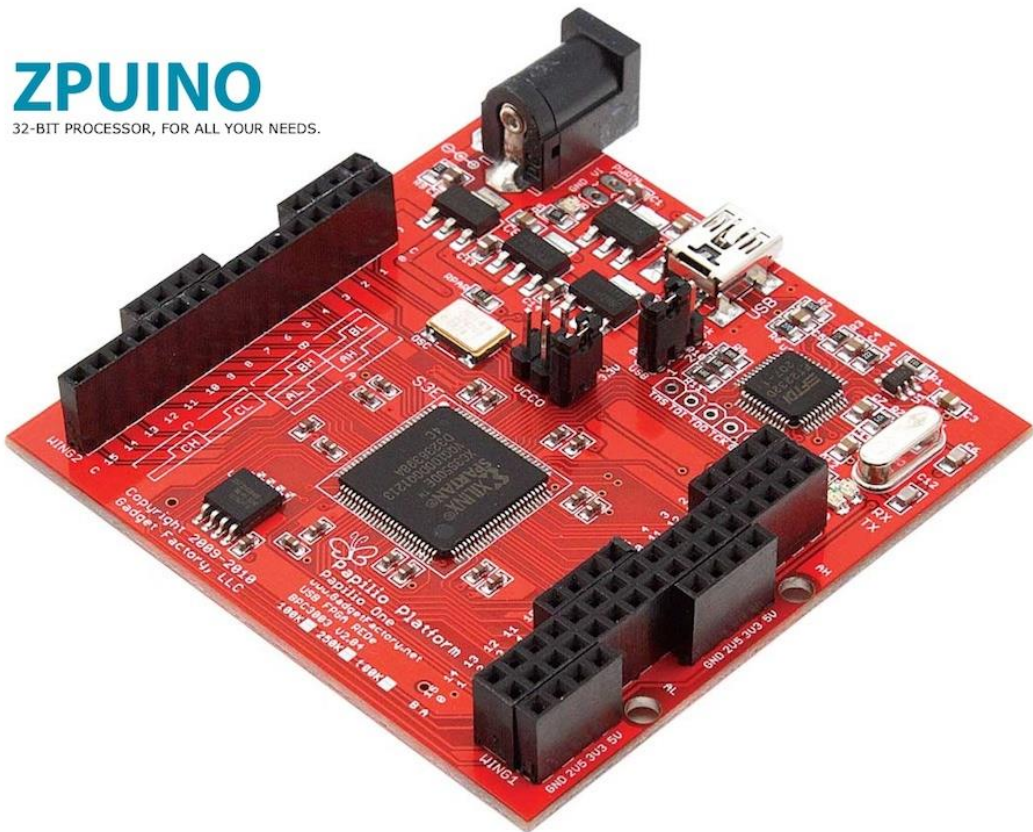


# *Plataforma ZPUino*



**Antonio Manuel Núñez Domínguez**

# Sesión 1 de Laboratorio

---

En este bloque de la asignatura, se va a trabajar con una **FPGA Papilio One 500k**. Esta **FPGA** viene con un **Xilinx XC3S500E** y 4Mbit de memoria **Flash**. Con los Wings que ofrece esta placa, se pueden usar módulos de hardware prefabricados, y módulos diseñados de forma personalizada.

Se utilizará un **SoC**, basado en el microprocesador **ZPY** de **Zylins**. Este **Soc** recibe el nombre de **ZPUino**. El entorno de desarrollo escogido para estas prácticas es **Design Lab**.

## Diseñando circuito para implementación en FPGA

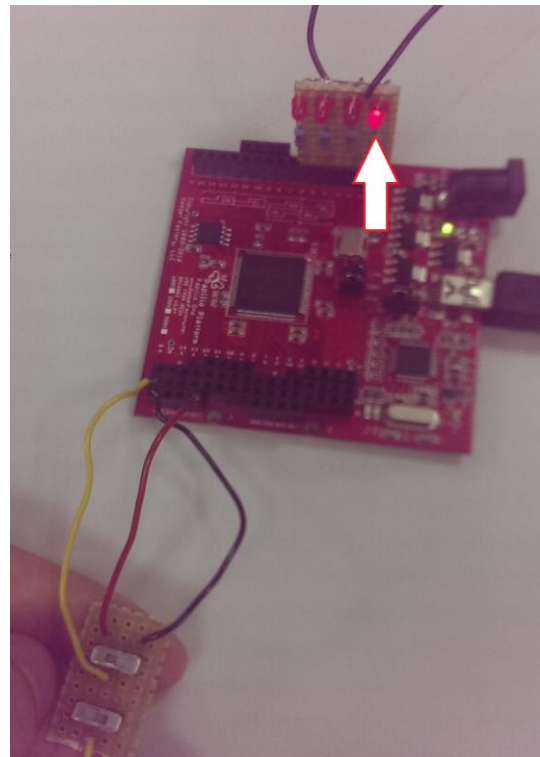
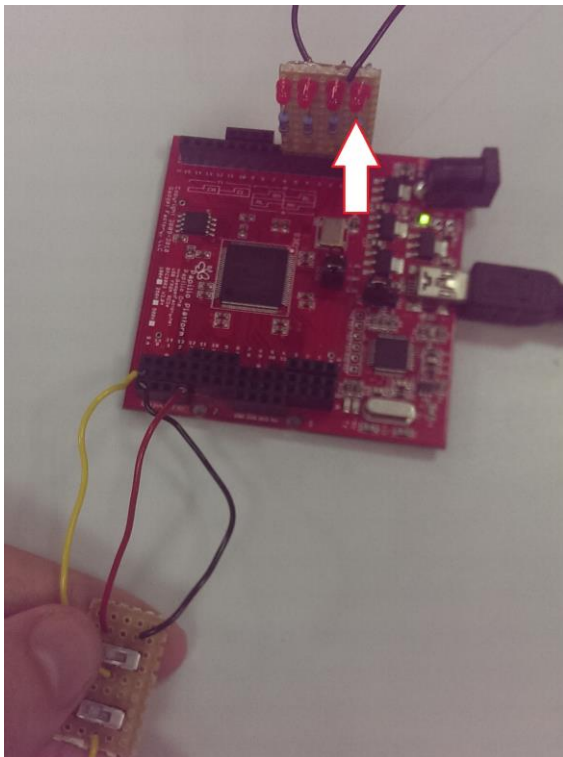
En esta primera parte, utilizaremos **Xilinx** para crear el circuito de un inversor. Asignaremos los conectores a 2 marcadores de **E/S**. Los pines que se utilizarán estarán en la placa **Papilio**.

En la práctica se utilizaron el **WING\_CL0** y **WING\_AL0**.

Se sintetiza el circuito y se generan los archivos de programación.

Una vez hecho esto, habrá que cargar el circuito en la **FPGA**.

Conectamos un LED al **WING\_CL0** y un switch al **WING\_AL0** y se comprueba que funciona como un inversor.



En el siguiente punto, se modificará el código de un sketch, para encender un LED vía serie.  
Lo primero es elaborar el nuevo código para ZPUino y el código en Python:

```
int led = 32;
void setup () {
  pinMode(led, OUTPUT);
  Serial.begin(9600); //Inicializo el puerto serial a 9600 baudios
}

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

```
import serial

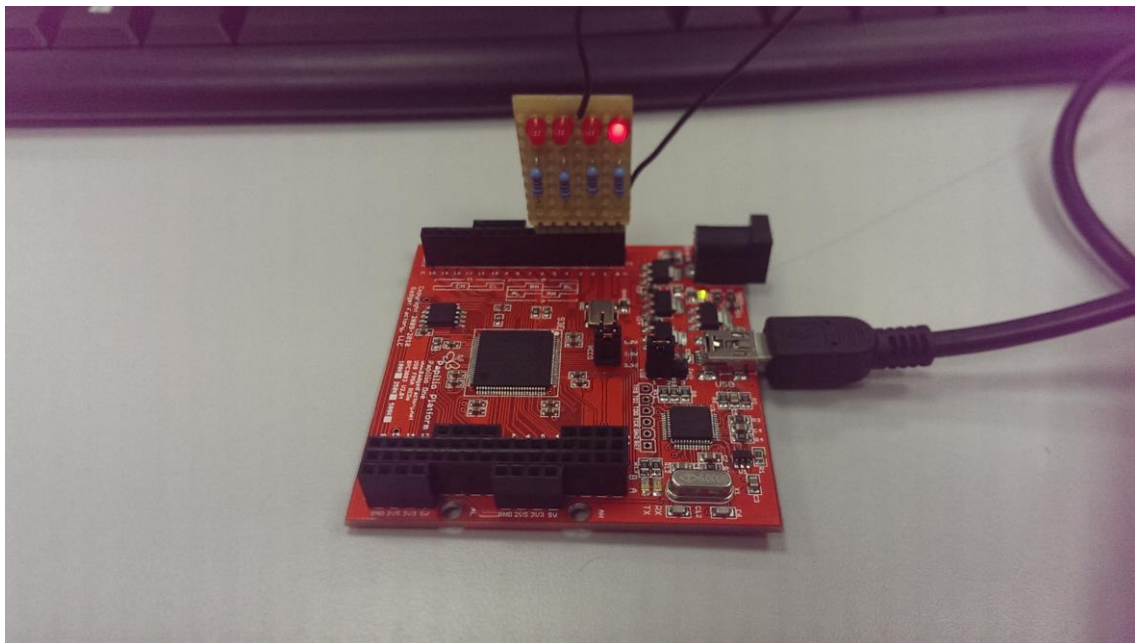
zpuino = serial.Serial('/dev/ttyUSB0', 9600)

print("Starting!")

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

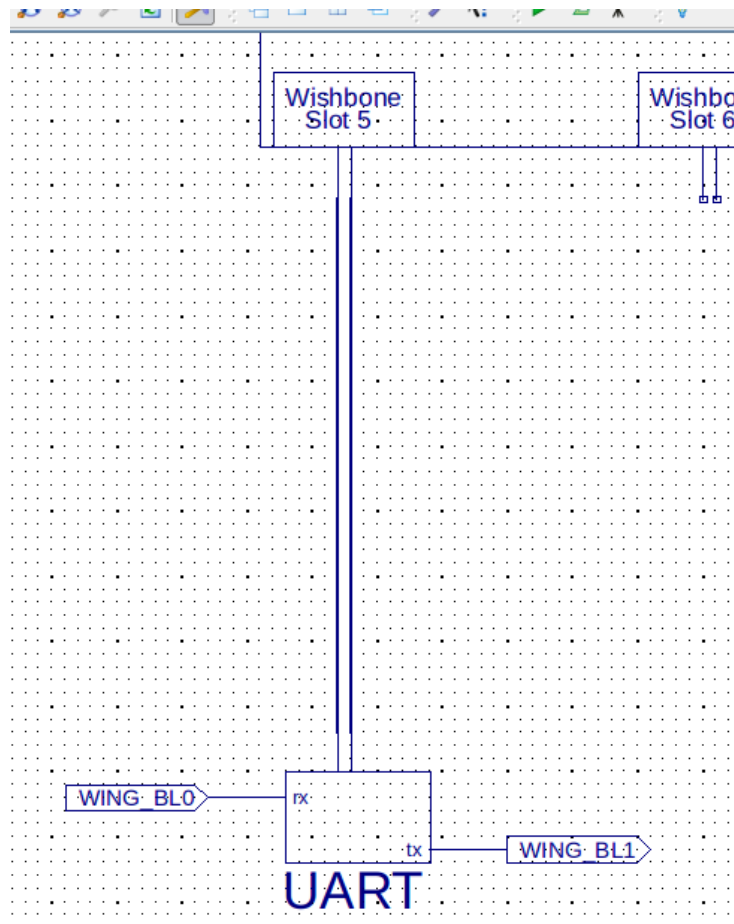
zpuino.close() #Finalizamos la comunicacion
```

Teniendo como resultado:



## Rediseñando el SoC para ZPUino

En esta parte, se utiliza **Xilinx** para diseñar el nuevo Soc. Se hará uso de un slot **Wishbone**, que es un bus que permite que partes de un circuito integrado se comuniquen entre sí. A este bus se conecta un **UART (Universal Asynchronous Receiver-Transmitter)**. Es un dispositivo que controla los puertos y dispositivos vía serie. Pero no se añade físicamente, es decir, se añadirá desde el entorno Xilinx, como se muestra en la siguiente imagen:



A la entrada **rx** asignamos el **WING\_BL0** de la placa Papilio, y a la entrada **tx** asignamos el **WING\_BL1**, habiendo eliminando los wings que hemos utilizado donde se encontraban previamente, para que no haya conflicto.

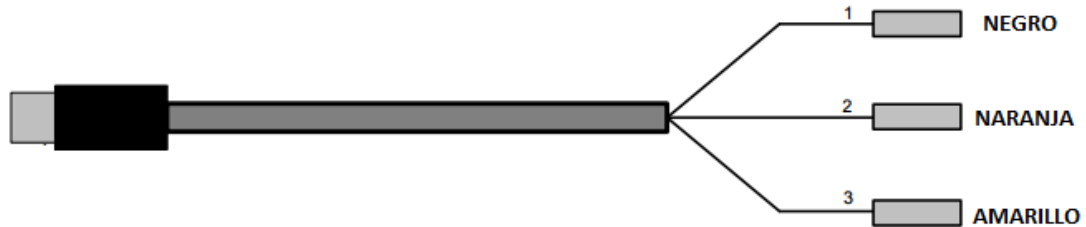
Se sintetiza el circuito y se generan los archivos de programación.

Para conectar el PC con la placa **Papilio** se ha utilizado un cable **TTL-RS232-USB**, así se podrá conectar vía serie. Este cable está formado por 3 cables de diferentes colores.

El **primer cable**, es **negro**, llamado **GND**, que está conectado a tierra.

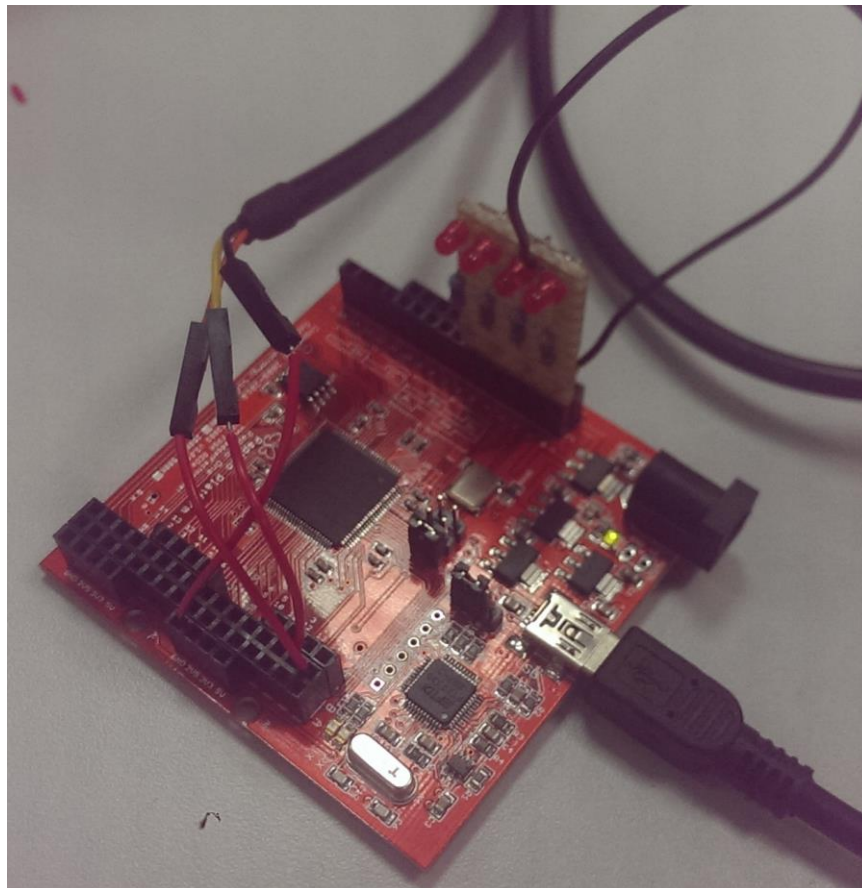
El **segundo cable**, de **color naranja**, llamado **TXD**, transmite los datos de salida.

El **tercer cable**, de **color amarillo**, llamado **RXT**, transmite los datos de entrada.



En la placa, conectaremos **GND** a tierra, el cable **TXD (Naranja)** al **WING\_BL0**, que es el que está conectado a la entrada **rx** del **UART**. El cable **RXD (Amarillo)** al **WING\_BL1**, que está conectado a la entrada **tx** del **UART**.

También conectaremos un LED para comprobar el correcto funcionamiento.



El software en **ZPUino** es el siguiente:

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

void setup() {

  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);
    }
  }

}
```

El código en **Python** es el siguiente:

```
import serial

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

print("Starting!")

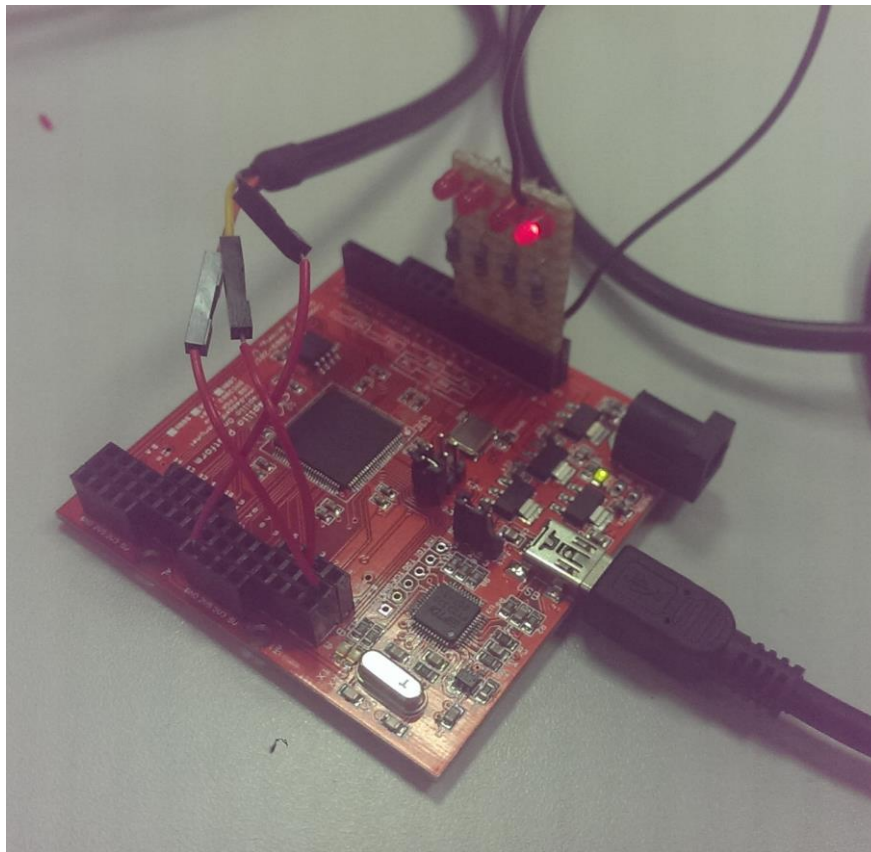
while True:
    comando = raw_input('Introduce un comando: ') #Input
    zpuino.write(comando) #Mandar un comando hacia ZPUino
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

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



Por último, se ejecuta el código **Python** desde la consola y comprobamos el funcionamiento:

```
practicass@mCR-94:~/Escritorio$ python uart.py
Starting!
Introduce un comando: L
LED APAGADO
Introduce un comando: H
LED ENCENDIDO
Introduce un comando: 
```



## Analizador Lógico

En esta última parte, se va a convertir la placa **Papilio** en un **Analizador Lógico**, que nos permitirá capturar los datos de un circuito digital, en este caso, utilizaremos la placa **Arduino**, en la cual cargaremos el ejemplo **Fade**.

```
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

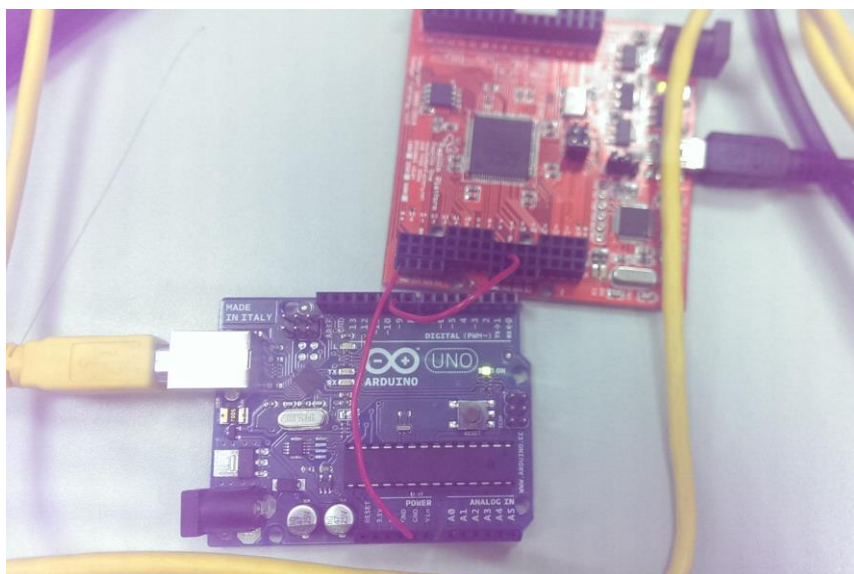
// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Primero se montó el circuito y se conectaron ambas placas como se indica en el código.

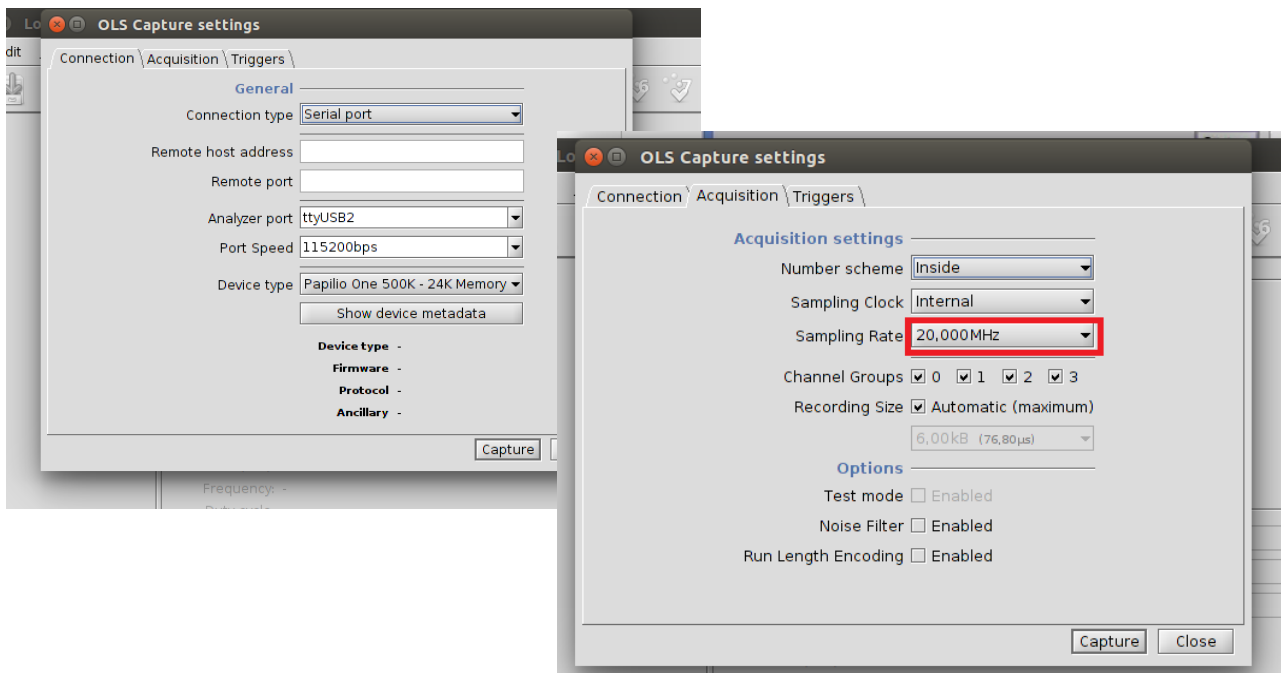




Después se abre el **Analizador Lógico** desde **Design Lab**.



Antes de empezar a capturar, se indica el tipo de conexión, el puerto que está conectado al analizador, en este caso, el que conecta con la placa **Papilio**, y el dispositivo que se va a utilizar como analizador.



Después se indica la frecuencia con la que se va a extraer las muestras. Aunque en la segunda imagen aparezca **20MHz**, después se cambió a **20KHz**, y el resultado se muestra la siguiente imagen, en el **canal 9** del analizador.

