



6 DE ENERO DE 2020

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGUIDOR DE LUZ BASADO EN NIOS ii MANUAL TÉCNICO

DISEÑO DE SISTEMAS DIGITALES

DAYANA GARZÓN Y ALDAIR SOLEDISPA
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

OBJETIVOS

I. General

Adquirir señales de dos fotorresistencias LDR situadas en un panel fotovoltaico y controlar la posición angular del panel fotovoltaico hasta que el valor de las dos LDR sean iguales.

II. Específicos

- Diseñar la arquitectura digital del sistema.
- Establecer el direccionamiento en el sistema operativo Linux.
- Crear archivos ejecutables para realizar lectura de los datos de ángulo de posición, cantidad de luz de dos Ldr y la dirección del panel.
- Configurar el servidor web en el HPS.
- Visualizar en una aplicación web los datos de los Ldrs.

DESCRIPCIÓN DEL PROYECTO

El sistema constará de diseñar e implementar un sistema realiza una comparación entre la lectura de dos fotorresistencias (LDR) y empezará a mover el motor hasta que las lecturas de estas LDR sean iguales, lo cual se determina con el error calculado entre la resta de las lecturas: $LDR1 - LDR2$. El sistema de monitoreo, control y predicción del sistema se la implementará en el Hard-Processor System (HPS) de la FPGA DE-10 Standard, será manejada por puerto serial el cual establecerá comunicación con el arduino. El proyecto se lo desarrollará en tres etapas: La primera etapa constará de montar el sistema operativo Linux con el servidor web en la HPS y poder acceder a ella vía Putty SSH, la imagen del sistema operativo será descargada desde la página de Terasic 1. La segunda etapa será instalar el servidor Apache Http, el cual nos permitirá visualizar mediante una interfaz gráfica las lecturas realizadas. Y la última etapa será de poder, desde una red neuronal, predecir la dirección que tendrá el panel solar en las próximas diez muestras.

DISEÑO DE LA ARQUITECTURA

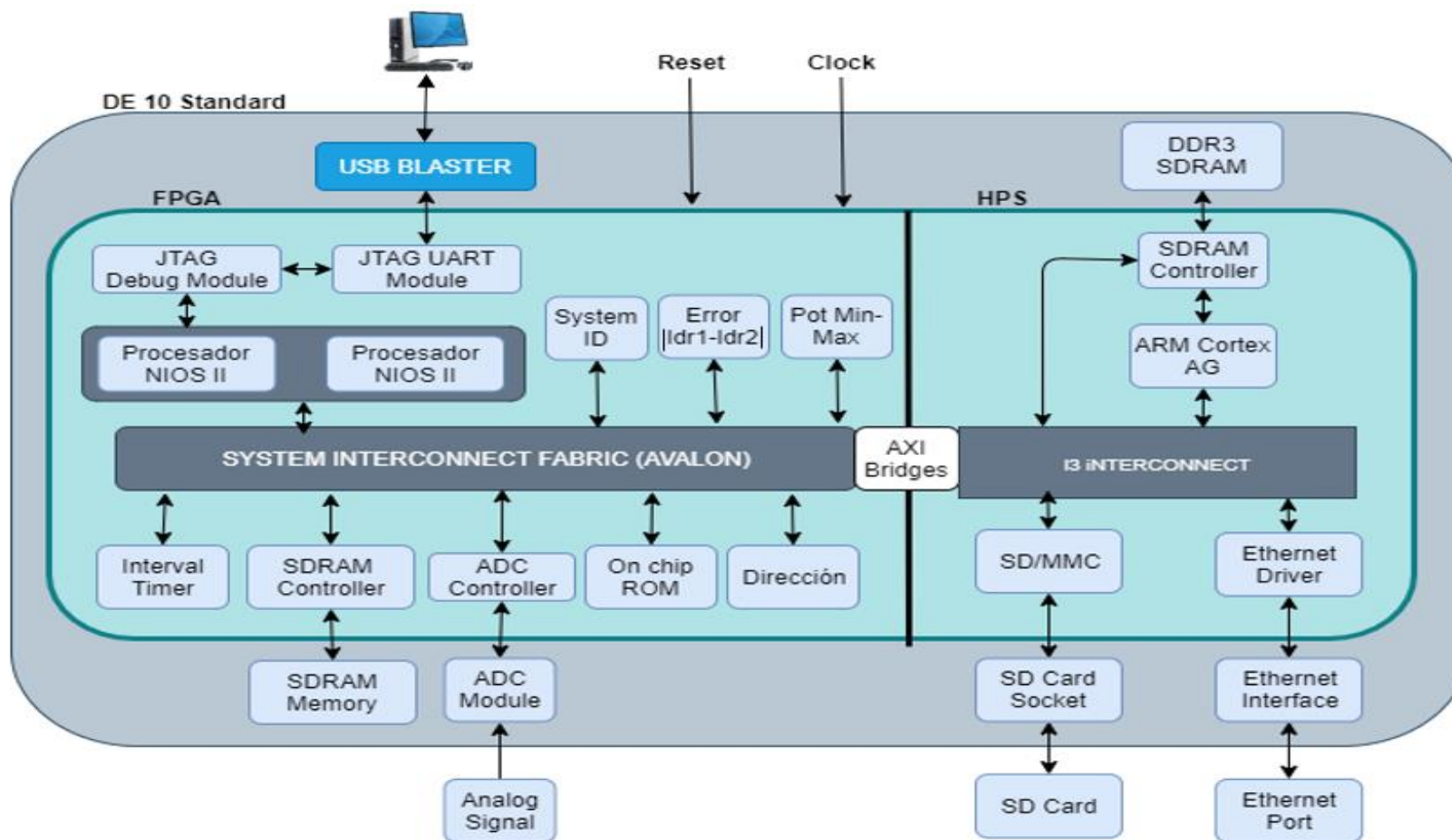


Ilustración 1 - Arquitectura

DESARROLLO

Instalación del Sistema operativo Linux en una MicroSD

1. Descargar la imagen Linux LXDE Desktop de la tarjeta DE10 Standard desde la página de Terasic.

http://www.terasic.com/downloads/cd-rom/de10-standard/Linux/DE10-Standard_LXDE.zip

Linux BSP (Board Support Package): MicroSD Card Image





Title	Version	Size(KB)	Date Added	Download
Linux Console (Kernel 4.5)	1.2 		2018-03-15	
Linux LXDE Desktop (Kernel 4.5)	1.2 		2017-06-06	

Ilustración 2 - Terasic - Linux

2. Descargue e instale el software *Win32 Disk Imager*.
<https://sourceforge.net/projects/win32diskimager/>
3. Conecte la *microSD* a una PC con Windows.
4. Ejecutar *Win32 Disk Imager*. Seleccione el archivo de la imagen para la tarjeta microSD
5. Seleccione el dispositivo donde deseamos escribir.

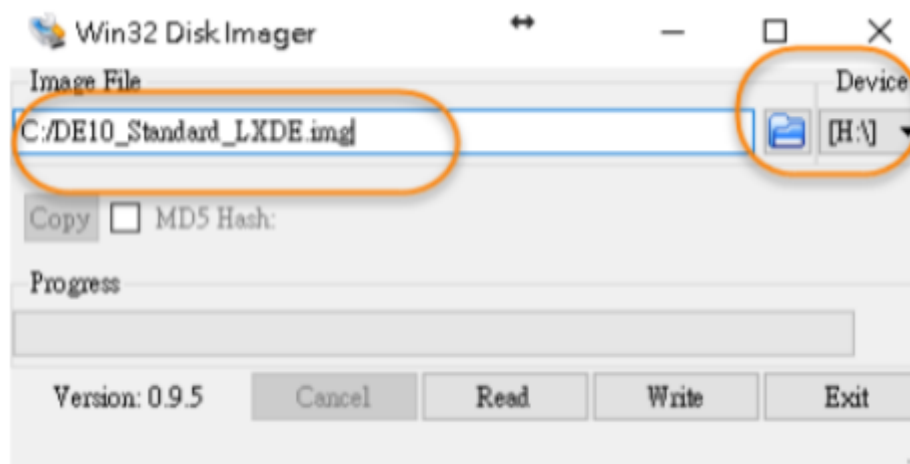


Ilustración 3 - Grabar la imagen en una MICROSD

6. Seleccione “Write” para proceder a escribir el archivo de imagen en la tarjeta. Espere hasta que esté muestre un mensaje de escritura exitosa.

Configuración del Servidor Apache http en Linux

1. Conectamos la MicroSD y el cable Ethernet a la tarjeta de desarrollo DE10 Standard.
2. Después de arrancar el sistema operativo, utilizar el usuario por defecto para iniciar sesión en el mismo. *User: root, sin contraseña.*
3. Para configurar la red y poder tener acceso a Internet. Modificamos el archivo de interfaces de red por medio del siguiente comando:
④ *vi /etc/network/interfaces*
4. Copie el siguiente código en el archivo modificando la IP, máscara de subred y Gateway.

```
#include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto lo
iface lo inet loopback

#auto eth0
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

iface eth0 inet static
address 200.126.14.134
netmask 255.255.255.128
gateway 200.126.14.129

#auto wlan0
#iface wlan0 inet dhcp
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Ilustración 4 - Archivo interfaces

5. Para tener acceso a Internet se configuran las DNS. Utilice el siguiente comando:
④ *vi /etc/resolv.conf*
6. Escribir lo siguiente en el archivo resolv.conf, guarde y cierre el editor de texto *vi* presionando ESC wq!.

```
nameserver 8.8.8.8
nameserver 192.168.1.17
nameserver 192.168.1.19
root@DE10-Standard:/etc#
```

Ilustración 5 - resolv.conf

7. Ingresar las siguientes líneas de comando para crear una ruta por defecto.
④ *ifconfig <interfaz> <dirección IP> netmask <Máscara de subred> route add default gw <Gateway>*
8. Reiniciamos el servicio de networking para hacer efectivos los cambios realizados.
④ *systemctl restart networking.service*
9. Realizamos un ping para probar conectividad a Internet
④ *ping www.google.com*

10. Para configurar el servidor Apache Http primero realizamos la búsqueda del paquete Apache con el siguiente comando

🔗 `sudo apt-cache search apache`

11. Una vez que se obtiene el nombre del paquete Apache, se procede a ingresar el comando para realizar la instalación:

🔗 `sudo apt-get install apache2`

12. Para verificar que se ha instalado correctamente se procede a ingresar al localhost:

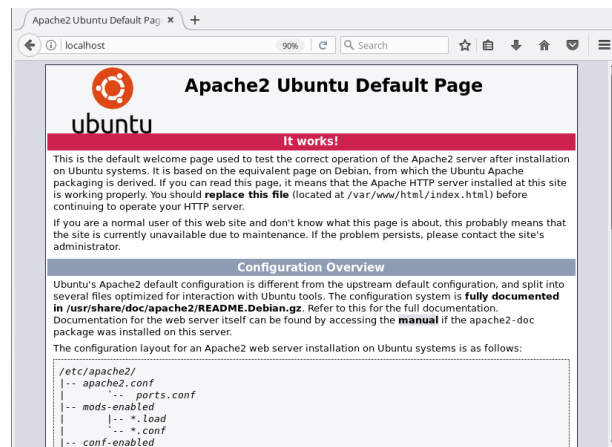


Ilustración 6 - Configurar complemento para validar la contraseña

13. Ahora procedemos a ingresar a la dirección donde se encuentra el archivo html por defecto; en el cual modificaremos para verificar su correcto funcionamiento.

🔗 `cd /usr/local/apache/htdocs`

🔗 `nano index.html`

```
<!DOCTYPE html>

<html>
<body>

<p><a href="https://www.linuxhelp.com">Visit our LINUX tutorial</a></p>

</body>
</html>
```

Ilustración 7 – Archivo html por defecto

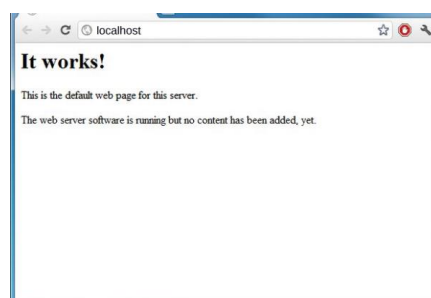


Ilustración 8– Resultado de modificar el archivo html por defecto.

En la página creada se visualizará los datos de las lecturas realizadas posteriormente

Configurar Arquitectura

1. Se procede con la creación de un nuevo proyecto en el programa Quartus.

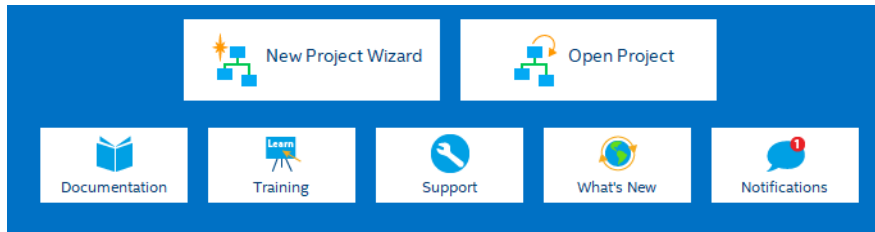


Ilustración 9: Creación de proyecto

2. Se procede a escoger la familia a la que pertenece la DE10 Standard y el dispositivo que permitirá la interacción y se finaliza la creación del proyecto.

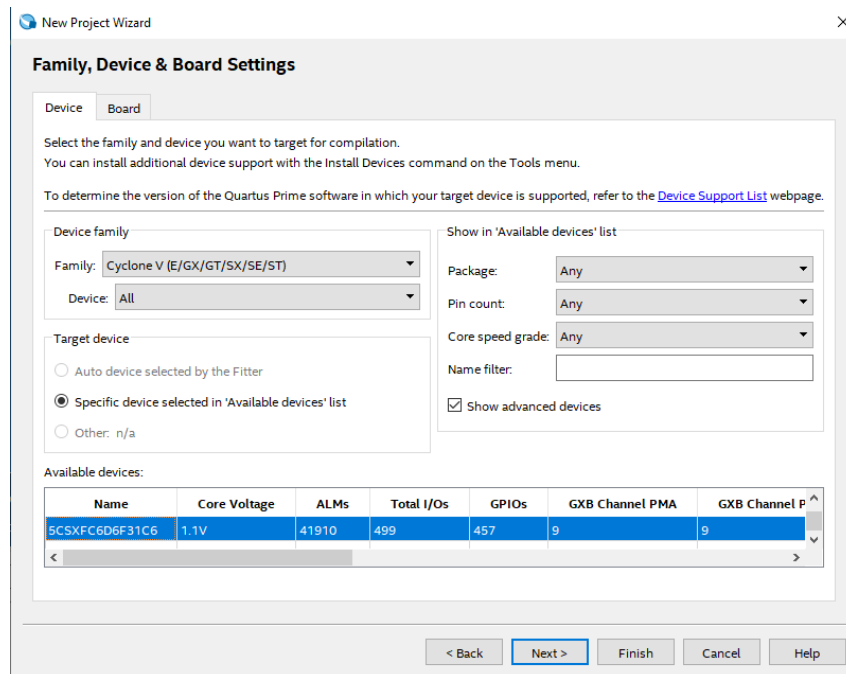


Ilustración 10: Selección de familia y dispositivo

3. Se hace uso de la herramienta Qsys para poder comenzar con la arquitectura del proyecto.

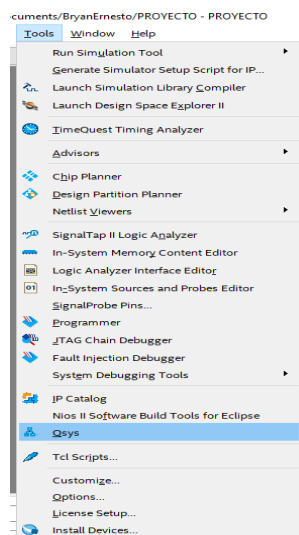


Ilustración 11: Qsys

4. Se proceden a crear cada uno de los bloques necesarios para el funcionamiento del proyecto:
- a. Timer
 - b. Procesador NIOS II
 - c. Controlador ADC
 - d. On-chip Ram
 - e. System ID
 - f. Controlador SDRAM
 - g. PLL

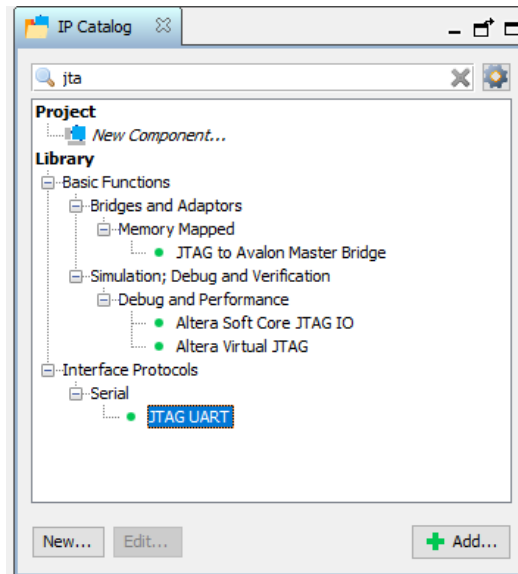


Ilustración 12: Creación del JTAG UART

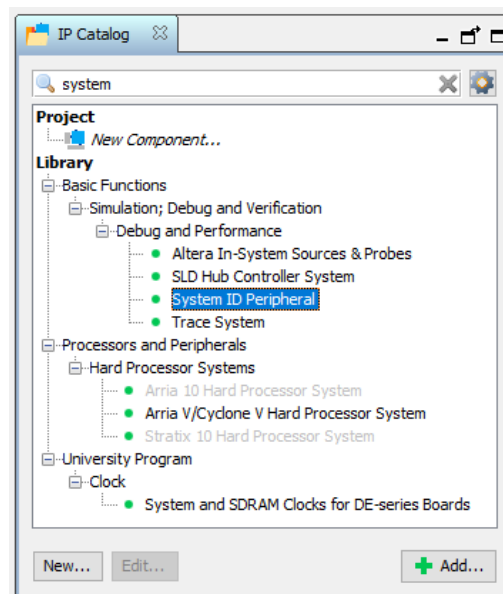


Ilustración 13: Creación de sistema ID

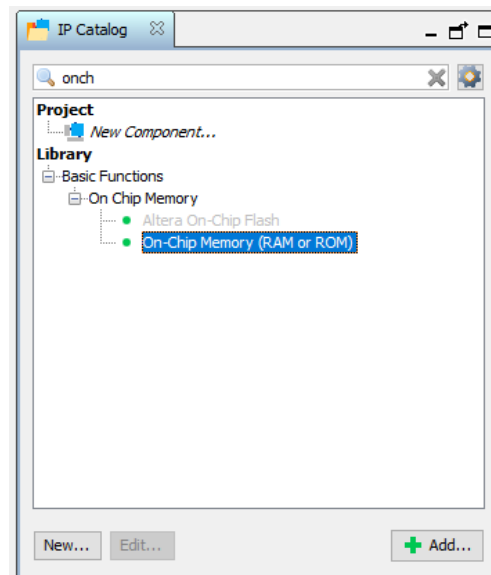


Ilustración 14: Creación de SRAM

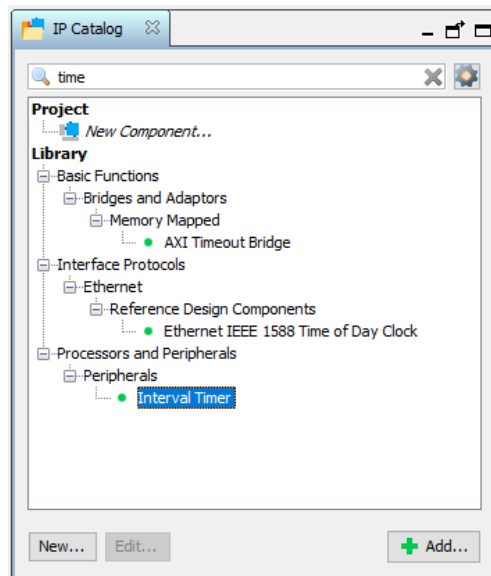


Ilustración 15: Creación de timer

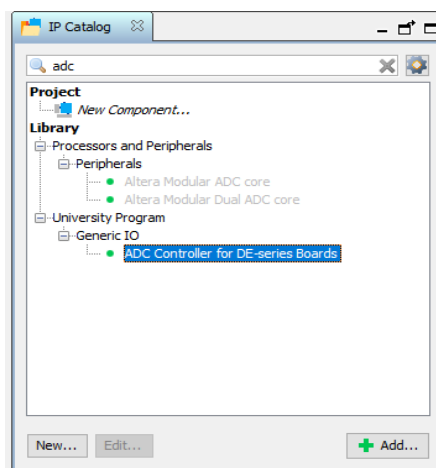


Ilustración 16: Creación del controlador ADC

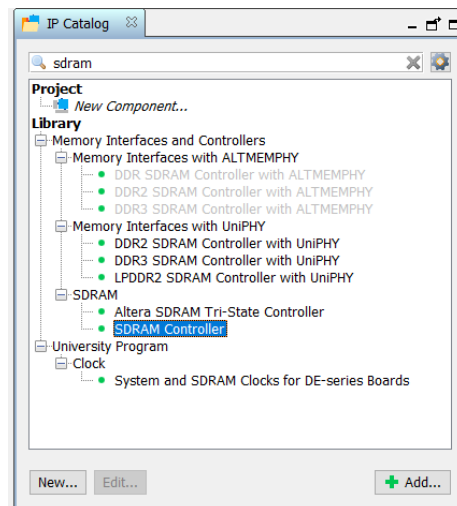


Ilustración 17: Creación del controlado SDRAM

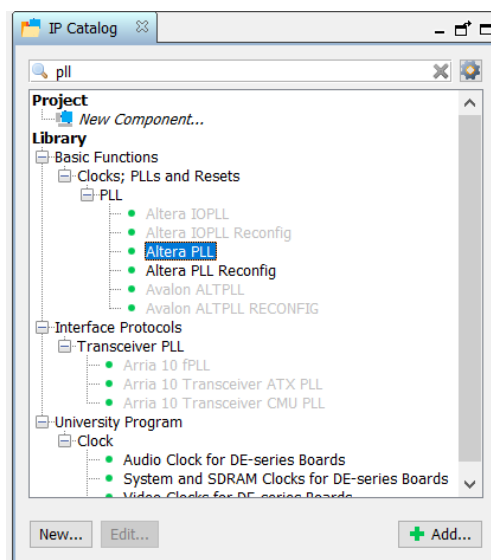


Ilustración 18: Creación de PLL

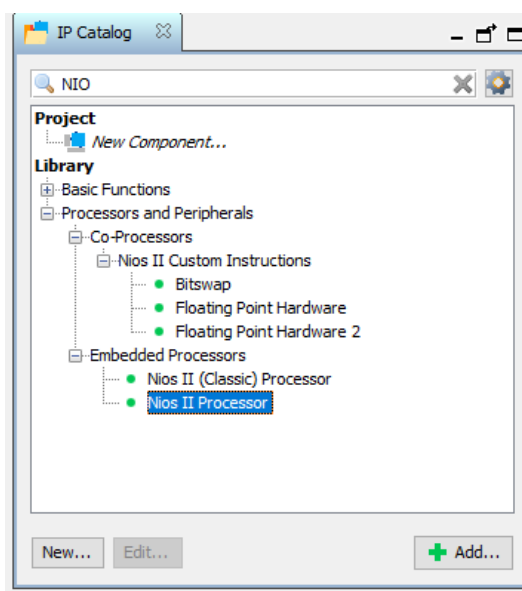


Ilustración 19: Creación de bloque del procesador

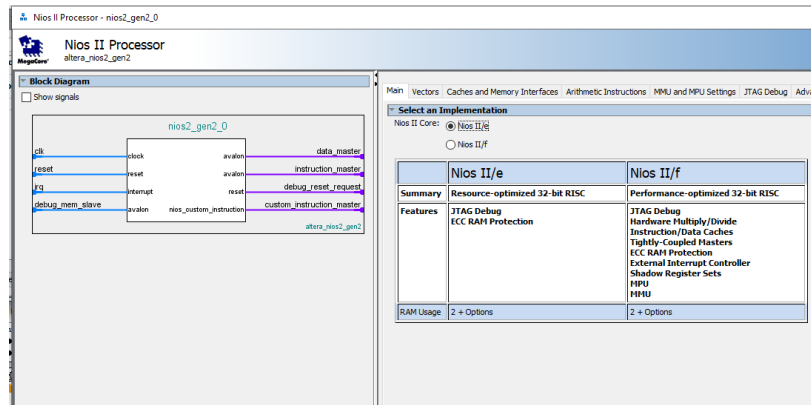


Ilustración 10: Selección de Core

- Es necesario realizar la creación de dos procesadores para que realicen los trabajos de forma independiente.

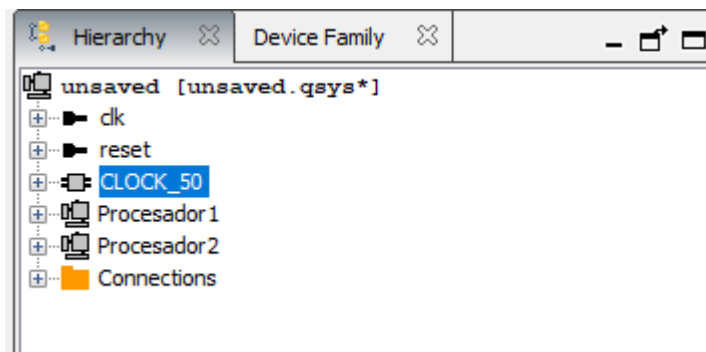


Ilustración 21: Creación de procesadores

- Se realiza a creación de las conexiones de cada uno de los reset en los bloques.

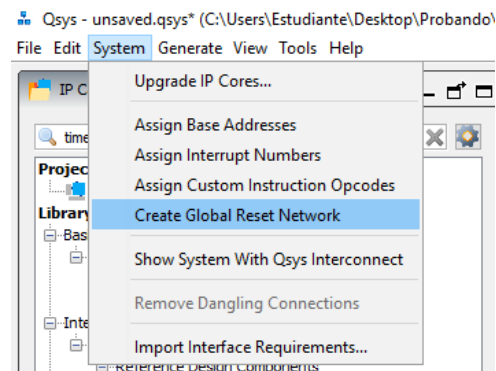


Ilustración 22: Creación de conexiones automáticas

- Se establece por defecto conexiones a través de un sistema automático.

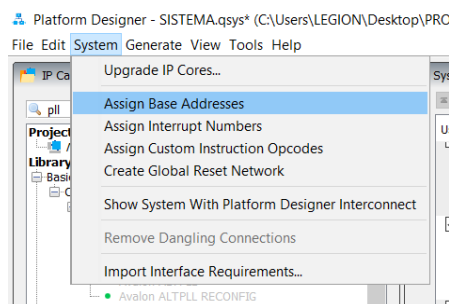


Ilustración 23: Estableciendo conexiones

8. Se procede a realizar las conexiones de los procesadores con cada uno de los bloques

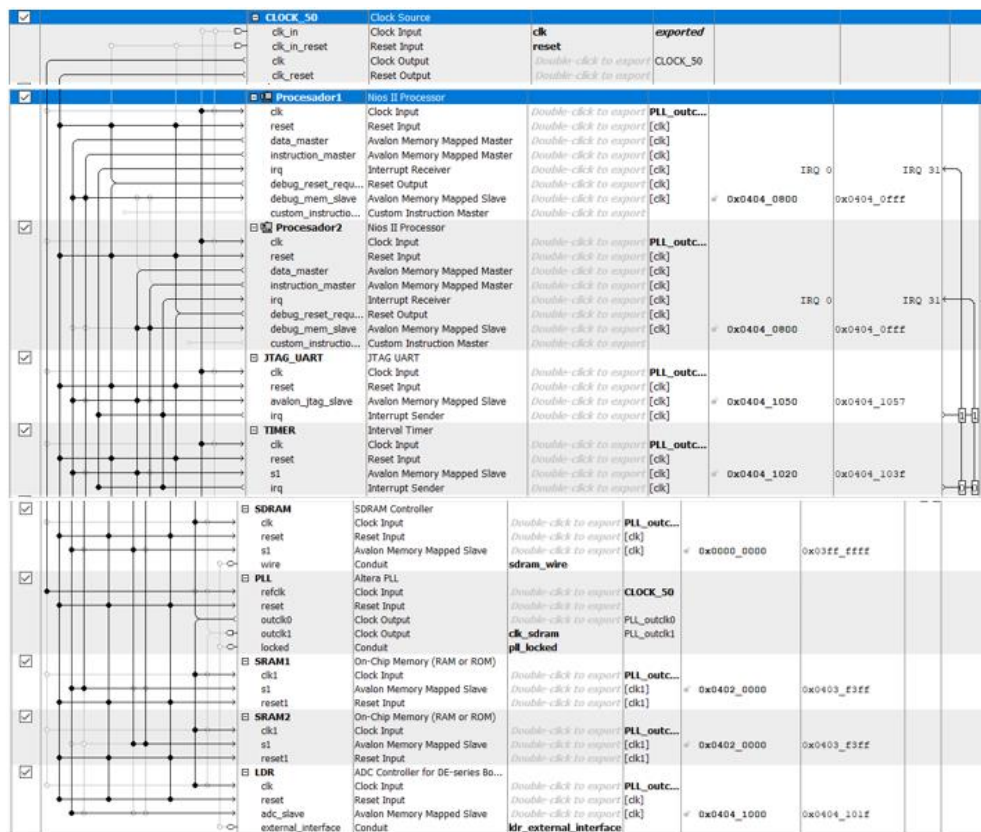


Ilustración 24: Conexiones de los bloques

9. Se genera el archivo HDL para luego la creación del archivo de eclipse.

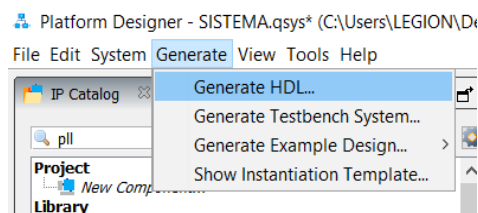


Ilustración 25: Se genera archivo HDL

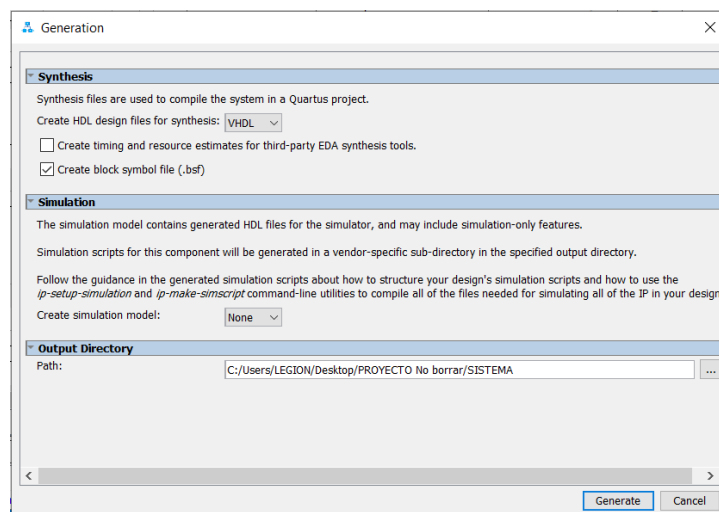


Ilustración 26: Se genera el archivo

10. Se procede a la compilación del sistema creado y para esto es necesario establecer como prioridad al archivo Qsys, para luego conectar la tarjeta embebida.

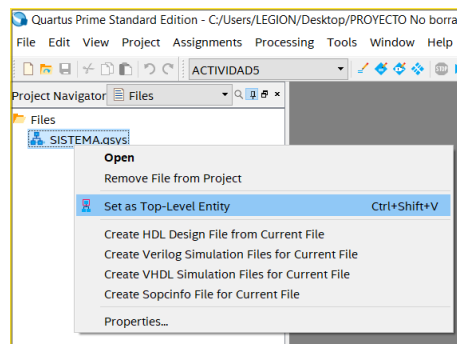


Ilustración 27: Establecer orden de prioridad

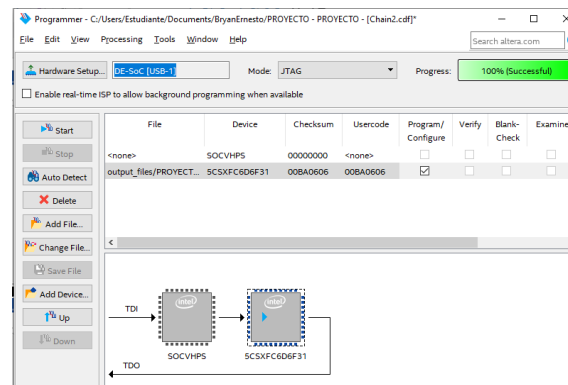


Ilustración 28: Cargar la configuración en la FPGA

11. Por último, se crea un archivo en la herramienta Eclipse para mostrar si la arquitectura del hardware se realizó de forma correcta.

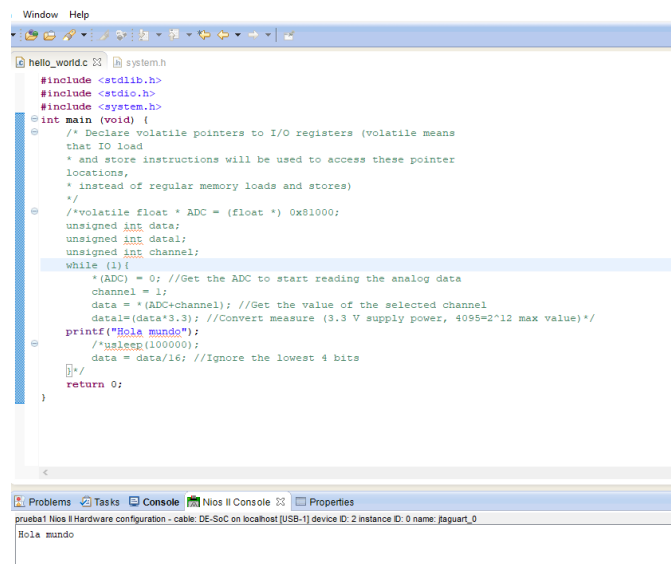


Ilustración 29: Construcción del proyecto

Lectura y predicción de datos

Estructura física

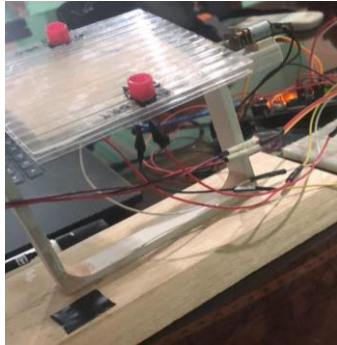


Ilustración 29: Las LDRs se encuentran en los extremos de la parte superior del panel solar

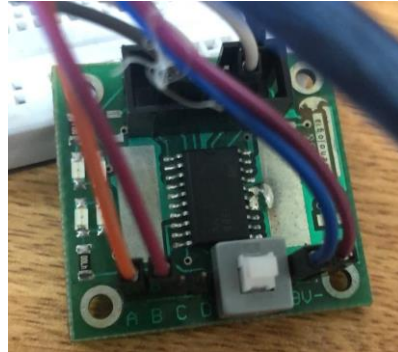


Ilustración 30: Módulo P.H I&T 04 para el control de dirección y velocidad del motor DC.

Se hizo uso de un código en Python el cual se ejecuta en el Hard-Processor System (HPS) de la FPGA DE-10 Standard, para la adquisición de las variables que se van utilizar:

- Posición: adquirido por un potenciómetro.
- LDRO y LDRT: datos adquiridos por resistores que varía su valor de resistencia eléctrica dependiendo de la cantidad de luz que incide sobre él.
- Dirección: adquirido por el módulo P.H I&T 04 como indica la ilustración 30.

Código para adquisición de datos

```
import serial
arduino = serial.Serial('COM3', baudrate=250000)
features = "Pot;LDRO;LDRT;DIR1,DIR2\n"
f=open("datos.txt","a")
while True:
    try:
        line = arduino.readline()
        line = line.decode().replace("\n", " ")
        f.write(line)
        print(line)
    except KeyboardInterrupt:
        f.close()
```

Instalación de seriales

```
(venv) C:\Users\Estudiante\PycharmProjects\yo>pip install serial
Collecting serial
  Downloading https://files.pythonhosted.org/packages/1f/51/6a260c498162c37d0759f3759b7647a10d8d30cabalcfc9aa4b5b1f0d08b/serial-0.0.97-py2.py3-none-any.whl (40kB)
    100% |#####| 40kB 315kB/s
Collecting iso8601>=0.1.12 (from serial)
  Downloading https://files.pythonhosted.org/packages/ef/57/7162609dab394d38bbc7077b7ba0aef10fb9d8b7701ea56faledc0c4345/iso8601-0.1.12-py2.py3-none-any.whl
Collecting future>=0.17.1 (from serial)
  Downloading https://files.pythonhosted.org/packages/45/0b/39b06f4d9b92dc2b6d5b75f900e97884c45bed42ff83203d933cf5951c9/future-0.18.2.tar.gz (829kB)
    100% |#####| 829kB 3.4MB/s
Collecting pyyaml>=3.13 (from serial)
  Downloading https://files.pythonhosted.org/packages/9e/5c/d4865f9b24c7cfe83181e892ec5ade1435cde46bc606bb5ac2b297d75c38/PYYAML-5.3-cp37-cp37m-win_amd64.whl (215kB)
    100% |#####| 225kB 3.4MB/s
Installing collected packages: iso8601, future, pyyaml, serial
Running setup.py install for future ... done
Successfully installed future-0.18.2 iso8601-0.1.12 pyyaml-5.3 serial-0.0.97
```

Código para la instalación de Python (En caso de problemas de instalación) en el Hard-Processor System (HPS) de la FPGA DE-10 Standard,

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt-get update
$ sudo apt-get install python3.6
```

Código de Arduino

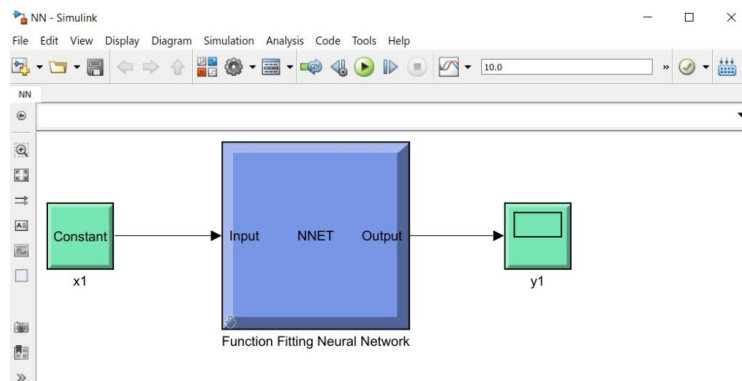
```
#define in1 2
#define in2 3
int LDRO;
int LDRT;
int poserror;
int error;
long valor; //posicion del potenciómetro en tanto por ciento
unsigned long t;
unsigned long previousMillis;
unsigned long startMillis;
unsigned long currentMillis;
const long interval = 2000;

void setup() {
  Serial.begin(250000);
  startMillis = micros();
  for (int i = 6 ; i<11 ; i++)
    pinMode( i, OUTPUT);
}

void loop() {
  analogWrite(10,255);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  LDRO = analogRead(A0);
  LDRT = analogRead(A1);
  valor = analogRead(A2);
  currentMillis = micros();
  t = currentMillis - startMillis;
  error= LDRO-LDRT;
  poserror = abs(error);
  if (poserror > 50){
    if (error > 0) {
      digitalWrite(in2, HIGH);
      //Serial.print("IN2 HIGH");
    }

    else if (error < 0) {
      //Serial.print("IN1 HIGH");
      digitalWrite(in1, HIGH);
    }
  }
  else if (poserror <= 50 ) {
    //Serial.print("AMBOS LOW");
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
  if (currentMillis - previousMillis >= interval) {
    Serial.print(valor);
    Serial.print(",");
    Serial.print(LDRO);
    Serial.print(" ,");
    Serial.print(LDRT);
    Serial.print(" ,");
    Serial.print(digitalRead(in1));
    Serial.print(",");
    Serial.println(digitalRead(in2));
    previousMillis = currentMillis;
  }
  delay(100);
}
```

Red neural para predecir la dirección a la que se encontrará el panel solar después de 10 muestras.



Accuracy

```
main.c
85 float w0[] = {0.42928518317597844215, -0.53838491222422162865, 1.8688984137531338195, -0.7505793763822586956, 0.74023888948374254472};
86 output1.DotProduct1(input1, w0) = (-0.48746229839456354904);
87 return mapminreverse(output1);
88 }
89 /*===== main =====*/
90 int main()
91 {
92     int n;
93     float *GPP1_mapminmax(n1NN);
94     float input1[5];
95     float input2[10];
96     for(n=0; n<5; n++)
97     {
98         input1[n] = GPP1[n];
99     }
100     float *GPP2_layer1(input1);
101     for(n=0; n<10; n++)
102     {
103         input2[n] = GPP2[n];
104     }
105     float output_layer2(input2);
106     printf("%f\n", output+0.35);
107     return 0;
108 }
109
110
111
112
113
114
```

Comments in Spanish:
//-----segun el numero de entradas
//-----Segun el numero de neuronas en el hidden Layer
//-----segun el numero de entradas
//-----Segun el numero de neuronas en el hidden Layer

Input: 0.651312

Program finished with exit code 0

ANEXOS

ENLACE EN GITHUB

En este enlace se encuentra alojado el paper, el presente documento (Manual Técnico); además de los códigos realizados.

<https://github.com/AldairSoledispa/PID-Control-for-angular-position-based-on-FPGA.git>