

MÁSTER UNIVERSITARIO EN

INGENIERÍA DE SISTEMAS EMPOTRADOS



|  |
| --- |
| **Electromiografía: diseño e implementación en un sistema embebido** |

|  |
| --- |
| **Alumno: Xabier Figueroa Sánchez** |
| **Director: Andoni Arruti** |
| **Instructor en la empresa: -** |

**Proyecto Fin de Máster, septiembre de 2017**

Contenido

[1. INTRODUCCIÓN 6](#_Toc488601666)

[1.1. DESCRIPCIÓN GENERAL DEL PROYECTO 6](#_Toc488601667)

[1.2. OBJETIVO DEL PROYECTO 6](#_Toc488601668)

[1.3. ENMARQUE DEL PROYECTO 6](#_Toc488601669)

[2. ESTADO DEL ARTE 7](#_Toc488601670)

[2.1. INGENIERÍA BIOMÉDICA 7](#_Toc488601671)

[2.1.1. Descripción 7](#_Toc488601672)

[2.1.2. Historia 7](#_Toc488601673)

[2.2. BIOSEÑALES 7](#_Toc488601674)

[2.2.1. Descripción 7](#_Toc488601675)

[2.2.2. Naturaleza 7](#_Toc488601676)

[2.2.3. Señales fisiológicas 7](#_Toc488601677)

[2.2.4. Tipos y clasificación 7](#_Toc488601678)

[2.2.5. Digitalización 7](#_Toc488601679)

[2.3. ELECTROMIOGRAFÍA (EMG) 7](#_Toc488601680)

[2.3.1. Descripción 7](#_Toc488601681)

[2.3.2. Uso 7](#_Toc488601682)

[2.3.3. Tipos de sensores 7](#_Toc488601683)

[2.3.4. Señales 7](#_Toc488601684)

[2.4. FPGA 7](#_Toc488601685)

[2.4.1. Arquitectura 7](#_Toc488601686)

[2.4.2. Tecnologías 7](#_Toc488601687)

[2.5. COMUNICACIONES 7](#_Toc488601688)

[2.5.1. Serie (RS232/UART) 7](#_Toc488601689)

[2.5.2. Bluetooth 7](#_Toc488601690)

[3. DESCRIPCIÓN DEL DISPOSITIVO 8](#_Toc488601691)

[3.1. DE0 NANO 8](#_Toc488601692)

[3.1.1. Descripción general 8](#_Toc488601693)

[3.1.2. ADC 8](#_Toc488601694)

[3.1.3. Cyclone 8](#_Toc488601695)

[3.2. MYOWARE 8](#_Toc488601696)

[3.3. HC-05 (MÓDULO BLUETOOTH) 8](#_Toc488601697)

[4. HERRAMIENTAS DE DESARROLLO 9](#_Toc488601698)

[4.1. DESARROLLO FPGA 9](#_Toc488601699)

[4.1.1. ModelSim 9](#_Toc488601700)

[4.1.2. Quartus 9](#_Toc488601701)

[4.2. CAPTURA DE DATOS 9](#_Toc488601702)

[4.2.1. Octave 9](#_Toc488601703)

[4.2.2. Matplotlib (Python) 9](#_Toc488601704)

[4.3. ECLIPSE (ARDUINO) 9](#_Toc488601705)

[5. TRABAJO DESARROLLADO 10](#_Toc488601706)

[5.1. DESCRIPCIÓN GENERAL 10](#_Toc488601707)

[5.2. ESPECIFICACIONES 11](#_Toc488601708)

[5.2.1. Especificaciones ADC128S022 11](#_Toc488601709)

[5.3. DISEÑO 12](#_Toc488601710)

[5.3.1. Diseño sistema 12](#_Toc488601711)

[5.3.2. FPGA: Esquema general 16](#_Toc488601712)

[5.3.3. FPGA: módulo control ADC 17](#_Toc488601713)

[5.3.4. FPGA: módulo principal 18](#_Toc488601714)

[5.3.5. FPGA: módulo transmisión UART 20](#_Toc488601715)

[5.3.6. FPGA: PLL 22](#_Toc488601716)

[5.3.7. PC: Script captura de datos 22](#_Toc488601717)

[5.4. IMPLEMENTACIÓN 22](#_Toc488601718)

[5.4.1. Simulaciones 22](#_Toc488601719)

[5.4.2. Desarrollo 22](#_Toc488601720)

[5.4.3. Resultados 23](#_Toc488601721)

[5.5. PROBLEMAS 23](#_Toc488601722)

[5.6. FUTUROS DESARROLLOS 23](#_Toc488601723)

[6. CONCLUSIONES 24](#_Toc488601724)

[7. REFERENCIAS 25](#_Toc488601725)

[ANEXO A 26](#_Toc488601726)

# INTRODUCCIÓN

## **DESCRIPCIÓN GENERAL DEL PROYECTO**

Hoy en día la biomedicina es una rama muy importante dentro de la propia medicina. En los últimos años se está llevado a cabo una rápida evolución, lo cual conlleva una mejora sustancial en la calidad de vida de las personas.

Gracias a las investigaciones y los desarrollos de sistemas muy sofisticados, es posible detectar problemas de salud con mayor antelación, e incluso detectar problemas que hasta ahora eran difícilmente detectables.

Esto es posible gracias a que los sistemas electrónicos son cada día más potentes, lo que dota a los investigadores y desarrolladores una posibilidad que hasta hace pocos años nadie podía imaginar.

El ser humano produce un sinfín de señales eléctricas que pueden ser capturas para su estudio, o para la detección de anomalías. Estas señales se denominan bioseñales Las bioseñales más estudiadas son:

* ECG (electrocardiograma)
* EEG (electroencefalograma)
* EMG (electromiograma)
* MMG (mecanomiograma)
* EOG (electrooculografía)
* GSR (respuesta galvánica de la piel)
* MEG (magnetoencefalograma)

Una técnica para la detect

COMPLETAR!!

## **OBJETIVO DEL PROYECTO**

## **ENMARQUE DEL PROYECTO**

# ESTADO DEL ARTE

## **INGENIERÍA BIOMÉDICA**

### Descripción

### Historia

## **BIOSEÑALES**

### Descripción

### Naturaleza

### Señales fisiológicas

### Tipos y clasificación

### Digitalización

## **ELECTROMIOGRAFÍA (EMG)**

### Descripción

### Uso

### Tipos de sensores

### Señales

## **FPGA**

### Arquitectura

### Tecnologías

## **COMUNICACIONES**

### Serie (RS232/UART)

### Bluetooth

# DESCRIPCIÓN DEL DISPOSITIVO

## **DE0 NANO**

### Descripción general

### ADC

### Cyclone

## **MYOWARE**

## **HC-05 (MÓDULO BLUETOOTH)**

# HERRAMIENTAS DE DESARROLLO

## **DESARROLLO FPGA**

### ModelSim

### Quartus

## **CAPTURA DE DATOS**

### Octave

### Matplotlib (Python)

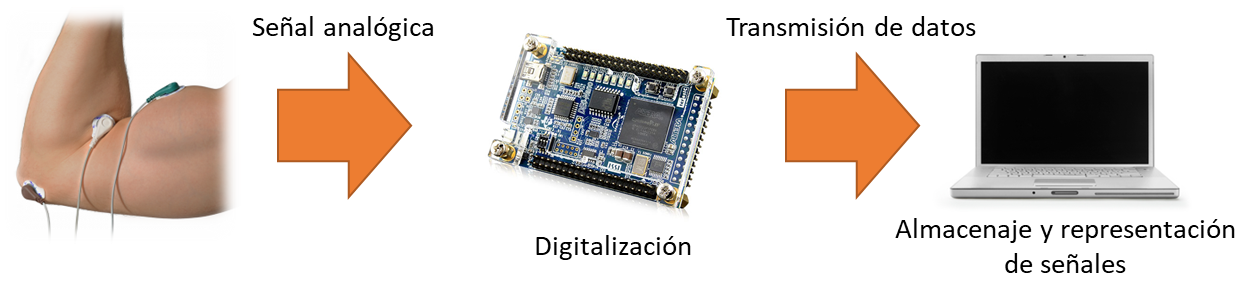
## **ECLIPSE (ARDUINO)**

# TRABAJO DESARROLLADO

## **DESCRIPCIÓN GENERAL**

El objetivo de sistema desarrollado es la captura de las señales eléctricas emitidas por un músculo, y su posterior almacenaje y representación en un ordenador. Para ello, se utilizará la placa de desarrollo *DE0-Nano* de *Terasic*. Esta placa contiene varios módulos embebidos, de los cuales, para este proyecto, se hará uso de:

* Altera Cyclone® IV EP4CE22F17C6N (FPGA)
* ADC128S022 (Conversor A/D de 8 canales)



Las señales analógicas procedentes del músculo son llevadas al ADC. El ADC es controlado por la FPGA, la cual establecerá qué canal y en qué momento (teniendo en cuenta las especificaciones de tiempo del propio ADC) ha de ser muestreado.

La FPGA, aparte de gobernar el ADC, será la encargada de empaquetar los datos recibidos, para su posterior envío al ordenador. Este envío se realizará mediante una comunicación serie (RS232).

La comunicación entre FPGA y ordenador será inalámbrica a través de Bluetooth. Para ello, se dispondrá de un módulo HC-05 y un PC con conexión Bluetooth, o en su defecto, con un módulo Bluetooth en formato USB.

El ordenador es responsable de desempaquetar los datos recibidos y guardarlos en un fichero para su posterior estudio.

Cabe destacar, que este sistema ha de ser flexible, por lo que su diseño ha de ser modular. Al propio ADC podría ser conectada cualquier señal analógica (teniendo en cuenta las limitaciones de la placa), y los datos podrían ser enviados a cualquier sistema con conexión UART.

Por ejemplo, se podrían sustituir los sensores EMG por unos sensores de monitorización cardíaca, y el ordenador, por un dispositivo móvil con sistema Android.

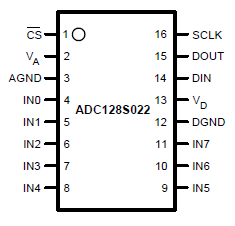
## **ESPECIFICACIONES**

### Especificaciones ADC128S022

El ADC es capaz de leer hasta 8 entradas analógicas, con una resolución de 12bits cada una de ellas. Para esto es necesario seguir las especificaciones dadas por el fabricante en cuestión de tiempos.

A continuación, se explicará cuáles son los parámetros a tener en cuenta para el diseño del controlador de este dispositivo.

En primer lugar, se han de estudiar los pines conectados a las entradas y salidas de la FPGA o a los headers. La siguiente tabla contiene una breve descripción de los pines a tener en cuenta.



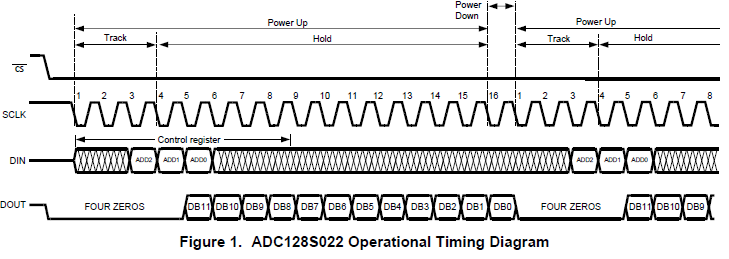
|  |  |  |  |
| --- | --- | --- | --- |
| PIN | | TIPO | DESCRIPCIÓN |
| NOMBRE | Número |
| CS\_n | 1 | Digital I/O | Chip select (negado). La conversión comienza con el flanco descendente de CS\_n. La conversión continua mientras CS\_n se mantenga en nivel bajo. |
| IN0 – IN7 | 4-11 | Analog I/O | Entradas analógicas. Estas señales tienen un rango de 0V a Vref |
| DIN | 14 | Digital I/O | Entrada digital de datos. El Registro de Control es cargado a través de este pin en los flancos de subida del pin SCLK |
| DOUT | 15 | Digital I/O | Salida digital de datos. Los datos muestreados son enviados a través de este pin en los flancos de bajada del pin SCLK. |
| SCLK | 16 | Digital I/O | Entrada digital del reloj. El rango de frecuencias de trabajo para esta entrada es de 0.8 MHz a 3.2 MHz. Este reloj controla directamente los procesos de conversión y lectura. |

En la placa DE-Nano, los pines IN0-IN7 están conectados al header 2X13. Estos pines han de conectarse directamente con el exterior (señales analógicas que se desean muestrear).

Desde el punto de vista de la FPGA, los demás pines son tratados con entradas o salidas:

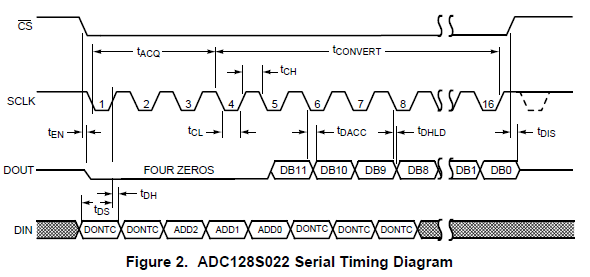
* Entradas: DOUT
* Salidas: CS\_n, DIN, SCLK.

En segundo lugar, se ha de respetar los tiempos especificados por el propio fabricante. A continuación, se muestra el diagrama de tiempos.



Se puede observar que cada 16 pulsos de reloj se repite la secuencia, por lo que se puede considerar al conjunto de 16 pulsos con un ciclo.

* Pin DIN: a través de esta línea se ha de especificar qué canal se quiere muestrear. Los datos del canal especificado en el ciclo actual, serán entregador por el ADC en el siguiente ciclo. Los datos del canal serán leídos durante los flancos de subida de los pulsos 3, 4 y 5 (MSB en primer lugar).
* Pin DOUT: a través de este pin se enviarán los datos muestreados. Los datos enviados en el ciclo actual corresponden a los datos de canal que se envió por la línea DIN en el ciclo anterior. Los datos del canal serán enviados durante los flancos de bajada de los pulsos 5 a 16 (MSB en primer lugar).



## **DISEÑO**

### Diseño sistema

El diseño se ha realizado teniendo en cuenta que la mayor limitación del sistema en su conjunto es el tiempo de procesado de un dato de ADC: desde que se envía el canal que se quiere muestrear hasta que el dato es procesado por el PC.

Para que el sistema sea más rápido, la comunicación entre PC y FPGA será sólo en un sentido. La FPGA enviará los datos muestreados al PC, y éste tan sólo se limitará a procesar dichos datos. En ningún caso permite la comunicación del PC hacia la FPGA, ya que esto ralentizaría la obtención de datos.

Debido a que el PC sólo obtiene datos procedentes de la FPGA, está tendrá que enviar los datos del ADC junto con el canal al que corresponde esos datos. Si esto no fuese así, el PC no sería capaz de relacionar el dato recibido con un canal en concreto.

Teniendo en cuenta lo antes mencionado, el número de bits a enviar son 15 (12 del valor muestreado y 3 del número de canal), lo que hace un total de dos bytes por cada dato muestreado.

El siguiente diagrama muestra el orden de estos bits:

**Byte MSB**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | X | NCanal2 | NCanal1 | NCanal0 | Dato11 | Dato10 | Dato9 | Dato8 |

**Byte LSB**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | Dato7 | Dato6 | Dato5 | Dato4 | Dato3 | Dato2 | Dato1 | Dato0 |

El PC tendrá que desempaquetar estos datos para obtener el número de canal y el dato muestreado antes de almacenarlos.

Se pretende que el sistema sea modular, por lo que en la FPGA se implementa tres módulos con funciones muy definidas:

* ADC Controller: este módulo será el encargado de comunicarse con el ADC. Enviará el número de canal que ha de muestrear y recogerá los datos enviados por el ADC (valores de conversión)
* UART TX: será el encargado de enviar los bytes por línea serie.
* Main Block: el módulo principal. Este módulo será en intermediario entre los dos anteriores. Es el único dotado con cierta inteligencia. Sus funciones son:
  + Enviar al “ADC Controller” en número de canal que se quiere muestrear.
  + Recibir del “ADC Controller” los datos y el número de canal al que corresponden esos datos, para a continuación empaquetarlos para su envío a través de la UART.
  + Enviar a “UART TX” el byte que se quiere enviar en cada momento.

En este sistema, se muestrearán todos los canales. Se comienza obteniendo los datos del canal 0 y se continua de forma ascendente hasta el canal 7. Una vez alcanzado este canal, se vuelve a comenzar del canal 0. Es decir, se muestrean todos los canales por igual de forma circular.

Configuración UART

El siguiente paso consiste en seleccionar la correcta configuración de la UART, así como la velocidad del reloj del ADC.

La siguiente tabla muestra la configuración de la UART:

|  |  |
| --- | --- |
| UART | |
| Bits | 8 |
| Stop | 1 |
| Parity | 0 |
| Muestreo | |
| Bytes por dato | 2 |
| Canales | 8 |

Con los datos arriba presentados, se calcula el periodo de muestreo de los canales. Cada byte enviado consta de 10 bits (Startbit, 8bits de dato, Stopbit), y cada conversión supone el envío de dos bytes al PC.

La fórmula anterior describe el periodo de envío de datos, pero es interesante por calcular cuál es el periodo de muestreo de cada canal. Para ello se modifica la fórmula, sabiendo que se disponen de 8 canales.

La siguiente tabla muestra los períodos calculados para las diferentes velocidades.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Velocidad  (baudios) | Tiempo por bit (s) | Tiempo por byte (s) | Periodo muestreo un canal (s) | Periodo muestreo todo los canales (s) |
| 115200 | 8,68056E-06 | 8,68056E-05 | 0,000173611 | 0,001388889 |
| 76800 | 1,30208E-05 | 0,000130208 | 0,000260417 | 0,002083333 |
| 57600 | 1,73611E-05 | 0,000173611 | 0,000347222 | 0,002777778 |
| 56000 | 1,78571E-05 | 0,000178571 | 0,000357143 | 0,002857143 |
| 38400 | 2,60417E-05 | 0,000260417 | 0,000520833 | 0,004166667 |
| 19200 | 5,20833E-05 | 0,000520833 | 0,001041667 | 0,008333333 |
| 14400 | 6,94444E-05 | 0,000694444 | 0,001388889 | 0,011111111 |
| 9600 | 0,000104167 | 0,001041667 | 0,002083333 | 0,016666667 |

Reloj ADC

Como ya se ha explicado antes, el ADC permite un rango de trabajo de reloj entre 0.8 MHz y 3.0 MHz.

Teniendo en cuenta que el reloj integrado en la placa DE0-Nano es de 50MHz, se puede obtener una señal de reloj dentro del rango de trabajo del ADC de manera muy sencilla a través de un PLL: Basta con dividir la frecuencia entre 25 para obtener una señal de 2 MHz.

Esta velocidad no es limitante para el sistema, ya que la velocidad de la UART será más lenta incluso trabajando a 115200 (ver tabla X.X). A continuación, se realizan los cálculos que lo demuestran.

El ADC trabaja en ciclos de 16 bits (cada bit es una señal de reloj). Cada ciclo de ADC tiene un periodo:

\* 86,8 : periodo de envío de un byte por UART a 115200 baudios

La fórmula anterior muestra el tiempo que necesita el ADC para un ciclo entero. Hay que tener en cuenta, que el canal que se quiere muestrear se le envía en el ciclo anterior al ciclo en el que se obtienen los datos, por lo que desde que se selecciona el canal y se obtienen los datos son necesarios dos ciclos.

\* 86,8 : periodo de envío de un byte por UART a 115200 baudios

Debido a que el ADC es más rápido la UART, se muestrearán varios datos de un mismo canal de manera continua. Hasta que un dato completo no es transmitido a través de la UART, no son leídos los datos del siguiente canal del ADC. A efectos prácticos no tiene ninguna importancia, ya que se procesará siempre un dato de cada canal.

Procesado de datos en PC (servidor)

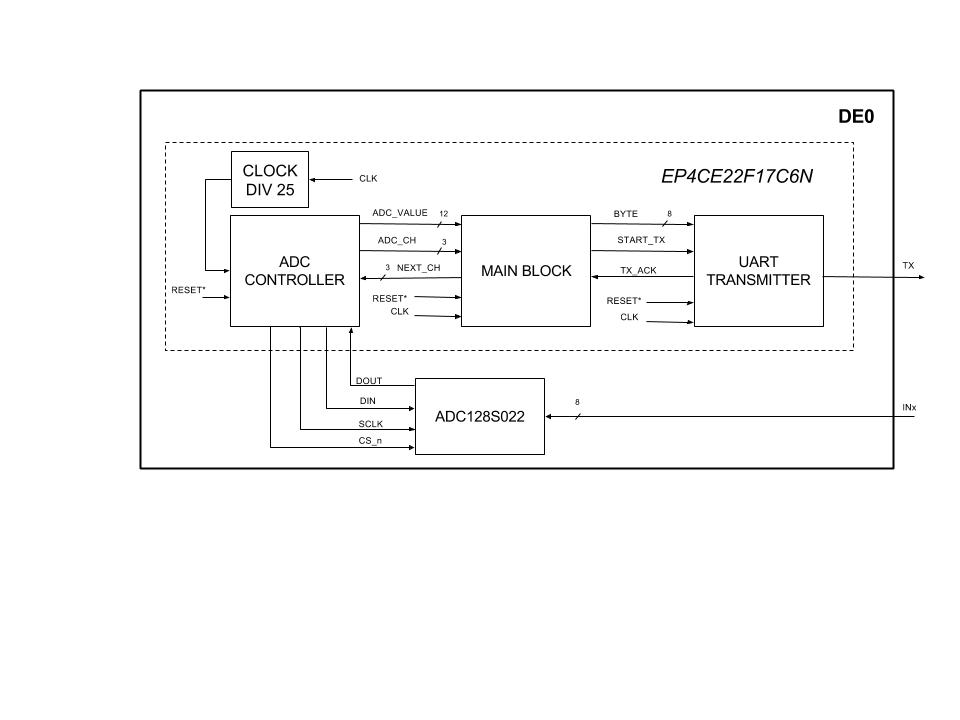
Se han considerado diferentes formatos en los cuales almacenar los datos. Tras un filtro previo, la decisión final quedó entre formato “excel” o “csv”. El problema que tiene “Excel” frente a “csv” es que su manejo a través de aplicaciones de escritorio o scripts (Matlab, Octave, Python…) es más completo, mientras que casi todos los lenguajes de procesado tienen funciones para manejo de ficheros “csv”.

El formato en el que se almacenan los datos recibidos será en un fichero “CSV” para cada canal. Esto supone que por cada sesión de trabajo se obtendrán 8 ficheros, con el nombre “data\_channel\_X.csv” (X corresponde al número de canal).

Los datos son almacenados en dos columnas. La primera corresponde al tiempo transcurrido desde el inicio de la captura de datos y el procesado del dato a almacenar. La segunda por su parte será el valor del dato recibido.

### FPGA: Esquema general

El siguiente esquema muestra el diseño completo de la FPGA



Se puede observar que el sistema completo trabajará con dos frecuencias diferentes:

* 50 MHz: Main Block, UART TX.
* 2 MHz (50MHz/25): ADC Controller.

Comunicación ADC Controller – ADC128S022

“ADC Controller” suminista su mismo reloj al ADC a través de “SCLK”. Este módulo también controla el “chip select” del ADC.

A través de la línea *DIN,* “ADC Controller” envía el siguiente canal a muestrear, y a través de *DOUT* se leerá el valor muestreado.

Comunicación Main Block - ADC Controller

El bloque principal envía al control de ADC en siguiente canal que se quiere muestrear.

El control del ADC a su vez, enviará al bloque principal el valor del ADC y el número de canal al que corresponde dichos datos. Estos datos se envían de manera independiente, ya que “Main Block” es el encargado de empaquetar estos datos de la manera que corresponda.

Comunicación Main Block – UART TX

El módulo principal envía el byte que el módulo UART debe comunicar al PC. Sólo en byte es enviado al módulo UART, ya que es éste es el encargado de empaquetar dicho byte con la configuración que corresponda (paridad, stopbit…) y transmitirlo a la velocidad que corresponda.

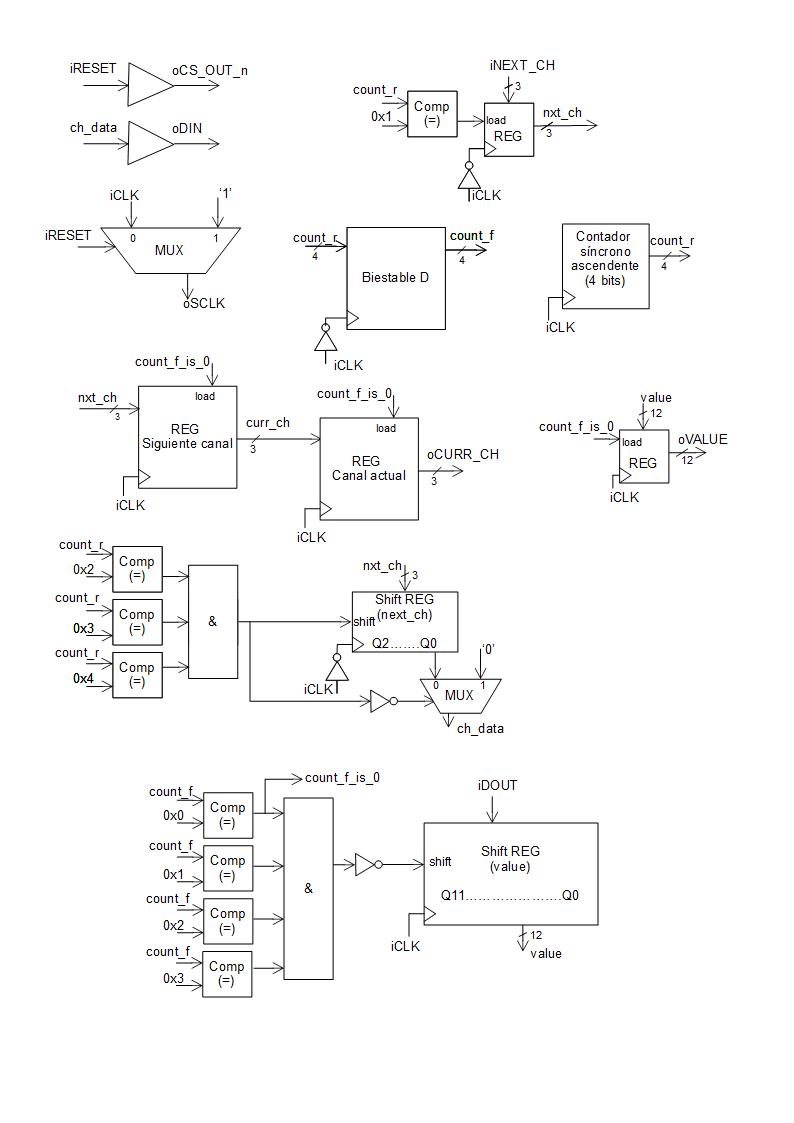
Cuando la UART haya enviado un dato, lo hará saber al módulo principal a través de *TX\_ACK*, de esta manera, “Main Block” sabe que puede enviar el siguiente byte.

La línea *START\_TX* hará saber a la UART que puede empezar a transmitir, ya que el módulo principal sabe cuándo son válidos los datos en *BYTE*. Si no existiera esta línea, la UART podría comenzar a transmitir datos no válidos, y el PC procesaría datos que no proceden del ADC.

### FPGA: módulo control ADC

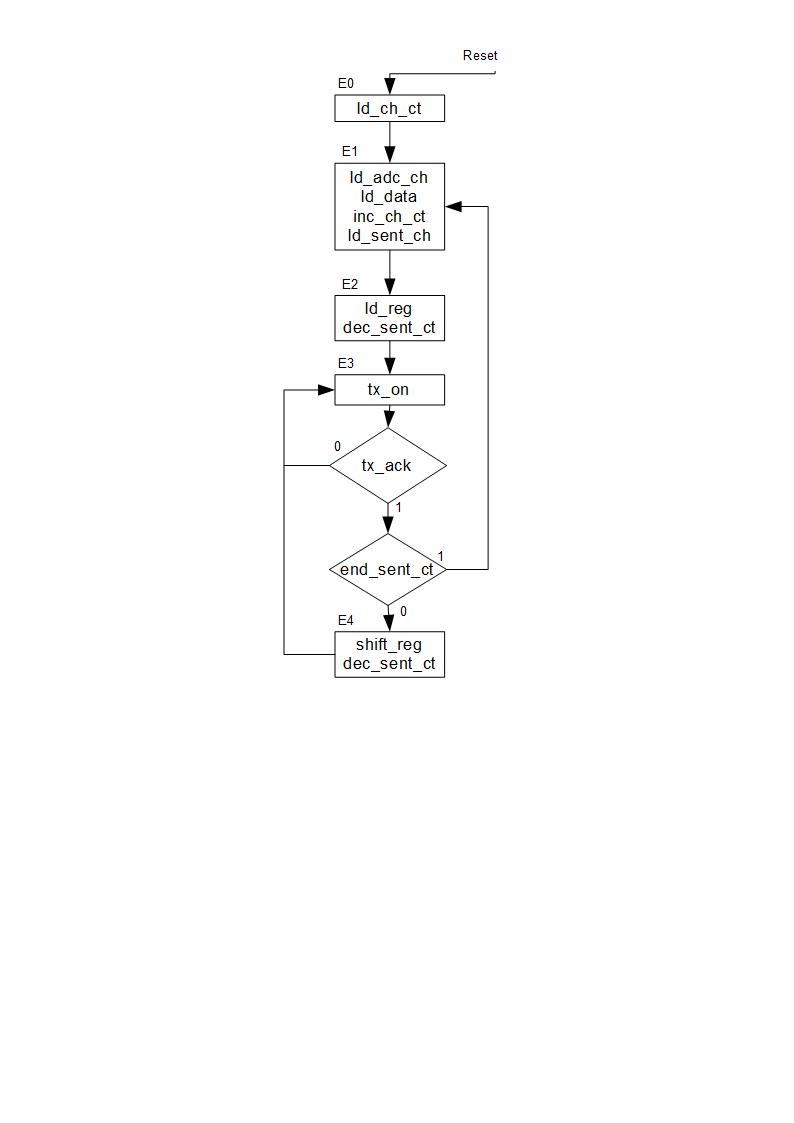
El módulo ADC no tiene unidad de control, ya que no es necesaria. Se ha implementado tan solo utilizando unidad de proceso.

Unidad de proceso

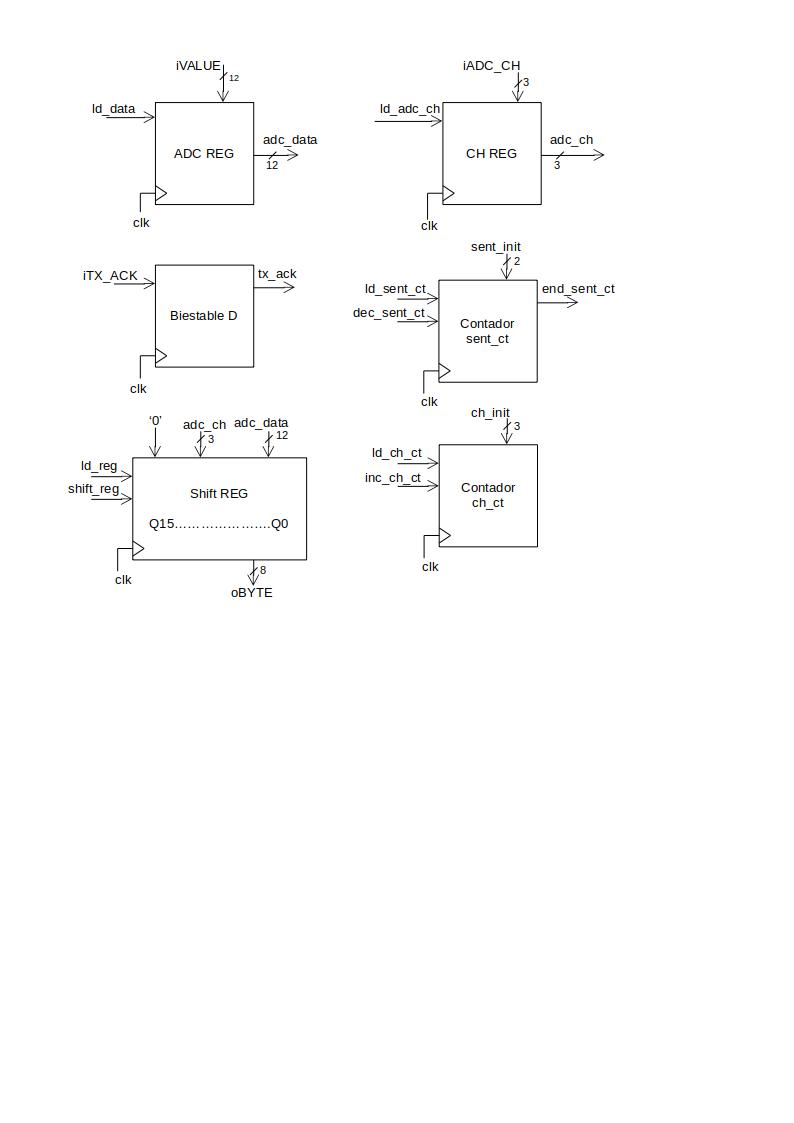


### FPGA: módulo principal

Unidad de control

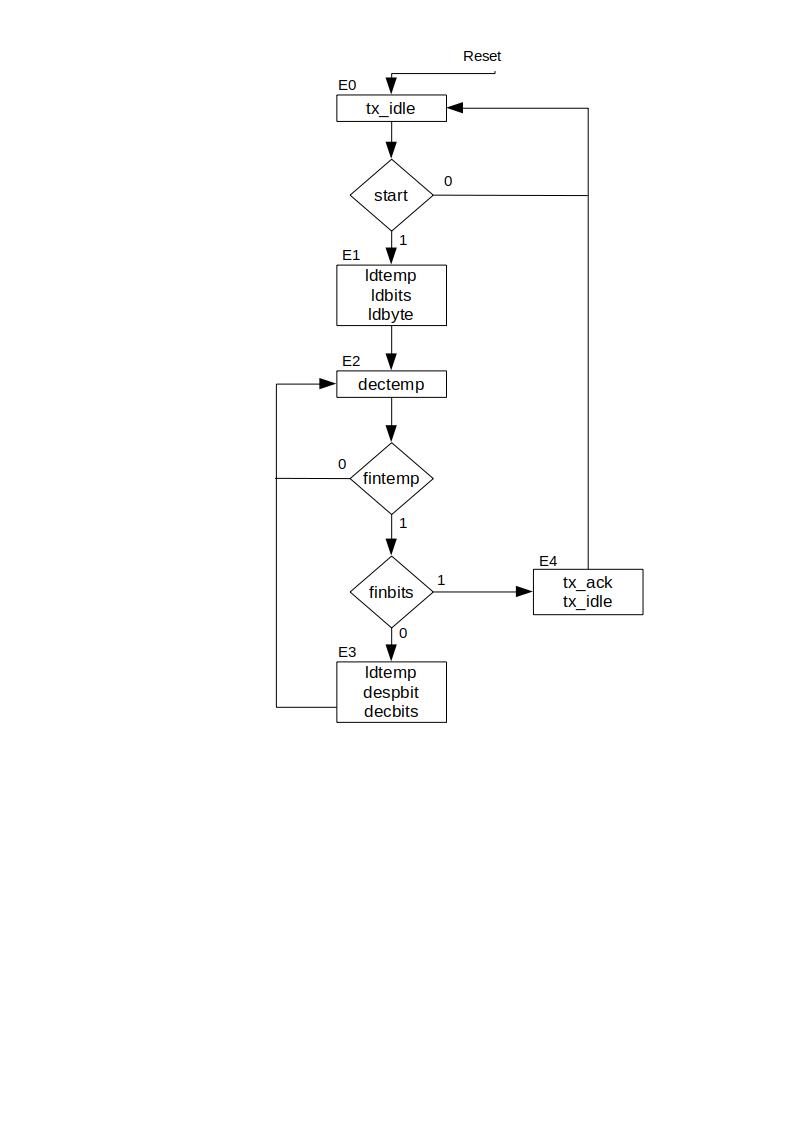


Unidad de proceso

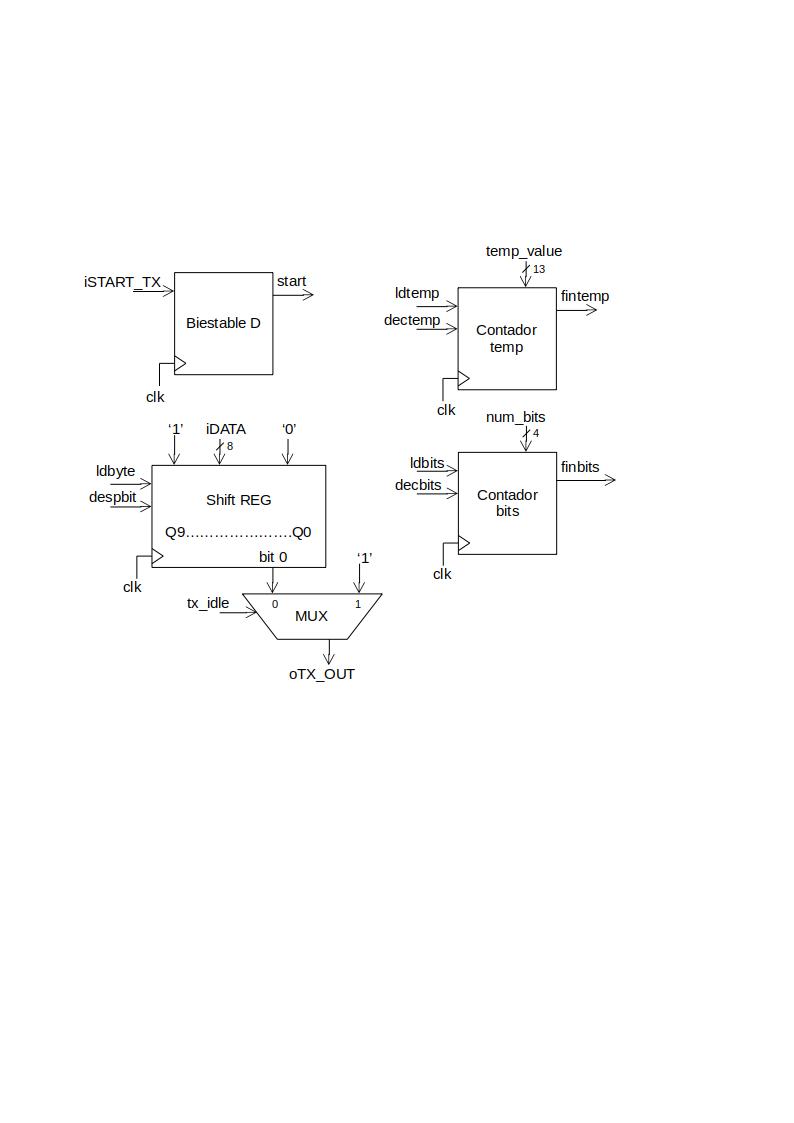


### FPGA: módulo transmisión UART

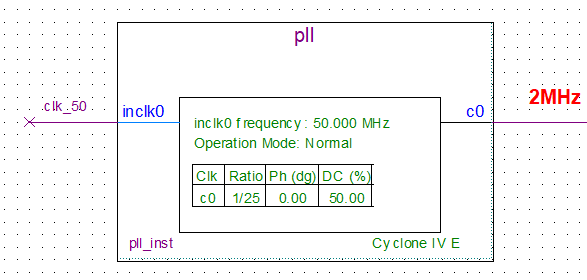
Unidad de control



Unidad de control



### FPGA: PLL



### PC: Script captura de datos

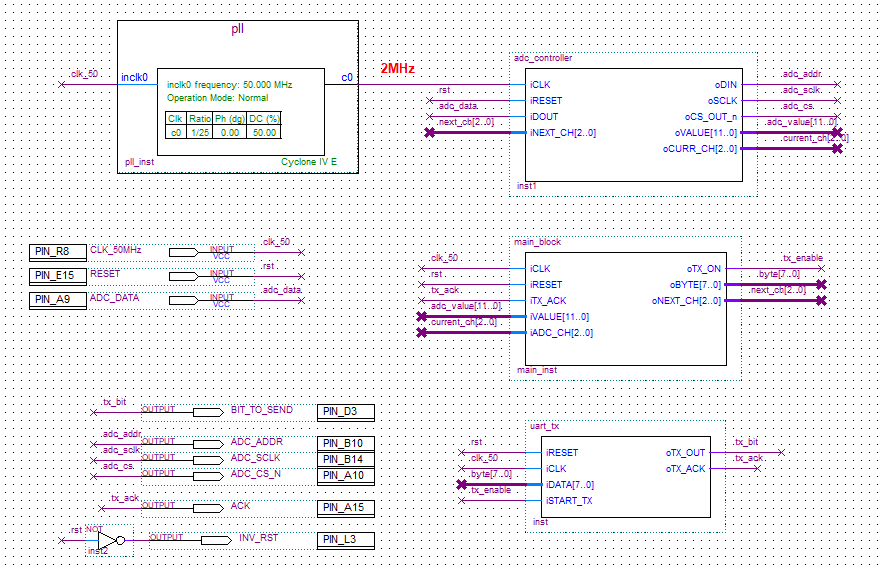
Diagrama de flujo

## **IMPLEMENTACIÓN**

### Simulaciones

### Desarrollo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entradas | |  | Salidas | |
| Nombre | Pin |  | Nombre | Pin |
| ADC\_DATA | PIN\_A9 |  | ACK | PIN\_A15 |
| CLK\_50MHz | PIN\_R8 |  | ADC\_ADDR | PIN\_B10 |
| RESET | PIN\_E15 |  | ADC\_CS\_N | PIN\_A10 |
|  |  |  | ADC\_SCLK | PIN\_B14 |
|  |  |  | BIT\_TO\_SEND | PIN\_D3 |
|  |  |  | INV\_RST | PIN\_L3 |



### Resultados

## **PROBLEMAS**

## **FUTUROS DESARROLLOS**

# CONCLUSIONES

# REFERENCIAS

# ANEXO A