho más compleja que las anteriores de Arduino, esta hemos podido trabajar con FPGA de otra forma a como trabajábamos en DSD en la asignatura de segundo. Aquí hemos hecho cosas mas interesantes sobre todo cuando hacemos la combinación Arduino FPGA y conectamos un sistema con otro. Nunca antes habíamos hecho el diseño de un SOC, y junto a la parte de ver el analizador lógico, me ha parecido muy interesante y útil.

Sí que es verdad que al principio cuesta coger el ritmo porque todo es nuevo, pero se aprende rápido a usarla y a entender el contenido de la placa y como usarlo.