

*LABORATORIO DE  
DESARROLLO DE HARDWARE  
PAPILIO "FPGA"*

## MEMORIAS

2016/2017

Rafael arroyo  
alemán

Ingeniería Informática de  
Computadores



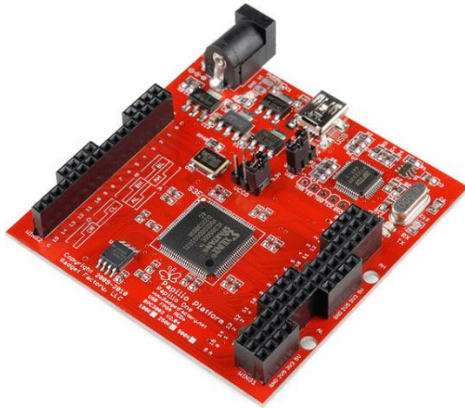
Escuela Técnica Superior de  
**Ingeniería Informática**

# ÍNDICE

Introducción .....	2
Descripción de la actividad .....	5
Instalación .....	5
Inversor .....	8
➤ Incidencias y anécdotas.....	11
Zpuino.....	11
Encendido y apagado de led por serial .....	12
Rediseñando el SoC. Añadiendo periféricos .....	14
➤ Evaluación después de la actividad.....	20
➤ Incidencias y anécdotas.....	20
Analizador Lógico .....	20
➤ Evaluación después de la actividad.....	23
➤ Incidencias y anécdotas.....	23

# Introducción

En estas prácticas nos vamos a centrar en la plataforma de desarrollo Papilio.



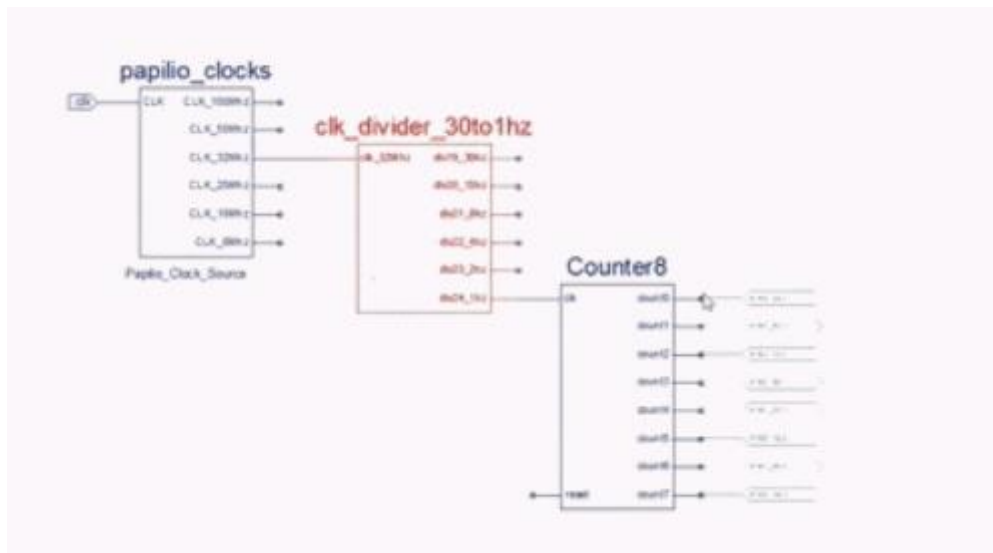
Estamos ante un proyecto de hardware y software de código abierto que pone a nuestra disposición la posibilidad de crear hardware sobre FPGA's "Spartan 3E".

El desarrollo se lleva a cabo a través del DesignLab el cual está basado en el IDE de Arduino.

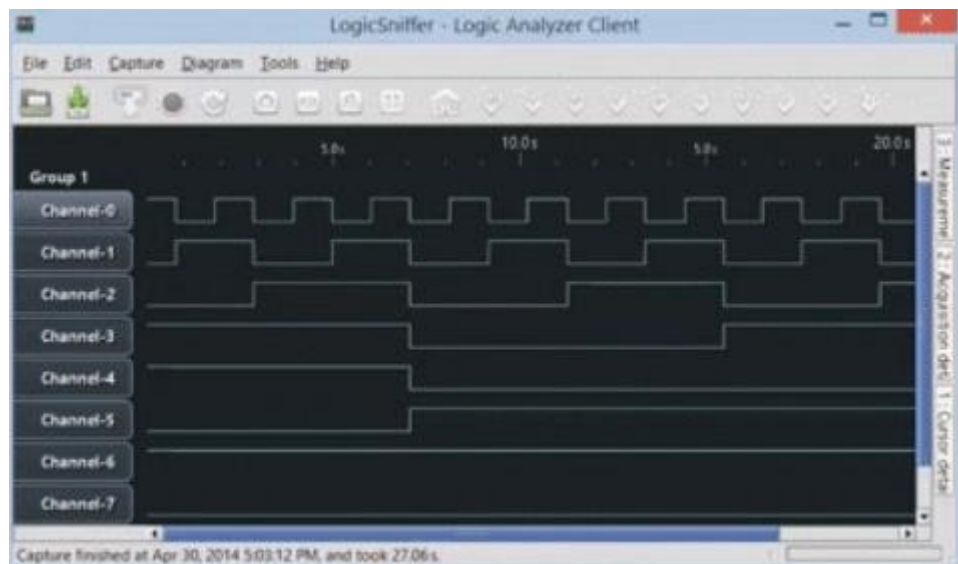
Va a ser un entorno de desarrollo muy familiar para el que está acostumbrado a programar con el entorno de Arduino pero con la diferencia de que en éste tenemos otras opciones de desarrollo de diseño de circuitos, con enlace al entorno de ISE Xilinx, analizador lógico, revisión de esquemas que no se encuentran en el IDE de Arduino ya que estamos ante la posibilidad de poder desarrollar nuestros propios circuitos de hardware.



A través de esta herramienta existe la posibilidad de ejecutar el Xilinx ISE el cual nos permite “Editar” circuitos utilizando el editor de esquemas junto con la biblioteca de circuitos Papilio.



También dispone de la posibilidad de usar la placa Papilio como analizador lógico para usarlo sobre cualquier circuito para saber en cada instante que es lo que se está haciendo con posibilidad de poder elegir el número de canales que se va a usar y la velocidad de captura.



La posibilidad de implementar Systems On Chip “Soft Core” en la FPGA, en el cual el DesignLab incluye el Soft Core Zpuino el cual usaremos para nuestras prácticas.



Este está implementado con un bus Wishbone, el cual nos va a dar la posibilidad de interconectar cualquier circuito periférico.

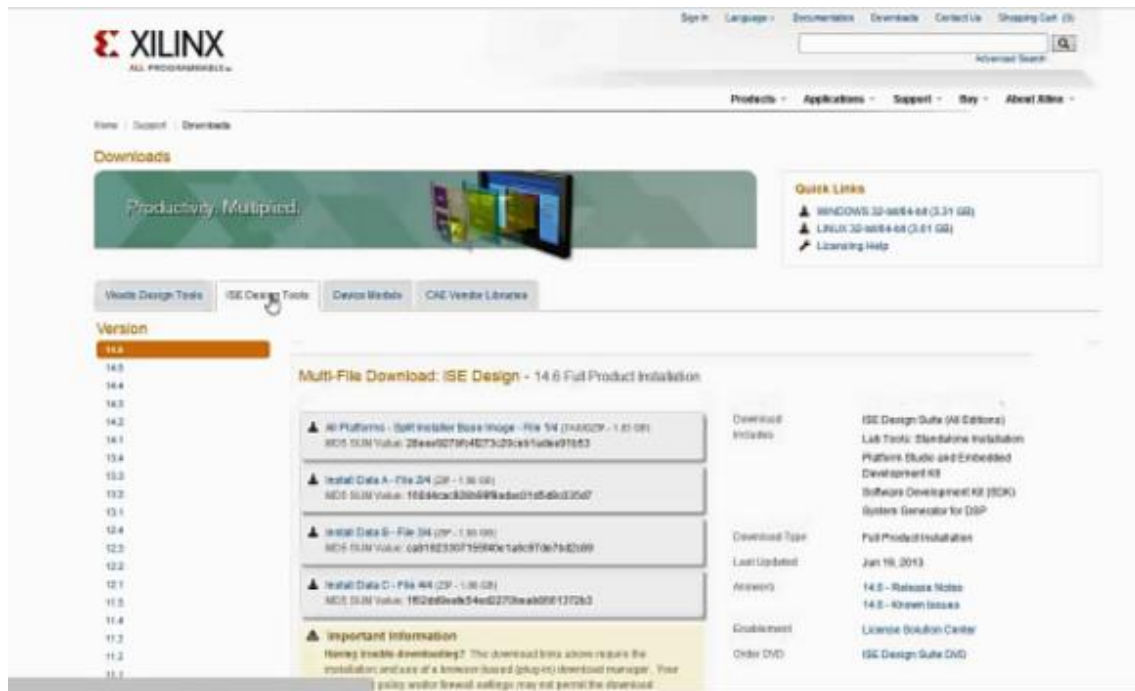
Pero la base de todo es que podemos implementar y desarrollar hardware con un coste muy bajo gracias a la FPGA.

# Descripción de la actividad

## Instalación

El primer paso para empezar a utilizar la placa de desarrollo Papilio va a ser la instalación del Design Lab junto con el Xilinx ISE.

El primer prerequisite si vamos a querer editar circuitos tenemos que tener el Xilinx ISE instalado, para ello nos vamos a la página de Xilinx nos registramos y nos descargamos la versión Full que ocupa unos 6 GB.



Una vez instalado nos descargamos el Papilio DesignLab IDE la última versión, que en nuestro caso lo ejecutaremos sobre Linux con la versión de 32 o 64 bits según queramos.

Para ello tan solo tenemos que descomprimir él .tar y ejecutar el archivo:

- “sudo ./Ubuntu-setup.sh”

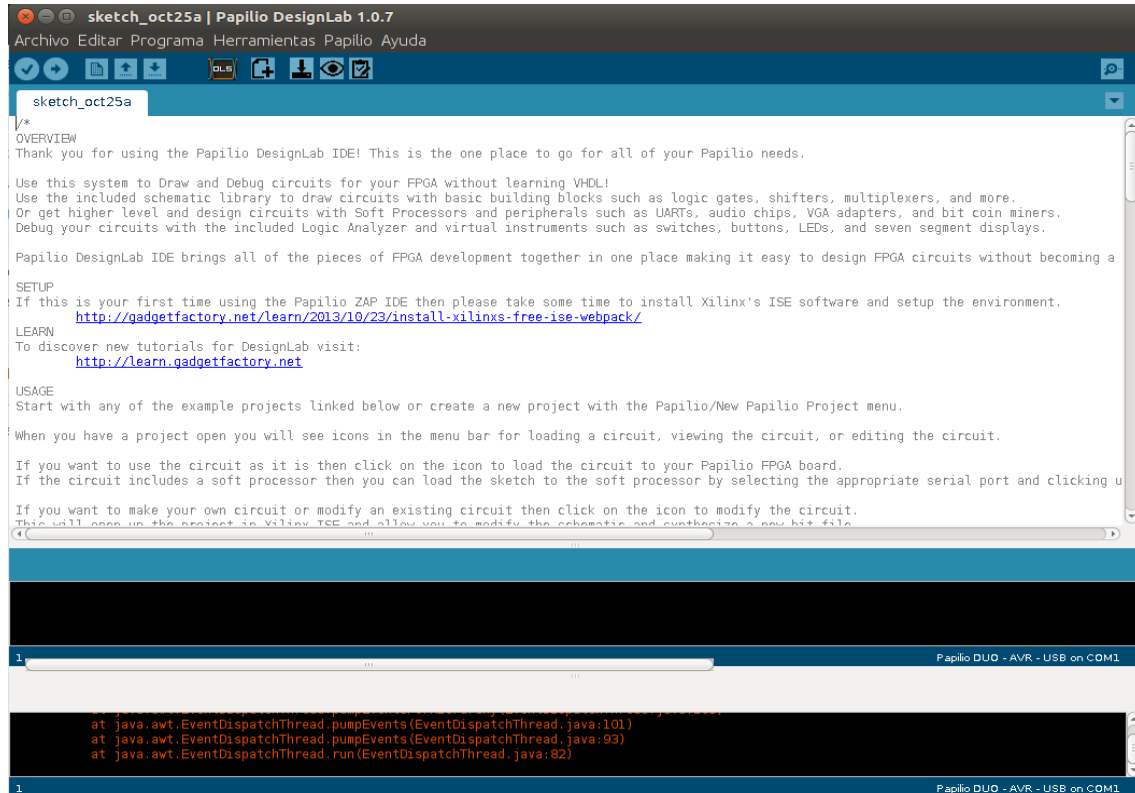
Una vez hallamos terminado la ejecución de éste, el DesignLab requiere de Java en cuyo caso no tengamos instalado un JRE o JDK debemos instalarlo ejecutando en la terminal:

- sudo apt-get install default-jre

Ya habiendo finalizado estos pasos podemos proceder a ejecutar la aplicación de DesignLab:

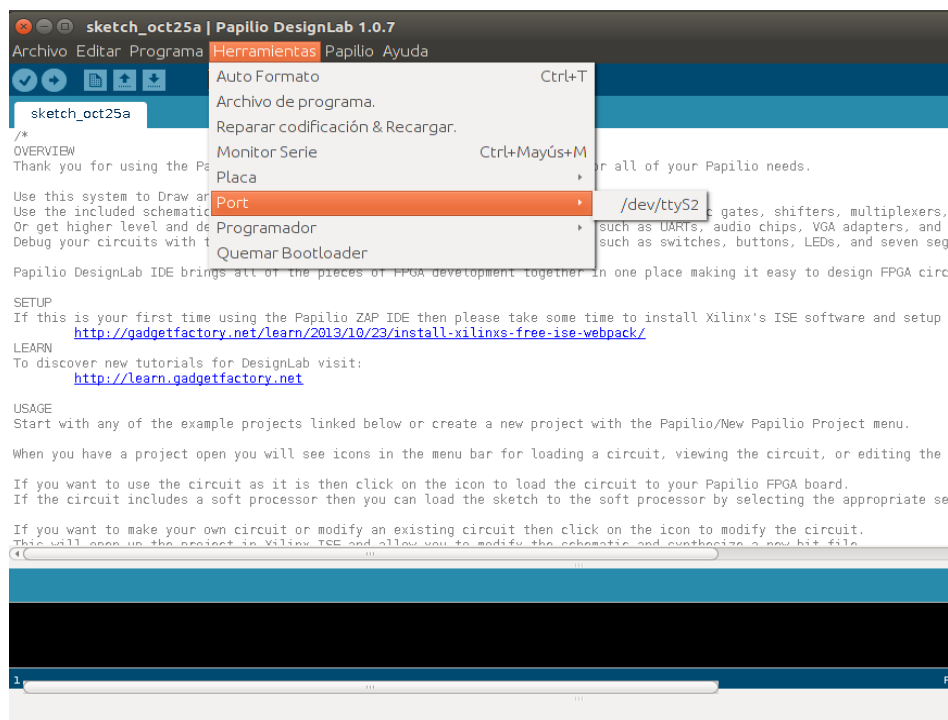
- ./DesignLab

Se nos abrirá el programa.



Y ya podemos proceder e probar nuestra placa en él, para ello conectamos la placa al usb y comprobamos que tengamos configurada la placa correcta que corresponda y el puerto serie (el cual puede variar dependiendo el pc) pero debe tener un formato parecido a este:

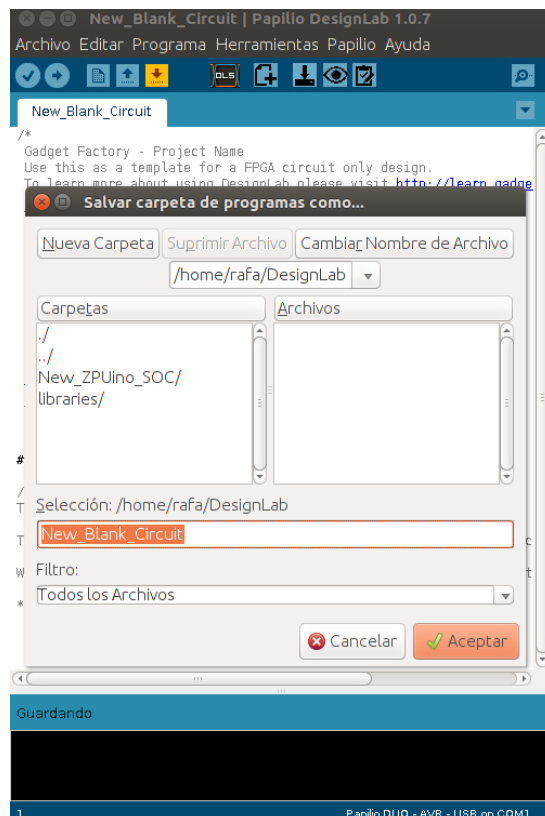
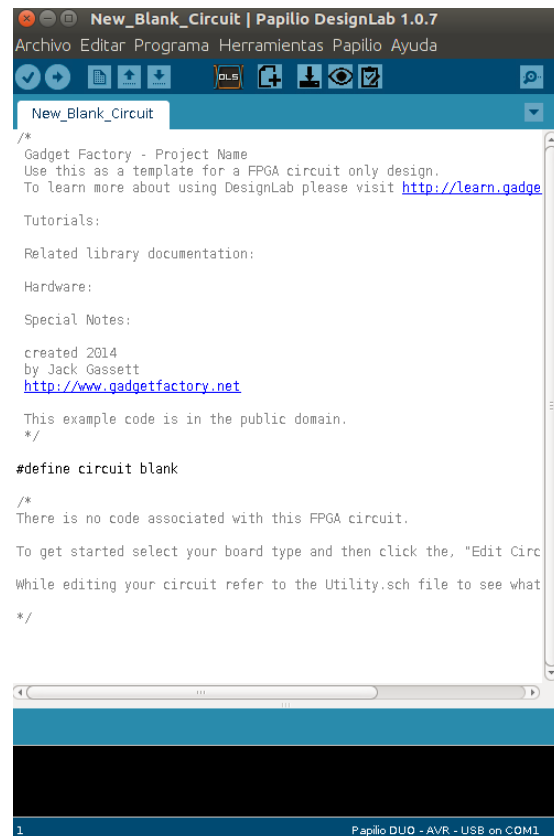
- /dev/ttyUSB1



Con ello ya podemos ponernos a diseñar circuitos básicos sobre la FPGA. Para ello hacemos click en Nuevo circuito FPGA:



Entonces se nos abre una ventana nueva en blanco:



Guardamos el proyecto donde nosotros queramos, pero **cambiando el nombre** (New\_Blank\_Circuit) por defecto que se nos coloca, esto es muy importante ya que así nos quitaremos de problemas a la hora de ejecutar el ISE Xilinx.

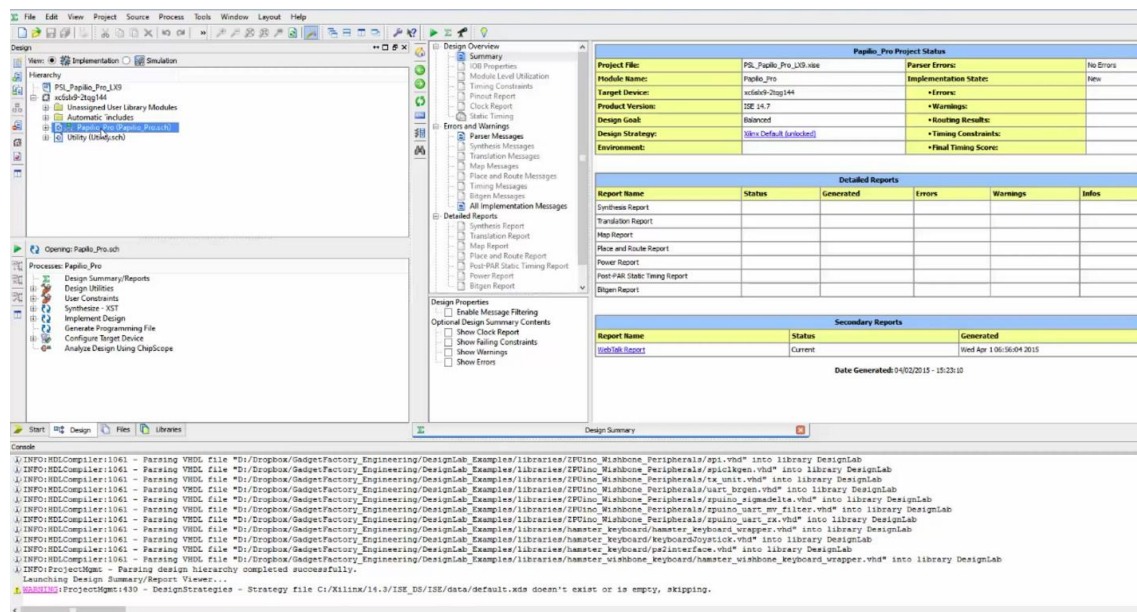


Y ya podemos editar el circuito, con lo que se nos abrirá el ISE de Xilinx.

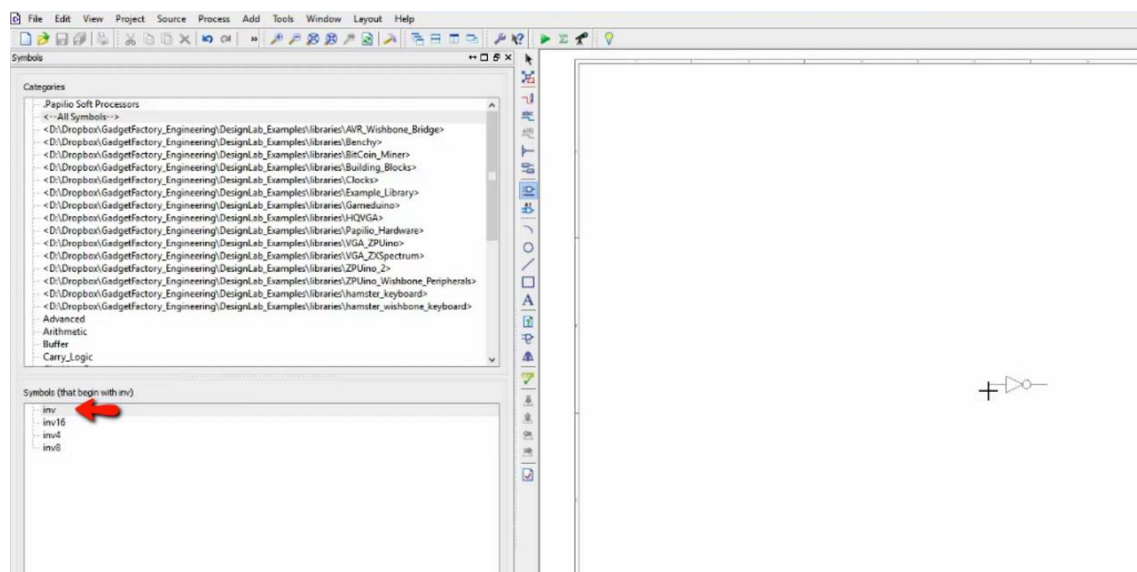


## Inversor

En él haremos una implementación de un inversor, lo primero que haremos será irnos en busca de un archivo llamado (Papilio ... .sch) y haremos doble click sobre él.



Y se nos abrirá el editor de esquemáticos en el que empezaremos haciendo click sobre el botón de símbolos para buscar el icono de inversor.



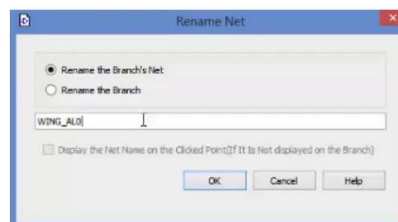
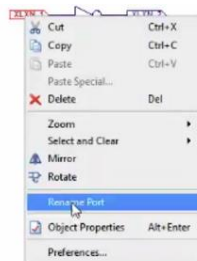
Tan solo se selecciona y se coloca sobre el editor de esquema y tendremos la posibilidad de elegir a que Wing lo vamos a conectar,



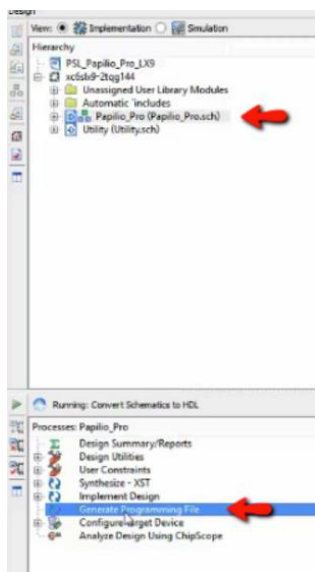
para ello tenemos que agregar la etiqueta de conector entrada salida y elegir el que nosotros queramos, para nuestro



caso por ejemplo lo hacemos al WING\_BLO y WING\_ALO haciendo click sobre la etiqueta y dándole a **renombrar puerto**



De esta manera queda listo nuestro circuito y tan solo nos queda sintetizar el archivo .bit, para ello volvemos a la ventana inicial y hacemos click sobre (Papilio ... .sch) y en la parte de debajo



le damos a generar archivo de programación, y llegamos a ver los check verdes con un mensaje diciendo “archivo de programación generado correctamente” ya tendremos todo el proceso de diseño del circuito hecho.

### **NOTA:**

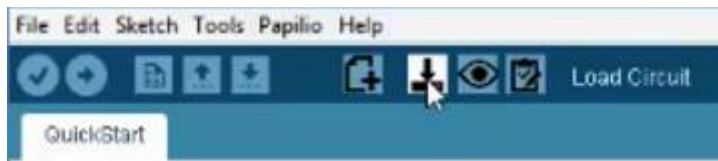
Se ha de tener en cuenta que antes de cargar el archivo, se debe **Eliminar en la carpeta <proyecto>/circuit/500K/:**

- **papilio\_one\_500k.bit**

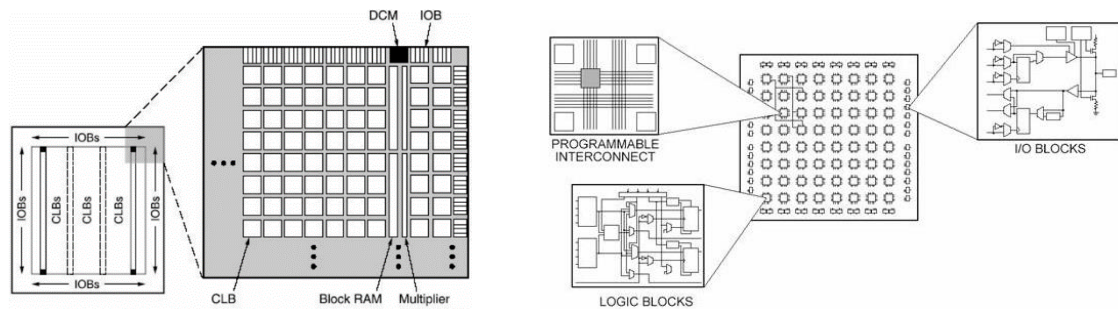
**y Renombrar:**

- **Papilio\_One\_500K.bit a papilio\_one\_500k.bit**

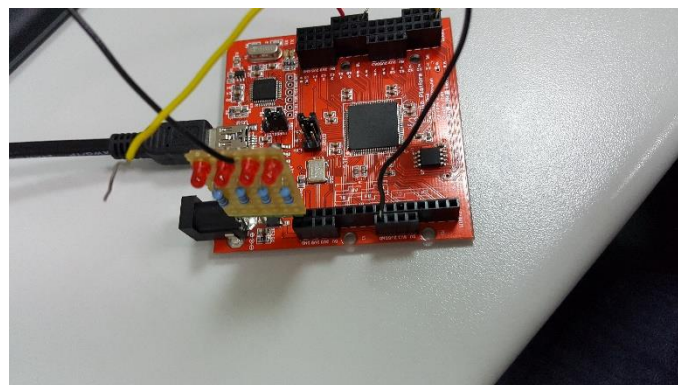
Llegados a este paso tan solo nos queda cargar el .bit sobre la FPGA, esto se hace volviendo al DesignLab y haciendo click sobre Load Circuit.



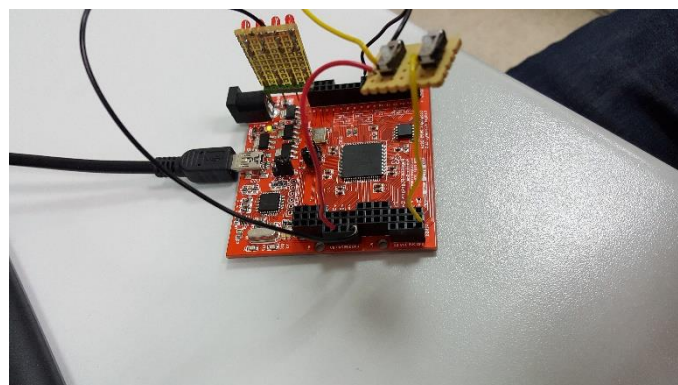
De esta forma se realizarán las interconexiones para implementarlo.



Una vez se haya realizado con éxito la configuración de la FPGA se ha realizado la conexión de un led a través del inversor con un interruptor.



**Perspectiva 1**



**Perspectiva 2**

Demostración del resultado:

[https://1drv.ms/v/s!Ao2\\_30mhw3bivQQMJ36Hn9n8ebg4](https://1drv.ms/v/s!Ao2_30mhw3bivQQMJ36Hn9n8ebg4)

➤ Incidencias y anécdotas

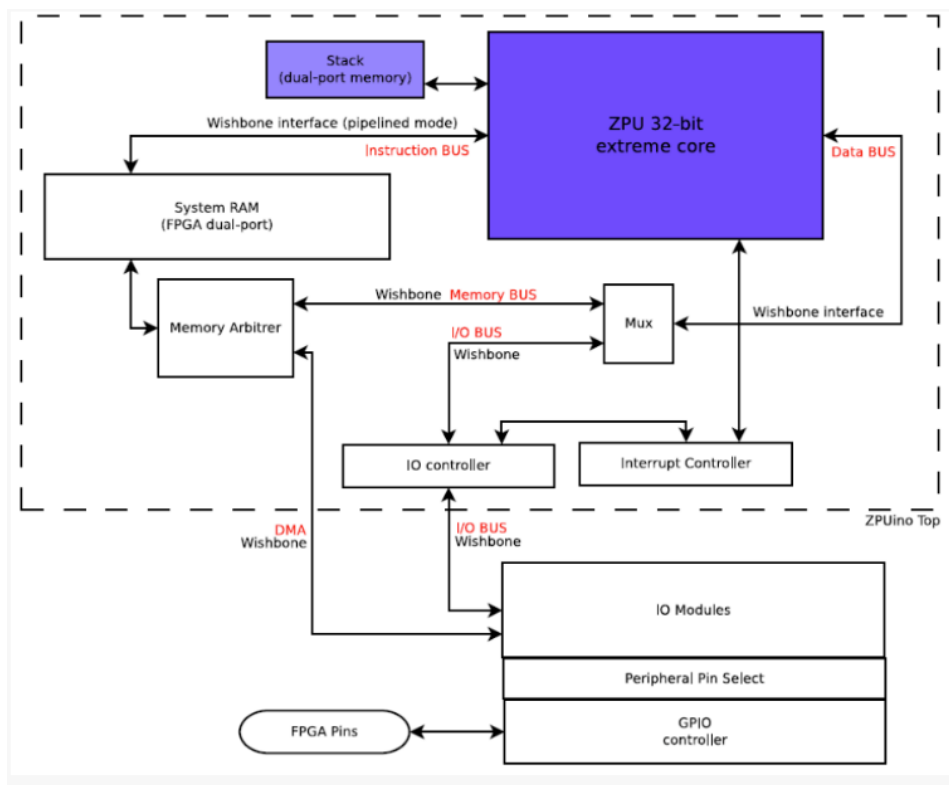
- Reconocimiento de los Wings en la placa Papilio, ver bien cual lado es cual

## Zpuino

ZPUINO es un procesador de 32 bits “Soft” que corre a unos 96 Mhz con una librería de periféricos Wishbone, cualquier cosa es controlado por un sketch y al estilo de las librerías de Arduino.

Estamos ante un ZPU SoftCore, esto quiere decir que es sintetizable a nivel de transferencia de registros flexible a la hora de implementarlo en diferentes FPGA's y adaptarlo.

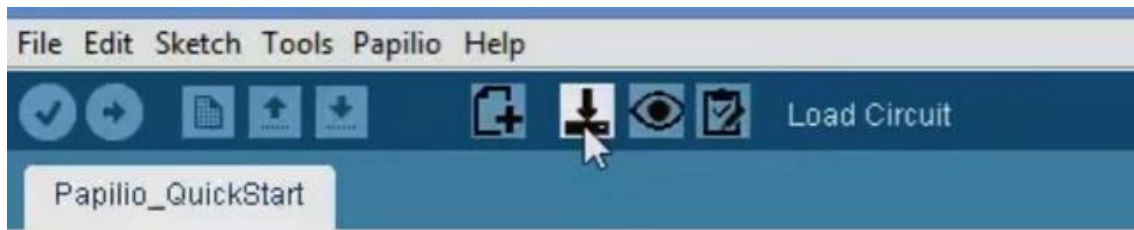
### Estructura de ZPUINO(Álvaro Lopes)



La placa Papilio usa un esquema de numeración llamada “Wing” la cual es agrupada en grupos de 16 u 8 bits, por ejemplo, para un esquema de 16 bits el esquema sería este:



Una vez seleccionado esto procedemos a cargar el circuito sobre la FPGA dándole a Load Circuit.

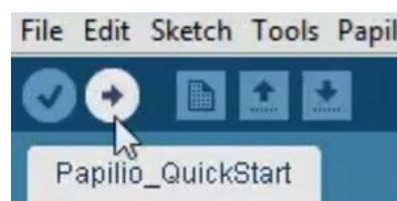


Y comprobamos que se halla cargado correctamente, podemos observar el circuito si queremos con el botón a la derecha de Load Circuit, por si queremos revisar las conexiones, también podemos editarlo (para añadir puertos, por ejemplo).

En este caso lo que vamos a hacer es probar el ejemplo de encendido y apagado del led vía serial, para ello en la parte principal de escribimos el código y lo cargamos:

```
1 int led = 8;
2 void setup () {
3   pinMode(led, OUTPUT); //LED 8 como salida
4   Serial.begin(9600); //Inicializo el puerto serial a 9600 baudios
5 }
6
7 void loop () {
8   if (Serial.available()) { //Si está disponible
9     char c = Serial.read(); //Guardamos la lectura en una variable char
10    if (c == 'H') { //Si es una 'H', enciendo el LED
11      digitalWrite(led, HIGH);
12    } else if (c == 'L') { //Si es una 'L', apago el LED
13      digitalWrite(led, LOW);
14    }
15  }
16 }
```

### Ejecución de carga de Sketch



### Código Arduino



Una vez cargado, ejecutamos el código Python → Python ejemploPython.py

### Código python

```
import serial

arduino = serial.Serial('COM3', 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
```

Y ya tendremos nuestro ejemplo apagando y encendiendo el led.

El siguiente ejemplo que realizaremos será el mismo, pero en lugar del puerto serial del MiniUsb lo haremos a través de un puerto Serial que nosotros añadiremos a ZPUINO.

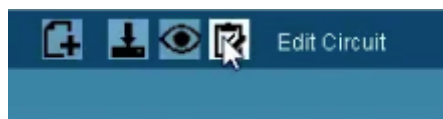
### Rediseñando el SoC. Añadiendo periféricos

Para este ejemplo desarrollaremos un ejemplo básico de como añadir un periférico al SoC de ZPUINO, hacer la síntesis, generar el bit file, programarlo en la FPGA y utilizarlo desde un sketch.

Para ello con el DesignLab abierto vamos a hacer click sobre nuevo proyecto con la tecla **control** pulsada para crear directamente un proyecto ZPUINO SoC:

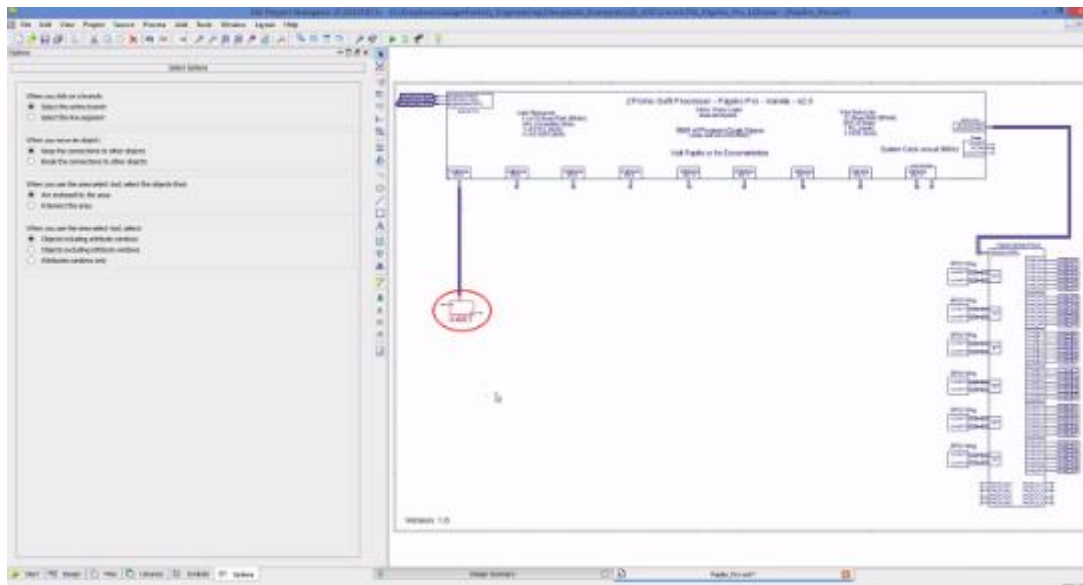


Y cuando ya nos aparezca la nueva ventana guardamos el proyecto con un **nombre distinto** al que se pone por defecto y le damos click a editar circuito:



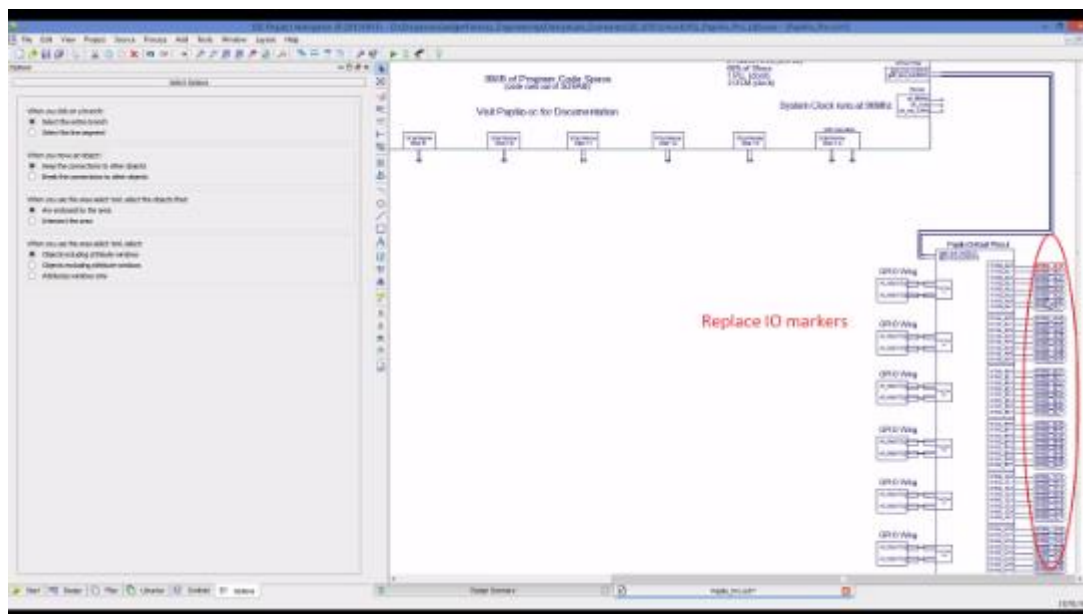
Y se nos abrirá el ISE de Xilinx donde hacemos doble click sobre "Papilio\_Pro.sch" para que se nos abra la ventana de esquemáticos.

Entonces buscaremos en la ventana de símbolos por COM\_zpuino\_wb\_UART para añadir un nuevo puerto serie y lo **enlazaremos** con el slot Wishbone 5 por ejemplo:



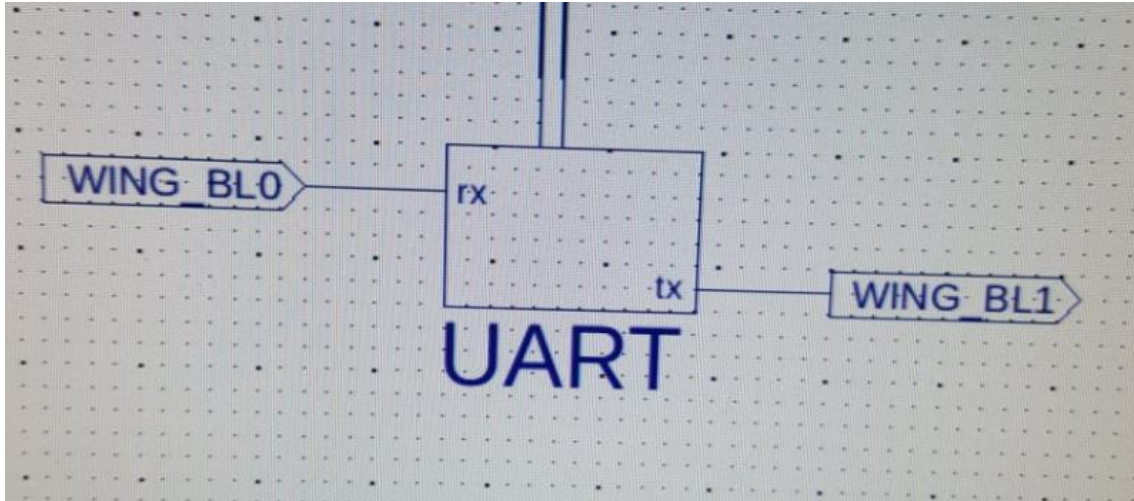
Y eliminaremos 2 Input / Output en la que elegiremos en este caso el:

- WING\_BLO
- WING\_BLI





para que sean las de nuestro nuevo puerto serie las cuales colocaremos como Tx y Rx en la nueva UART que queremos implementar, esto lo haremos renombrando los puertos de I/O de la misma.



Ya una vez echo esto tenemos nuestro puerto serie incorporado, tan solo nos quedará volver a la ventana principal del ISE.

Haremos click sobre Papilio\_Pro (Papilio\_Pro.sch) y en la parte de abajo doble click sobre generar archivo de programación y esperamos a que todo el proceso se haya chequeado en verde y podamos leer **“Proceso de generación de archivo de programación completado correctamente”**.

```

Tutorials:

Related library documentation:

Hardware:

Special Notes:

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

This example code is in the public domain.
*/

HardwareSerial mySerial1( WishboneSlot(5) );

// #define circuit ZPUino_Vanilla

int led = 13;

void setup() {
  // put your setup code here, to run once:

  pinMode(led, OUTPUT);

  mySerial1.begin(9600);
}

```

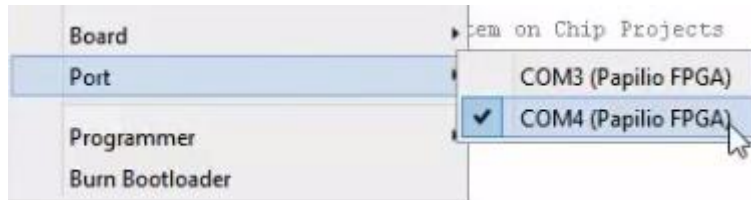
Ya entonces podemos volver a la ventana de sketch en la cual deberemos añadir :

- HardwareSerial “Nombre” (WishboneSlot (“Nº Slot”) ) ;
- “Nombre”.begin(9600);

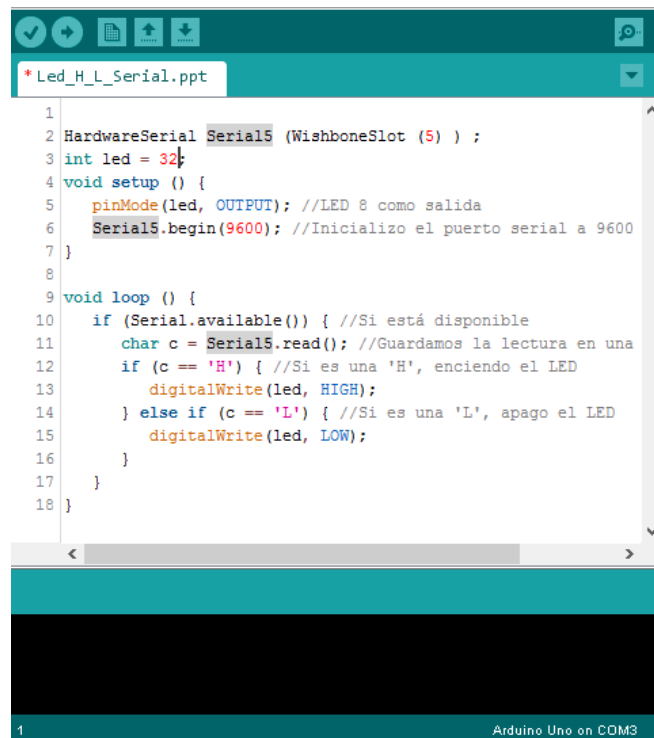
Ya por ultimo cargamos el circuito modificado en la FPGA:



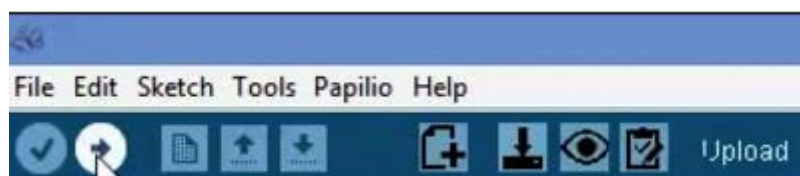
Y subimos el archivo a ZPUINO, para esto antes comprobamos que tengamos la **placa** correspondiente a la nuestra seleccionada correctamente y el **puerto** Serial.



## Código



Click en UPLOAD



Y tendremos nuestro circuito personalizado cargado y funcionando.

## Código Python a ejecutar para encender y apagar Led

```
import serial

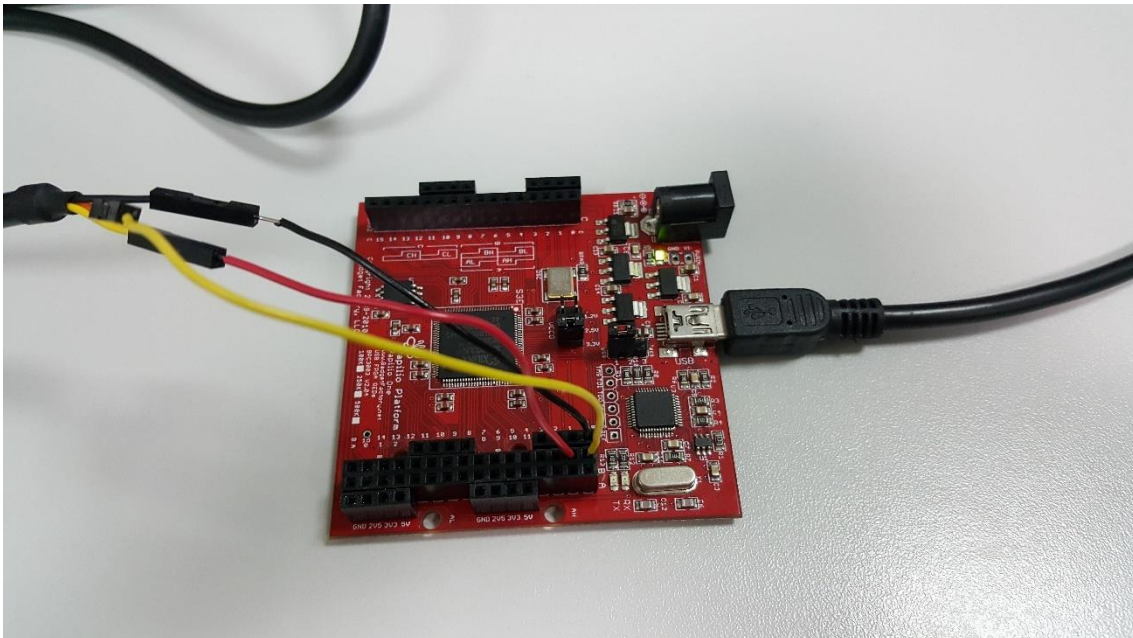
arduino = serial.Serial('COM3', 9600) // AQUI PONEMOS EL VALOR DEL SERIAL RECIEN CREADO

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
```

Cables RS232 USB a UART TTL conectada a papilio.



Se debe prestar atención a la hora de realizar el conexionado de los cables Tx y Rx del USB a la placa Papilio,esto es:

**4.2 TTL-232R-RPi Debug cable Signal Descriptions**

Header Pin Number	Name	Type	Colour	Description
1	GND	GND	Black	Device ground supply pin. Connect to ground pin on RPi board
2	TXD	Output	Orange	Transmit Asynchronous Data output. Connect to RXD input on RPi board
3	RXD	Input	Yellow	Receive Asynchronous Data input. Connect to TXD output on RPi board

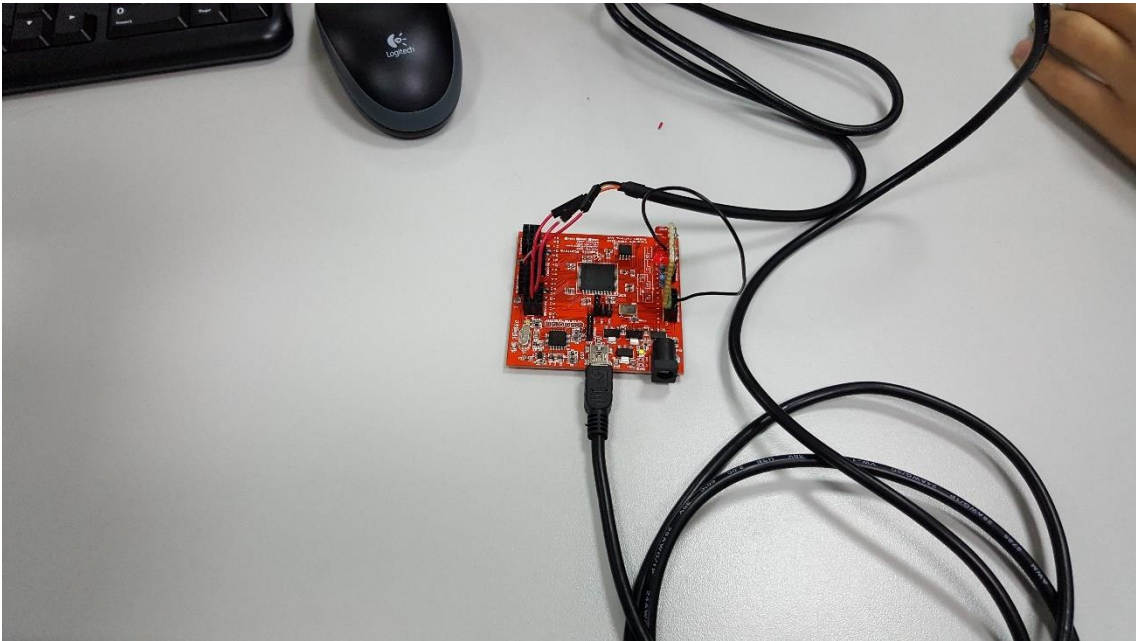
**Table 4-1 – TTL-232R-RPi Debug cable Signal Descriptions**

**NOTA IMPORTANTE**

*El cable Tx del USB debe ir al Rx de la papilio*

*El cable Rx del USB debe ir al Tx de la papilio*

Aquí tenemos un ejemplo de carga de led por el puerto **Serial** recién creado y que responde a **nuestros comandos** del programa Python.



➤ Evaluación después de la actividad

Hemos logrado apreciar la importancia de los diseños Soft Core y de como de lo que ya existe se puede reaprovechar para adaptarlo a nuestras necesidades.

➤ Incidencias y anécdotas

- Prestar especial atención en el programa Python a la hora de configura el puerto Serial y colocar el mismo nombre de aquel que hallamos creado

### Analizador Lógico

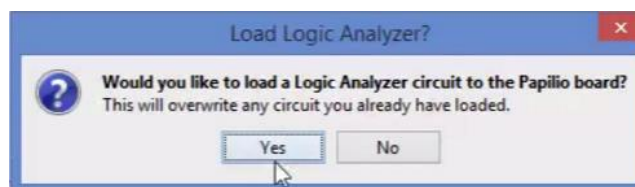
Ya para finalizar nuestras prácticas con papilio hacemos uso de la aplicación que incluye el DesignLab para usar el analizador Lógico.

Para ello abrimos el designLab, comprobaremos que tenemos correctamente referenciada nuestra placa y nuestro puerto serie seleccionado e iremos a hacer click en Analizador Lógico:

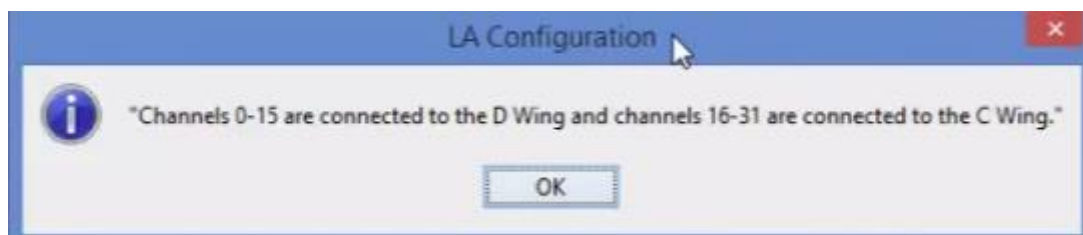


Nos aparecerá un mensaje el cual nos preguntará si queremos sobrescribir el circuito que se puede encontrar cargado en la placa papilio.

Le damos que si:

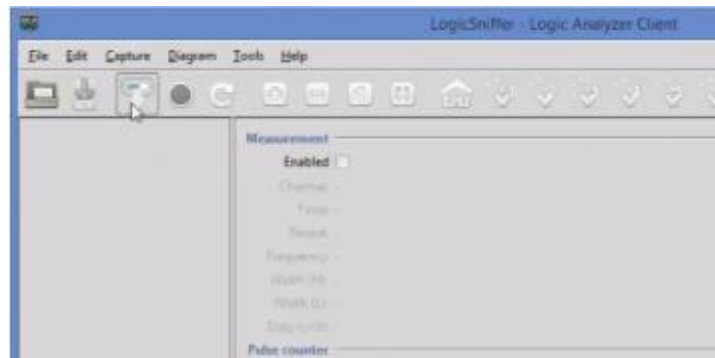


Después un archivo .bit será escrito y veremos un mensaje que nos indique cuales canales sern conectados:

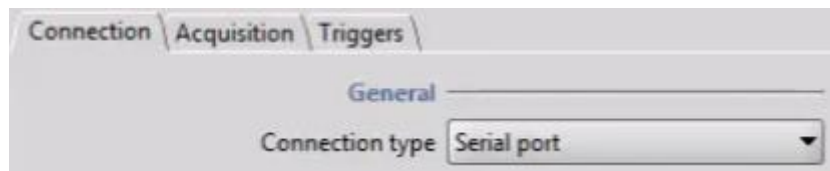


Y ya podremos utilizar el analizador para capturar datos, pero antes tenemos que hacer una breve configuración sobre dicho programa:

Damos click sobre empezar a capturar datos:



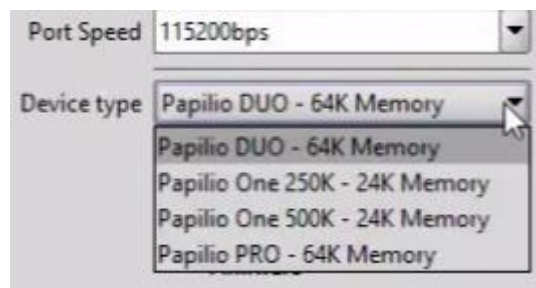
Y en las opciones de captura seleccionaremos el tipo de conexión, que en nuestro caso será **puerto serie**.



Elegimos nuestro puerto a través del cual vamos a capturar:



Dejamos el valor de velocidad del puerto a 115200bps y seleccionamos nuestra placa:

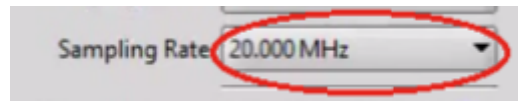


Nos vamos a la pestaña de adquisición de datos y realizamos unas ultimas configuraciones:

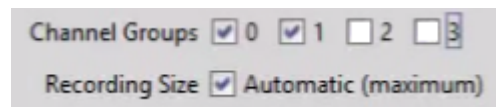


Ponemos el valor de la tasa de muestreo que nosotros prefiramos:

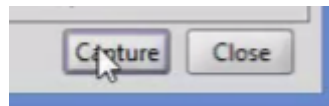
En este ejemplo usamos 20000 Mhz pero con 100 Khz también estaría bien para observar.



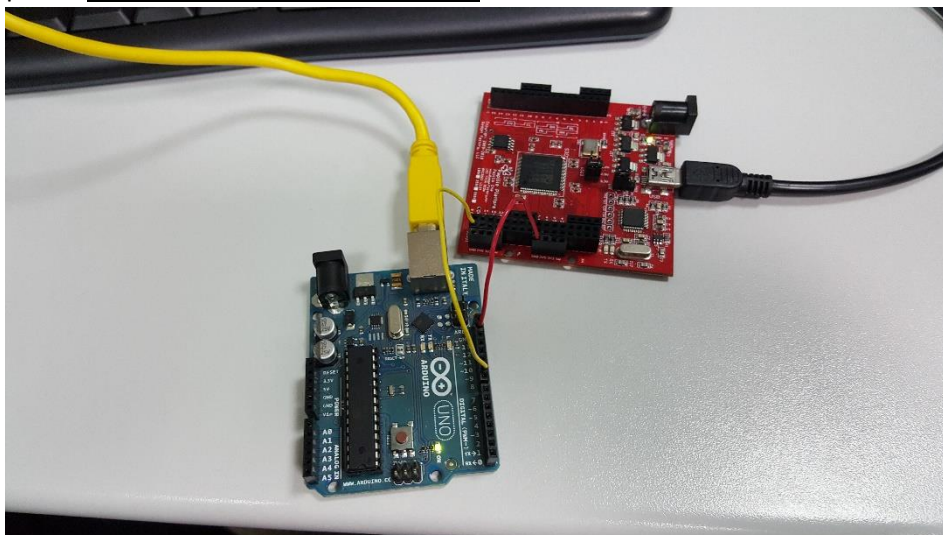
Seleccionamos por ejemplo dos grupos de canales para tener más memoria y dejamos el campo de tamaño de grabación de la captura en automatico.



Y ya podemos empezar la captura:

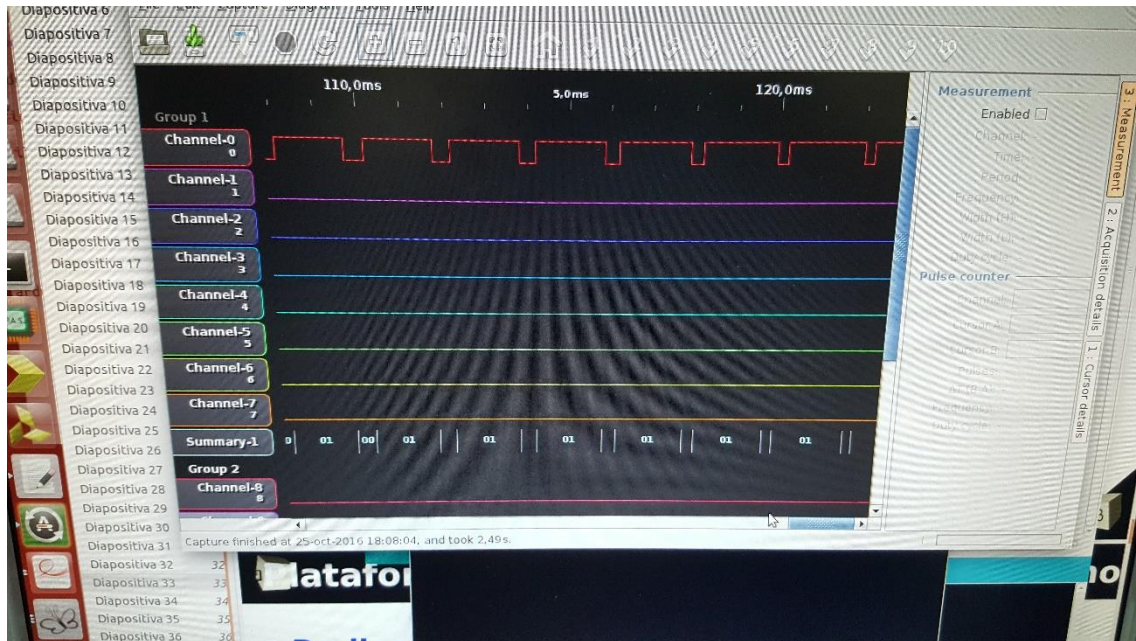


Pues bien, para ello hemos decidido usar la placa arduino en una de sus salidas de PWM para poder observar cómo se capturaría una señal modulada por ancho de pulsos. Hay que tener cuidado a la hora de realizar las capturas ya que para que se realice correctamente ambas placas **deben compartir el mismo GND.**



**NOTA:**  
Simplemente en arduino se ha realizado la carga del sketch de ejemplo fade para observarlo.

Y aquí tendríamos el resultado de la captura realizada por nuestro programa ejecutado en el DesignLab a través de la placa Papilio del ejemplo **Fade** ejecutado en Arduino.



Ejemplo del resultado:

[https://1drv.ms/v/s!Ao2\\_30mhw3bivH-MnTRCJbm783Kt](https://1drv.ms/v/s!Ao2_30mhw3bivH-MnTRCJbm783Kt)

➤ Evaluación después de la actividad

Una práctica muy lucrativa a la cual se le puede sacar bastante a la hora de realizar testado de circuitos, así como el estudio del comportamiento de circuitos.

➤ Incidencias y anécdotas

- Se debe tener paciencia a la hora de realizar el testado de señales lógicas ya que a veces puede dar error, para ello conectar y desconectar la placa Papilio.
- Observar bien que puerto Serial es el que se está ejecutando
- Tener cuidado de que las dos placas tengan el mismo GND