UNIVERSIDADE DA CORUÑA

ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01045**

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

AUTOR: **NATALIA PÉREZ GARCÍA**

TUTOR: **Mª CARMEN MEIZOSO LÓPEZ**
ESTEBAN JOVE PÉREZ

FECHA: **SEPTIEMBRE DE 2019**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

ÍNDICE GENERAL

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice de figuras

Índice de tablas

Listado de códigos de programación

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice del documento MEMORIA

1 Objeto

El objeto de este proyecto consiste en utilizar la plataforma de desarrollo de circuitos digitales Nexys4 DDR basada en la FPGA de la familia Artix-7, para la adquisición de datos de sensores que utilicen la interfaz de comunicación I2C, procesarlos y mostrarlos en un display. Se completará el diseño añadiendo una UART para enviar los datos recogidos a un PC.

2 Alcance

Se pretende obtener un sistema totalmente funcional que se desarrolle según las etapas siguientes:

1. Análisis y selección del sensor o sensores a emplear.
2. Desarrollo en VHDL de la interfaz de comunicación I2C.
3. Desarrollo en VHDL de la UART.
4. Simulación y pruebas sobre el sistema físico.

3 Antecedentes

Hoy en día un tema muy recurrente es la Industria 4.0, también conocida como la cuarta revolución industrial, consistente en la completa digitalización de la industria. Si esto se está pudiendo llevar a cabo es gracias a un concepto conocido como IOT, o Internet de las cosas, y al Big Data, entre otros. Una de las bases de esta revolución es la evolución de Internet hasta tal punto de que la gran mayoría de dispositivos son capaces de conectarse a él, permitiéndonos así realizar un control y manejo remoto desde cualquier parte, siendo esto lo que conocemos como Internet de las Cosas. El IOT podría verse como una red que permite conectar una gran cantidad de objetos y dispositivos, siendo su objetivo que todas las cosas pudieran llegar a estar interconectadas y facilitarnos así la vida. En el momento en el que el campo de aplicación de esta tecnología es la industria, pasa a conocerse como Internet Industrial de las Cosas (IIOT). La disponibilidad de acceso a internet en los dispositivos industriales permite aumentar la eficiencia de la producción industrial, reducir los tiempos de inactividad de las máquinas, una monitorización remota y obtención de datos en tiempo real. Aquí es donde entra en juego el Big Data, ya que estos avances permiten tener acceso a una cantidad de información inmensa, con el problema de almacenamiento que esto conlleva. Pero no debemos olvidar que toda esta información accesible debido al IIOT la obtenemos gracias a los sensores existentes en la industria, y no sólo en esta, sino en el día a día. Es por esto por lo que es tan importante poder recibir los datos proporcionados por los sensores como poder acceder a ellos en cualquier instante, así que no debemos dejar a un lado la complejidad del acoplamiento entre un sensor y el dispositivo que procesa sus datos.

Este proyecto se ha centrado en esta última parte expuesta. Hoy en día existen un gran número de formas de comunicación como pueden ser mediante interfaz SCSI, USB, SPI, I2C... Pero las dos últimas son las más comunes para comunicar circuitos integrados, es decir, el sensor con el dispositivo que procesa sus datos (la Nexys4 en este caso). Se ha optado por implementar la comunicación mediante interfaz I2C, que será explicada más adelante, aunque la FPGA soporta ambas; y este proceso se completa con el diseño de una UART para poder enviar los datos al PC y pasar a formar parte, como ya se ha explicado, de esa compleja red que interconecta todas las cosas.

El diseño de la comunicación se llevará a cabo de forma general para que mediante unas modificaciones mínimas, se pueda emplear en otros sensores con la misma interfaz de comunicación. Hoy en día existen infinidad de sensores con interfaz I2C, pero en este proyecto se va a emplear un sensor de pulso. Esta elección se ha llevado a cabo ya que cada día la electrónica adquiere más importancia en el campo de la medicina, y no solo a nivel profesio-

nal como puede ser en los quirófanos, sino también en la vida cotidiana. Cada vez la gente tiene más conciencia de la importancia de tener unos hábitos saludables y de la posibilidad de prevenir ciertas enfermedades o problemas de salud gracias a dispositivos electrónicos como tensiómetros, pulsímetros, termómetros digitales,etc. Además, los smartwatches y los wearables se han puesto muy de moda, ya que es una tecnología reciente que permite dar otro paso en el IOT. La mayoría de los wearables disponen de un sensor de pulso que permite medir la frecuencia cardíaca y son muy empleados a la hora de hacer deporte.

Debido al auge de esta tecnología, a su gran campo de aplicación, y a que a pesar de que durante la carrera se han utilizado muchos sensores, pero nunca uno de este tipo, ha resultado especialmente interesante su implementación en el proyecto.

3.1. Funcionamiento oxímetro de pulso

La mayoría de los sensores de pulso emplean mediciones PPG (fotopletismografía) para obtener sus resultados, siendo estos la saturación de oxígeno en los vasos sanguíneos o las pulsaciones por minuto de un individuo. Estas mediciones se basan en las variaciones del color de la sangre, puesto que dependiendo de la cantidad de oxígeno que transporta, sus propiedades ópticas varían en las regiones del espectro visible (entre 400 y 700nm) y en las próximas al infrarrojo (entre 700 y 1000nm), ya que cuando la sangre está oxigenada tiene color rojo, mientras que cuando tiene una cantidad de oxígeno baja se caracteriza por una coloración azul oscura. Es debido a este efecto que estos sensores pueden utilizar dos longitudes de onda, una en cada uno de estos dos espectros, para realizar las mediciones.

En la región roja del espectro visible, alrededor de los 660nm, la hemoglobina (Hb) absorbe una mayor cantidad de luz que la oxihemoglobina (HbO_2), es decir, hemoglobina asociada con oxígeno. Sin embargo, en la región próxima al espectro infrarrojo, cerca de 940nm, sucede lo contrario. En la figura 3.1.0.1 se ve representado el coeficiente de absorción óptica de ambos componentes de la sangre a diferentes longitudes de onda.

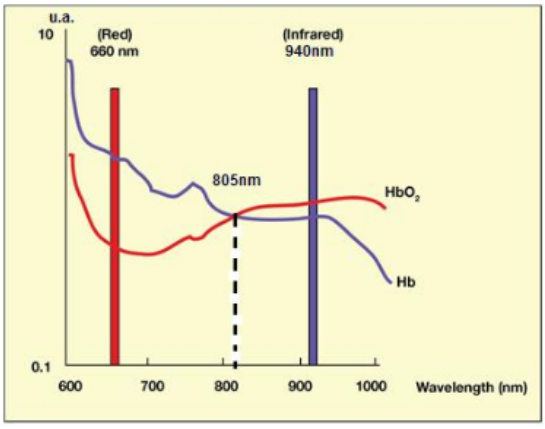


Figura 3.1.0.1 – Absorción de la luz sanguínea oxigenada frente a la no oxigenada en espectro IR y rojo [1]

El inconveniente de estas mediciones es que no solo dependen de la hemoglobina y oxihe-

moglobina, puesto que no son los únicos elementos que dispersan la luz emitida por el sensor, sino que también influyen otros factores como el color o grosor de la piel, de los tejidos o de los músculos.

La oximetría de pulso se lleva a cabo mediante la emisión de un haz de luz sobre la piel del individuo mediante un led integrado en el sensor empleado, y la recepción de la señal devuelta por los vasos sanguíneos, siendo esta señal dependiente de las variables ya mencionadas anteriormente y de la variación de la cantidad de sangre arterial que circula. Este tipo de técnica se conoce como fotopletismografía, photoplethysmographic en inglés, término del que proceden las siglas PPG. La ventaja de esta medición es que no es invasiva como puede ser la gasometría, pero no permite medir tantos parámetros como esta última y es menos exacta. Además, se realiza en extremidades como pueden ser los dedos de las manos o pies, o en los lóbulos de las orejas, mientras que la gasometría mide en el interior de las arterias.

La cantidad de sangre arterial está directamente relacionada con las contracciones y relajaciones periódicas del corazón. En la sístole, es decir, durante la contracción del corazón, hay una mayor cantidad de sangre circulando por las arterias hacia el resto del cuerpo, por lo tanto esta absorbe más luz y la intensidad de la señal PPG recibida va a ser menor. Sin embargo, durante la diástole, es decir, en la relajación del corazón, hay una menor cantidad de sangre circulando por las arterias ya que la presión de estas disminuye y por lo tanto sucede lo contrario, se devuelve más luz que antes.

La señal PPG recibida tiene dos componentes: una pulsátil considerada como la componente alterna (AC) y otra no pulsátil considerada como la continua (DC) y generada principalmente por la sangre venosa, piel y tejido. Se pueden compensar las posibles desviaciones de la sensibilidad del brillo o del detector del led, así como la parte continua de la señal que no tiene efecto sobre la medición, dividiendo la componente alterna entre la continua tanto en el espectro infrarrojo (IR) como en el rojo (R):

$$\frac{R}{IR} = \left(\frac{AC_R}{DC_R} / \frac{AC_{IR}}{DC_{IR}}\right) \tag{3.1.0.1}$$

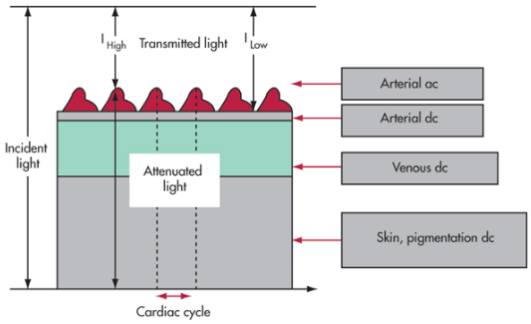


Figura 3.1.0.2 – Variaciones en la atenuación causadas por el tejido [1]

Podemos relacionar empíricamente este cociente con la SpO_2, es decir, el nivel de saturación periférica de oxígeno en sangre, que mide en porcentaje la cantidad de moléculas de hemoglobina unidas al oxígeno. Los oxímetros se pueden calibrar mediante los datos reco-

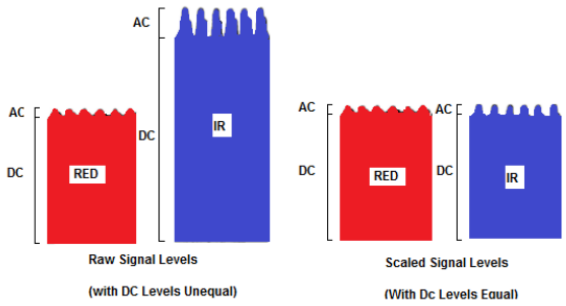


Figura 3.1.0.3 – Normalización de longitudes de onda R e IR [1]

gidos por un cooximetro, dispositivo que mide la pérdida de capacidad de oxigenación de la hemoglobina midiendo la cantidad de CO en el aire espirado, buscando un valor empírico de SpO_2, y podemos obtener una estimación de la saturación arterial de oxígeno (SaO_2), la cual se obtiene mediante métodos invasivos como la gasometría, mediante la siguiente ecuación:

$$SaO_2 = A - B \cdot (R/IR) \tag{3.1.0.2}$$

Siendo A y B dos coeficientes de regresión lineales relacionados con los coeficientes de absorción específicos de la hemoglobina y de la oxihemoglobina.

Los oxímetros de pulso leen el SaO_2 de la sangre de forma suficientemente exacta para uso clínico bajo condiciones normales puesto que utilizan una curva de calibración basada en datos empíricos como se muestra en la figura 3.1.0.4 .

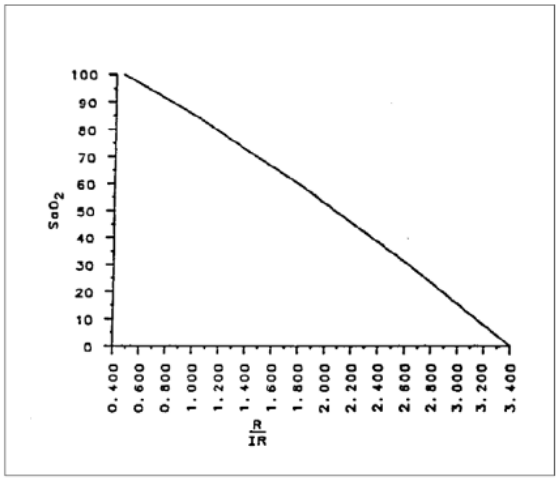


Figura 3.1.0.4 – Relación empírica entre SaO2 arterial y cociente R/IR normalizado [1]

Para una persona sana, los niveles normales de SaO_2 y de SpO_2 son entre el 95 y el 100 %, por lo que si los valores medidos son inferiores a este rango sería necesario que el individuo fuera al médico o se le tratara con mayor urgencia dependiendo del valor obtenido.

En el caso de la medición de la frecuencia cardíaca, que es la medida que se empleará en este proyecto, los valores normales de una persona en reposo son entre 40 (siendo el caso de un atleta muy entrenado) y 100. Realizando una actividad deportiva estos valores están entre

100 y la frecuencia cardíaca máxima del individuo (Fcmax), calculándose esa de la siguiente forma:

$$Fcmax = 220 - ES \tag{3.1.0.3}$$

Siendo ES la edad del sujeto.

4 Normas y referencias

4.1. Bibliografía

[1] *SpO Pulse Ox Wrist Oximeter Reference Design*, Texas Tnstruments, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: http://www.ti.com/lit/ug/tidu124/tidu124.pdf

[2] *Si1171 Data Short. Optical Heart Rate Sensor Module*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/si1171-short.pdf

[3] *Si1172 Data Short. Biometric Sensor Module*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/si1172-short.pdf

[4] *Si1173 Data Short. Biometric Sensor Module*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/si1173-short.pdf

[5] *Si1175 Data Short. Optical Heart Rate Sensor Module*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/si1175-short.pdf

[6] *Si1141/42/43-M01*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-sheets/Si1141-42-43-M01.pdf

[7] *Si1181 Data Short Optical Heart Rate Sensor Module*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/si1181-short.pdf

[8] *Si1182 Data Short Biometric Sensor*, Silicon Labs, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://www.silabs.com/documents/public/data-shorts/Si1182-short.pdf

[9] *MAX30112 Optimized Pulse-Oximeter and Heart Rate AFE for Wearable Health*, Maxim Integrated, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX30112.pdf

[10] *MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*, Maxim Integrated, [Fecha de la consulta: 20 de Febrero de 2019]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf

[11] *MAX14595 Low-Power Dual-Channel Logic-Level Translator*, Maxim Integrated, [Fecha de la consulta: 21 de Febrero de 2019]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX14595.pdf

[12] *MAX1920/MAX1921 Low-Voltage, 400mA Step-Down DC-DC Converters in SOT23*, Maxim Integrated, [Fecha de la consulta: 21 de Febrero de 2019]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX1920-MAX1921.pdf

[13] *Nexys 4 DDR Reference Manual*, Digilent, [Fecha de la consulta: 19 de Febrero de 2019]. Disponible en: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual

[14] *Pulse Oximeter With Much Improved Precision*, [Fecha de la consulta: 15 de Mayo de 2019]. Disponible en: https://www.instructables.com/id/Pulse-Oximeter-With-Much-Improved-Precision/

[15] *MAXREFDES117#: HEART-RATE AND PULSE-OXIMETRY MONITOR*, Maxim Integrated [Fecha de la consulta: 05 de Marzo de 2019]. Disponible en: https://www.maximintegrated.com/en/design/reference-design-center/system-board/6300.html

[16] *Comunicación I2C*, Sergio Andrés Castaño Giraldo [Fecha de la consulta: 20 de Junio de 2019]. Disponible en: https://controlautomaticoeducacion.com/microcontroladores-pic/comunicacion-i2c/

[17] *Descripción y funcionamiento del Bus I2C* [Fecha de la consulta: 20 de Junio de 2019]. Disponible en: http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/

[18] *Capítulo 21: Baudios y transmisión* [Fecha de la consulta: 20 de Junio de 2019]. Disponible en: https://github.com/Obijuan/open-fpga-verilog-tutorial/wiki/Cap%C3%ADtulo-21:-Baudios-y-transmisi%C3%B3n

[19] *Wikipedia: the free encyclopedia*. Wiki en Internet. Wikimedia Foundation, Inc. 2001. [Fecha de la consulta: 21 Junio 2019]. Disponible en: http://en.wikipedia.org/

4.2. Software empleado

- **Mathworks MATLAB 2015b:** Lenguaje de programación de alto nivel que permite el rápido desarrollo de scripts de prueba y verificación. Se ha empleado para la recepción de los datos de la FPGA a través de la UART.

- **Microsoft Office Excel 2013:** empleado para el almacenaje de datos del sensor, así como visualización de estos en gráficas.
- **Xilinx ISE Design Suite 14.7:** Software para la descripción de sistemas digitales mediante lenguaje VHDL.
- **Xilinx ISim Simulator:** Software para la simulación de sistemas digitales descritos mediante lenguaje VHDL.

5 Definiciones y abreviaturas

- **PPG:** de las proviene de la palabra photoplethysmography. La fotopletismografía permite conocer el volumen de un cuerpo a partir de la luz que refleja.
- **Hb:** hemoglobina.
- **HbO$_2$:** oxihemoglobina.
- **SpO$_2$:** saturación periférica de oxígeno en sangre.
- **SaO$_2$:** saturación arterial de oxígeno en sangre.
- **F$_{cmax}$:** frecuencia cardíaca máxima del individuo.
- **ECG:** electrocardiograma. Registra la actividad eléctrica del corazón que se produce en cada latido.
- **I2C:** Inter-Integrated Circuit, en español circuito inter-integrado. Es un tipo de comunicación entre circuitos integrados.
- **SPI:** Serial Peripheral Interface, en español interfaz de periféricos serie. Es un tipo de comunicación entre circuitos integrados.
- **AD:** analógico-digital. Suele emplearse para denominar un tipo de convertidores.
- **FIFO:** First In First Out, es decir, primero en entradar primero en salir. Se emplea para determinar un tipo de almacenaje, como sucede con algunos tipos de memorias.
- **LSB:** bit menos significativo.
- **MSB:** bit más significativo.
- **UART:** Transmisor-Receptor Asíncrono Universal.
- **f.a.:** función de autocorrelación.

6 Requisitos de diseño

En este proyecto se pretende realizar la comunicación entre una FPGA y un sensor de pulso empleando el protocolo de comunicación I2C.

Es necesario programar y controlar la comunicación con el sensor de pulso escogido respetando los tiempos establecidos en la hoja de características de este, así como realizar una comunicación general que sirva para otros sensores que utilicen este tipo de protocolo. Además, se procesará la señal obtenida del sensor de forma que se obtenga la frecuencia cardíaca de la manera más exacta posible.

Este proyecto se complementa con la realización de una UART en la FPGA para comunicar esta con el PC, de forma que se puedan visualizar los datos leídos del sensor en el ordenador, procesarlos, guardarlos o realizar las pruebas que se deseen.

La frecuencia cardíaca obtenida deberá visualizarse en el display de la Nexys4 DDR.

7 Análisis de las soluciones

Actualmente en el mercado hay un gran número de sensores de pulso a disposición de cualquiera que quiera adquirirlos, por lo que es necesario realizar un estudio de mercado para elegir el sensor que mejor se pueda adecuar a la realización de este proyecto.

7.1. Sensores de pulso en el mercado

A continuación, se presentan distintos sensores de pulso existentes en el mercado y sus principales características, así como la justificación de su uso o no en el presente proyecto. Los sensores que carecen de protocolo I2C ya no se han considerado puesto que este es un requisito indispensable para realizar la comunicación con la FPGA.

7.1.1. Si117x (1171/1172/1173/1175)

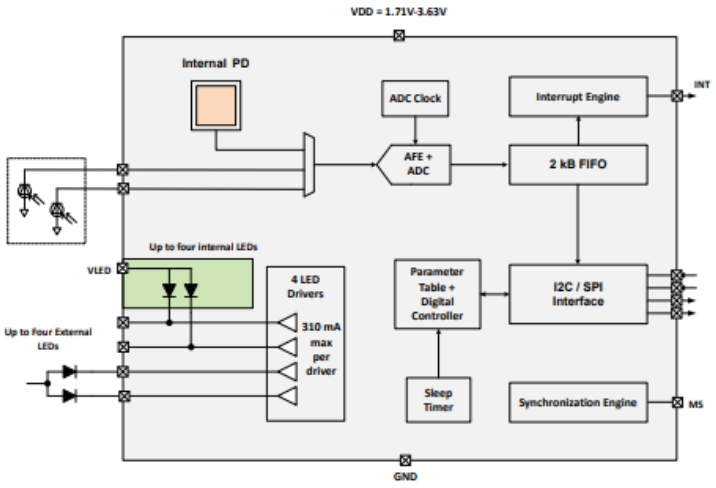


Figura 7.1.1.1 – Esquema de bloques del sensor Si117x [2]

Estos sensores ópticos de Silicon Labs poseen 4 drivers leds, es decir, tienen capacidad para soportar hasta 4 leds de diferentes longitudes de onda, variando, dependiendo del modelo, entre 2 y 3 el número de leds que tienen integrados, y pudiendo añadirse el resto externamente hasta completar los 4. Los leds internos soportan hasta 100 o 310mA dependiendo del modelo, y para los externos el límite de corriente es siempre de 310mA. Los sensores tienen un fotodetector de 1 mm^2 y se les puede añadir hasta dos más externos.

Los modelos 1172 y 1173 tienen una interfaz que soporta mediciones ECG (para electrocardiograma) independientes de las mediciones PPG, es decir, las que se emplearán en el proyecto.

Todos los modelos disponen de un convertidor AD interno con una resolución de 24 bits, un procesador de comunicaciones, una memoria FIFO de 2kB e interfaz de comunicaciones I2C y SPI con salida de interrupción programable. Además, son de bajo consumo puesto que su corriente media es inferior a 50μA y su tensión de funcionamiento está entre 1.71 y 3.63V, soportando así los 3.3V proporcionados por la Nexys 4.

7.1.2. Si114x-AAGX-GM (1143/1144)

Ambos sensores de Silicon Labs tienen integrados un led verde perfecto para mediciones en la muñeca debido a su longitud de onda de 525nm, y soporte para otros dos leds externos con un espectro de entre 525 y 940nm, por lo que poseen tres drivers leds con diversos niveles seleccionables programables para operar entre 6 y 360mA.

También disponen de un convertidor AD, filtros, fotodiodos de alta sensibilidad, un procesador de comunicaciones, salida de interrupciones, interfaz de comunicación I2C, velocidad de datos de hasta 3.4 Mbps, un tiempo de inicio de 25ms y una tensión de funcionamiento de entre 1.71 a 3.6V además de bajo consumo con una corriente de espera inferior a 500nA.

La diferencia existente entre el modelo 1143 y el 1144 es que este último incluye un algoritmo para compensar el movimiento mediante el uso de datos de un acelerómetro externo.

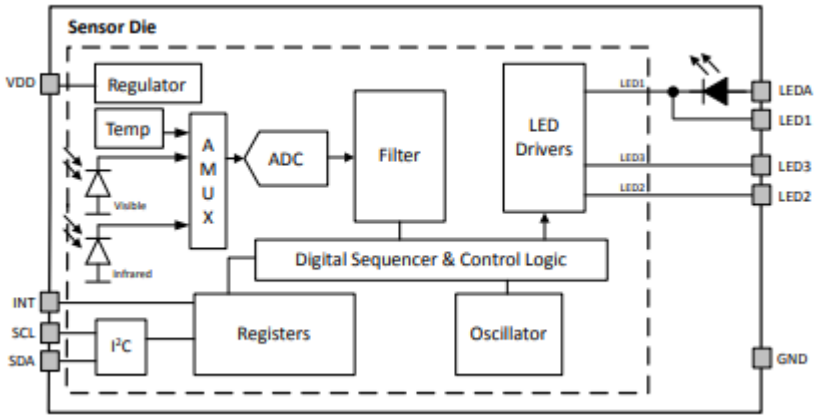


Figura 7.1.2.1 – Esquema de bloques del sensor Si114x-AAGX-GM [6]

7.1.3. Si118x-GM (1181/1182)

Sensores de Silicon Labs que disponen de 4 drivers para leds que se pueden programar independientemente unos de otros para operar con una corriente de entre 1.7 a 310mA. Estos drivers les permiten tener una capacidad de hasta 4 leds externos puesto que no tienen ninguno integrado. Poseen un fotodiodo de 1 mm_2 y se les puede añadir hasta 2 externos.

A diferencia del 1181, el modelo 1182 permite realizar mediciones de tipo ECG, independientes de las mediciones PPG.

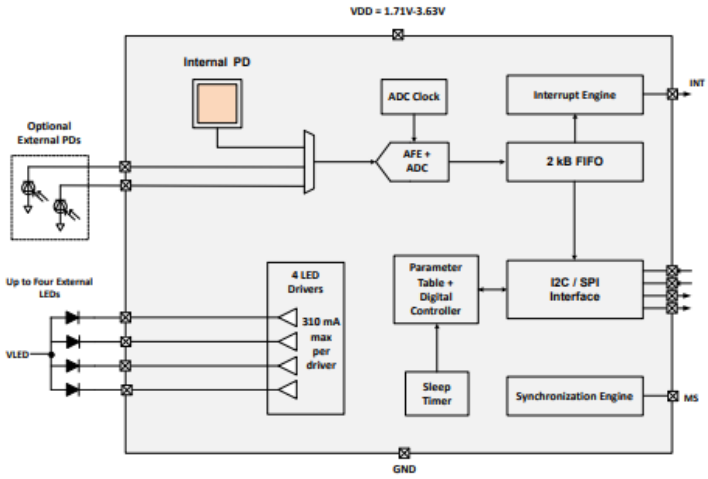


Figura 7.1.3.1 – Esquema de bloques del sensor Si118x-GM [7]

También tienen integrado un convertidor AD de 24 bits, una memoria FIFO de 2kB, un procesador de comunicaciones, soporte para comunicaciones con interfaz I2C y SPI, salida por interrupción y soporte para sincronizarse con un acelerómetro externo. Además, tienen bajo consumo puesto que su corriente media es inferior a 50μA.

Estos sensores no se han seleccionado ya que carecen de los leds necesarios para realizar las mediciones.

7.1.4. MAX 30112

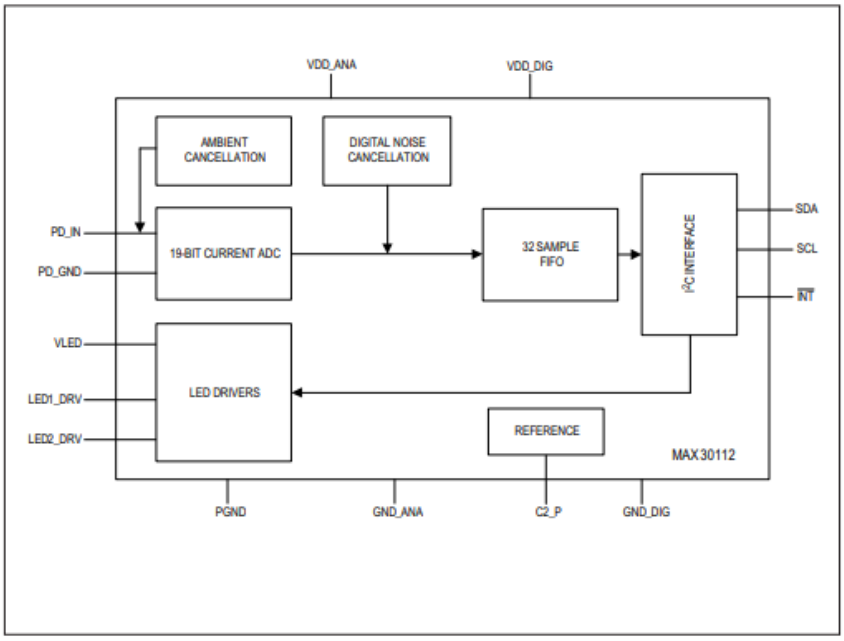


Figura 7.1.4.1 – Esquema de bloques del sensor MAX30112 [9]

Este sensor de Maxim Integrated carece de leds integrados, pero tiene capacidad para soportar hasta 2 leds externos funcionando individualmente o a la vez con una corriente se-

leccionable de 50, 100, 150 o 200mA. También carece de fotodiodo, pero está adaptado para soportar uno externo.

Posee un convertidor AD de 19 bits que permite hacer entre 20 y 3200 muestras por segundo, una memoria FIFO de 32 muestras, filtros, soporta interfaz de comunicación I2C estándar, tiene salida de interrupción programable, así como modos de apagado mediante software con corriente de espera casi nula (1.6μA). Si la corriente que circula por el fotodiodo es inferior o igual a 200μA el sensor cancela la luz ambiental, haciendo así que pueda trabajar bajo condiciones de luz ambiental alta. Su coste es de 2.11e.

Su tensión de funcionamiento está entre 1.7 y 2V, rango inferior a los 3.3V que proporciona la Nexys4. Además, hay que alimentar los leds a una tensión entre 3.1 V y 5.25V, por lo que es necesario hacer llegar dos tensiones distintas al sensor.

Debido al problema a la hora de alimentar la distinta tensión al sensor y sus leds, además de su carencia de estos y de fotodiodo, este sensor no se ha escogido para la realización del proyecto.

7.1.5. MAX30102

Sensor de Maxim Integrated con una tensión de alimentación necesaria de entre 1.7 y 2V y corriente de 600μA. Este incluye un led rojo y otro infrarrojo con una tensión de funcionamiento de entre 3.1 y 5.25V, con corriente programable entre 0 y 50mA y ancho de pulso seleccionable entre 69μs y 411μs, además de un fotodetector que trabaja entre 600 y 900nm. También tiene un convertidor AD de 18 bits, una memoria FIFO de 32muestras y rechazo a la luz ambiental, lo que ayuda a que las mediciones sean más precisas. Además, admite comunicación de datos mediante interfaz estándar I2C, pudiendo apagarse el módulo mediante software con corriente de espera casi nula de 0.7μA.

Otra característica destacable es que posee una función de proximidad para reducir el consumo y la emisión de luz cuando el sensor no está cerca del dedo del usuario. Cuando se activa el modo de funcionamiento para realizar las mediciones, el led infrarrojo se pone en modo de proximidad con una corriente baja establecida por uno de los registros y cuando se detecta un objeto próximo se activa el modo normal.

También posee un sensor que mide la temperatura en el circuito integrado, lo que permite compensar el error de la medición SpO_2 relacionada a los cambios de la temperatura ambiente, ya que las mediciones del led rojo se ven afectadas por esta. En cambio, las mediciones realizadas por el led de infrarojo son independientes de la temperatura interna del sensor.

La empresa Maxim Integrated también comercializa la placa MAXREFDES117, en la que se incluye este sensor y el acondicionamiento necesario, lo que facilita su implementación en el proyecto.

Debido a lo completo que es este sensor, su bajo consumo y la existencia de la placa de acondicionamiento que lo integra, ha sido escogido para la realización del presente proyecto.

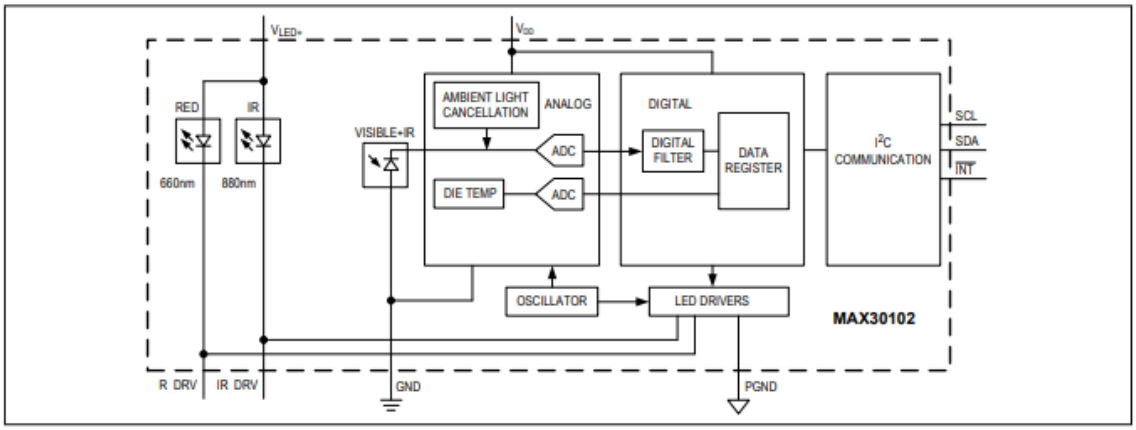


Figura 7.1.5.1 – Esquema de bloques del sensor MAX30102 [10]

7.1.6. Tabla comparativa

A continuación se muestra una tabla resumen con las características de los distintos sensores expuestos, justificando así la elección del sensor MAX30102 al ser el más completo y el que mejor se adecúa al proyecto a realizar.

	Sensor				
	Si117x	**Si114x**	**Si118x**	**MAX30112**	**MAX30102**
Leds Integrados	Si	No	No	No	Si
Fotofiodo Integrado	Si	Si	Si	No	Si
Resolución ADC (bits)	24	17	24	19	18
Corriente Leds (mA)	<0.5	0.15-1.4	<0.5	1.6	0.7
Precio (€)	8	3-4	2.57	2.11	3.52

Tabla 7.1.6.1 – Tabla comparativa de los distintos sensores de pulso en existentes el mercado.

Una característica determinante a la hora de escoger el sensor ha sido si este posee o no leds integrados, ya que si es necesario añadírselos el coste aumentará, además de que implicaría la necesidad de conexiones externas y posibles fallos. Puesto que solo dos de los sensores tratados tienen leds, se reduce así la elección al Si117x y al MAX30102. Ambos sensores cumplen todas las características necesarias para la realización del proyecto, además de que los dos tienen bajo precio y consumo, pero finalmente se empleará el MAX30102 puesto que en la página del fabricante se puede obtener un datasheet mucho más completo que en

el caso del Si117x, lo que facilitará la realización del proyecto, además de la existencia de la placa MAXREFDES117 que ya acondiciona la tensión y corriente de entrada al sensor.

7.2. Placa MAXREFDES117

Se ha decidido realizar el proyecto con la placa MAXREFDES117 en lugar de con el sensor únicamente ya que a pesar de que encarece significativamente el precio (ahora pasa a costar 15€), facilita considerablemente la comunicación con la FPGA, además de que ya están todos los circuitos necesarios integrados en un mismo dispositivo. Esta placa tiene 2 circuitos integrados a mayores del sensor de pulso MAX30102, que son el traductor de nivel MAX14595 y el convertidor MAX1921, como se muestra en la figura 7.2.0.1.

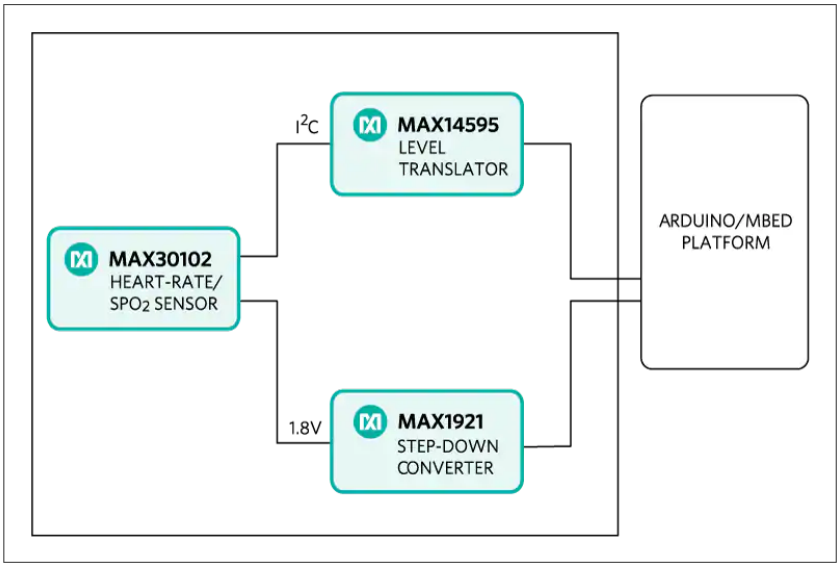


Figura 7.2.0.1 – Esquema de bloques de la placa MAXREFDES117 [15]

De este modo, con estos 2 circuitos integrados se solucionan 2 posibles problemas que pueden surgir al realizar la comunicación entre la Nexys y el sensor: la diferente tensión de alimentación de los leds con respecto a la del sensor de pulso, y los diferentes niveles lógicos a los que trabaja el sensor y la FPGA. Este último lo solventa el MAX14595. Este módulo permite que tanto el sensor como la FPGA puedan interpretar correctamente el valor transmitido por el bus I2C sin que haya ningún error debido a niveles lógicos distintos.

7.2.1. MAX14595

El MAX14595 tiene el esquema de la figura 7.2.1.1. Es un traductor de nivel lógico de dos canales bidireccionales. Este tiene dos alimentaciones, VL y VCC. La primera es la de menor valor y debe ser mayor o igual que 0.9V y menor o igual que VCC; la segunda debe estar comprendida entre 1.65 y 5.5V ambos inclusive.

La señal TS activa el integrado, y este no funcionará cuando TS esté a 0 o cuando el valor de Vcc sea menor que el de V_L. Cuando tenemos un 1 en TS, la señal lógica presente en el lado de V_L aparece en el lado de VCC y viceversa. Este integrado permite el continuo

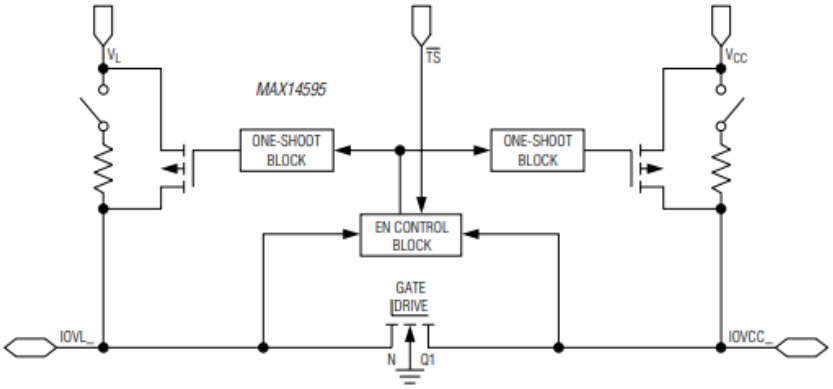


Figura 7.2.1.1 – Diagrama de bloques interno del MAX14595 [11]

funcionamiento del bus I2C en el lado con alimentación, aunque la función de traducción de nivel esté apagada, mientras el pin TS esté a nivel alto. Además, es óptimo para trabajar a altas velocidades y en colector abierto como puede suceder en los buses I2C.

7.2.2. MAX1921

Este integrado es un convertidor de tensión DC-DC con una corriente de suministro estable muy baja (50μA) y el cual garantiza una corriente de alrededor de 400mA para una salida de tensión mínima de hasta 1.25V. Tiene un rectificador síncrono interno que permite alcanzar una eficiencia superior al 90 % y eliminar el diodo Schottky necesario en los convertidores convencionales. Su frecuencia máxima de conmutación es de 1.2MHz, lo que permite utilizar componentes externos pequeños y de bajo coste. Incluye un arranque suave para limitar la corriente inicial y así reducir los requisitos del condensador de entrada. Su tensión de salida es ajustada de fábrica, al contrario que el modelo MAX1920 que es seleccionable, y puede ser de 3.3, 3.0, 2.4, 1.8 (la empleada en este caso) y 1.5V. Este dispositivo es una alternativa muy eficiente a los reguladores lineales en aplicaciones con un espacio reducido.

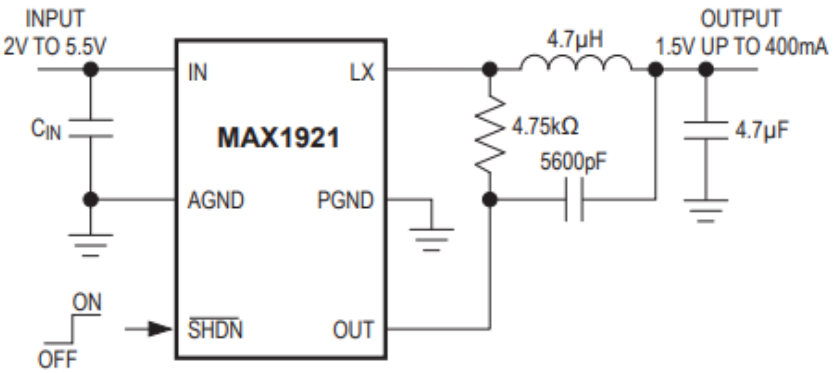


Figura 7.2.2.1 – Circuito típico de funcionamiento del MAX1921 [12]

Además, como los leds del MAX30102 trabajan con un ciclo de trabajo pequeño para aho-

rrar energía, las corrientes pulsantes que generan pueden provocar picos en el pin de alimentación VLED+, por lo que es necesario que el convertidor DC-DC tenga una resistencia e inductancia mucho menor de 1Ω y un condensador de al menos 1μF derivado de la alimentación a tierra, requisitos que cumple este integrado.

7.3. Configuración del sensor MAX30102

Este sensor se caracteriza por tener varios registros de configuración, los cuales es necesario escribir para que el sensor funcione en el modo correcto y así poder realizar las mejores mediciones posibles.

El sensor de pulso empleado posee 2 leds, uno rojo y otro infrarrojo, de forma que seleccionando el modo de funcionamiento se puede utilizar uno, otro u ambos.

Posee una memoria FIFO que permite alcanzar 32 medidas por canal, es decir, 32 muestras por cada led que se esté empleando. Una muestra está formada por 3 bytes, de forma que el primer byte que se lee contiene los bits más significativos. El tamaño de cada muestra varía dependiendo de la resolución del AD, siendo esta seleccionable entre 15 y 18 bits ambos inclusive. Los datos están guardados en la FIFO siguiendo el esquema de la figura 7.3.0.2. Si la resolución del AD fuera de 15 bits, los 3 bits menos significativos del tercer byte no proporcionarían información.

BYTE 1							FIFO_ DATA[17]	FIFO_ DATA[16]
BYTE 2	FIFO_ DATA[15]	FIFO_ DATA[14]	FIFO_ DATA[13]	FIFO_ DATA[12]	FIFO_ DATA[11]	FIFO_ DATA[10]	FIFO_ DATA[9]	FIFO_ DATA[8]
BYTE 3	FIFO_ DATA[7]	FIFO_ DATA[6]	FIFO_ DATA[5]	FIFO_ DATA[4]	FIFO_ DATA[3]	FIFO_ DATA[2]	FIFO_ DATA[1]	FIFO_ DATA[0]

Figura 7.3.0.2 – Representación de una muestra en la FIFO [10]

Si el sensor está funcionando en modo SpO_2, es decir, con los 2 leds en funcionamiento, en la FIFO se guardan alternativamente una muestra del led rojo y otra del led IR por este orden, hasta completar las 32 muestras de cada canal.

Esta FIFO nos permite que no sea necesario leer una muestra cada vez que el sensor realiza una medida, sino que podemos leerla cada cierto número de espacios libres en la FIFO o cuando ya esté llena.

Para poder leer las muestras del sensor, es necesario configurar 11 registros:

- **Interrupt Enable 1 (dir:0x02):** se configura con el valor 0x80 para poner a nivel alto el bit 7 y así habilitar la interrupción **A_FULL** del registro de interrupción 0x00. De esta forma, el MSB del registro 0x00 se activa a nivel bajo cuando, en modo SpO_2 y HR, el puntero de escritura de la FIFO tiene cierto número de espacios libres. Se puede configurar con cuantos espacios libres se activa esta interrupción. Se configura así para leer varias muestras de la FIFO y no una muestra de cada vez.

- **Interrupt Enable 2 (dir:0x03):** en este registro se escribe un 0x00 para desactivar la interrupción de conversión de temperatura interna finalizada (bit 1 del registro de interrupción

0x01).

- **FIFO Write Pointer (dir:0x04):** este registro apunta a la localización en la que el sensor escribirá la siguiente muestra. Se actualiza sólo cada vez que una muestra es guardada en la FIFO. Inicialmente es recomendable escribirle el valor 0x00 antes de entrar en el modo de funcionamiento de SpO_2 o HR para asegurarse de que no hay muestras antiguas en la FIFO.
- **Over Flow Counter (dir:0x05):** cuenta el número de muestras perdidas cuando la FIFO está llena. Este se resetea a 0 cuando una muestra completa se extrae de la FIFO. Como en el anterior registro, inicialmente es recomendable escribirle el valor 0x00 antes de entrar en el modo de funcionamiento de SpO_2 o HR para asegurarse de que la FIFO está vacía y en un estado conocido.
- **FIFO Read Pointer (dir0x06):** este registro apunta a la localización de la cual el procesador leerá la siguiente muestra de la FIFO. Se actualiza sólo cada vez que una muestra es leída de la FIFO. Igual que los anteriores registros, inicialmente es recomendable escribirle el valor 0x00 antes de entrar en el modo de funcionamiento de SpO_2 o HR para asegurarse de que la FIFO está vacía y en un estado conocido.
- **FIFO Configuration (dir:0x08):** registro para configurar la FIFO. Dependiendo de los bits en los que se escriba, se configura una opción u otra de la FIFO. Se escribe el dato 0x40 para configurar los bits de la siguiente forma:

  1. **SMP_AVE (Bits 7:5)**: se configura el número de muestras promediadas. En este caso se configurará con el valor “010”, es decir, una muestra de la FIFO se obtiene de promediar 4 lecturas. Las distintas configuraciones se pueden ver en la figura 7.3.0.3.

SMP_AVE[2:0]	NO. OF SAMPLES AVERAGED PER FIFO SAMPLE
000	1 (no averaging)
001	2
010	4
011	8
100	16
101	32
110	32
111	32

Figura 7.3.0.3 – Configuración del número de muestras promediadas del MAX30102 [10]

  2. **FIFO_ROLLOVER_EN (Bit 4)**: si se pone este bit a nivel bajo, la FIFO no se vuelve a llenar una vez esté completa, sino que espera a que se vacíe. Así se configura este bit.
  3. **FIFO_A_FULL (Bits 3:0)**: establece el número de muestras libres en la FIFO con el que se activa la interrupción A_FULL. En este caso se configuran estos bits con el valor 0x0h, de forma que la interrupción se activa cuando la FIFO está llena (32 muestras), pero se puede ver en la figura 7.3.0.4 las distintas configuraciones posibles.

FIFO_A_FULL[3:0]	EMPTY DATA SAMPLES IN FIFO WHEN INTERRUPT IS ISSUED	UNREAD DATA SAMPLES IN FIFO WHEN INTERRUPT IS ISSUED
0x0h	0	32
0x1h	1	31
0x2h	2	30
0x3h	3	29
...	...	...
0xFh	15	17

Figura 7.3.0.4 – Configuración de muestras libres en la FIFO al activarse la interrupción [10]

- **Mode Configuration (dir:0x09):** en este registro se establece el modo de funcionamiento del sensor. Dependiendo de los bits en los que se escriba, se configura una opción u otra del sensor. Se escribe el dato 0x03 para configurar los bits de la siguiente forma:

  1. **SHDN (Bit 7)**: al poner este bit a nivel alto, el sensor se pone en modo de ahorro de energía. Se pondrá este bit a 0.
  2. **RESET (Bit 6)**: al activar este bit a nivel alto se pone toda la configuración y los registros del sensor en su estado por defecto. Este bit se limpia automáticamente al finalizar la secuencia de reset. Se pone este bit a nivel bajo.
  3. **MODE (Bits 2:0)**: determina el modo de funcionamiento del sensor. Se configura con el dato "011", de manera que este operará en modo SpO_2, es decir, funcionando los 2 leds. En la figura 7.3.0.5 se pueden ver todos los modos de funcionamiento y su configuración.

MODE[2:0]	MODE	ACTIVE LED CHANNELS
000	Do not use	
001	Do not use	
010	Heart Rate mode	Red only
011	SpO2 mode	Red and IR
100–110	Do not use	
111	Multi-LED Mode	Red and IR

Figura 7.3.0.5 – Modos de funcionamiento del MAX30102 y su configuración [10]

- **SpO_2 Configuration (dir:0x0A):** si el sensor funciona en modo SpO_2, cómo es este caso, es necesario escribir en este registro para configurar este modo de funcionamiento. Dependiendo de los bits en los que se escriba, se configura una opción u otra del sensor. Se escribe el dato 0x27 para configurar los bits de la siguiente forma:

  1. **SpO_2_ADC_RGE (Bits 6:5)**: determina el rango de fondo de escala del convertidor AD del sensor. En este caso se empleará un fondo de escala de 4096nA, por lo que se escribe un "01" en estos bits. En la figura 7.3.0.6 se pueden ver las posibles configuraciones.
  2. **SpO_2_SR (Bits 4:2)**: permite configurar la velocidad de muestreo. Se ha configurado este con "001" para realizar 100 muestras por segundo. En la figura 7.3.0.7 se pueden ver las distintas configuraciones.
  3. **LED_PW (Bits 1:0)**: controla el ancho de pulso de cada led (tanto el IR como el rojo tienen el mismo) y por tanto la resolución del AD. El máximo valor que puede tomar

SPO2_ADC_RGE[1:0]	LSB SIZE (pA)	FULL SCALE (nA)
00	7.81	2048
01	15.63	4096
10	31.25	8192
11	62.5	16384

Figura 7.3.0.6 – Configuración de fondo de escala del ADC del MAX30102. [10]

SPO2_SR[2:0]	SAMPLES PER SECOND
000	50
001	100
010	200
011	400
100	800
101	1000
110	1600
111	3200

Figura 7.3.0.7 – Configuración de la velocidad de muestreo del MAX30102. [10]

este registro está directamente relacionado con la velocidad de muestreo, ya que a mayores velocidades menor será la posible resolución del AD. Se configura con el valor "11" para trabajar con una resolución de 18 bits. En la figura 7.3.0.8 se pueden ver las posibles configuraciones.

LED_PW[1:0]	PULSE WIDTH (µs)	ADC RESOLUTION (bits)
00	69 (68.95)	15
01	118 (117.78)	16
10	215 (215.44)	17
11	411 (410.75)	18

Figura 7.3.0.8 – Configuración del ancho de pulso de los leds del MAX30102[10]

- **LED1_PA (dir:0x0C) y LED2_PA (dir:0x0D):** estos registros determinan la corriente típica de cada led. Se configuran ambos con el valor de 0x24, de forma que esta corriente en cada uno de los leds será de 7.2mA, puesto que la unidad vale 0.2mA.
- **PILOT_PA (dir:0x10):** establece la potencia de los leds en el modo de proximidad. Se configura este con el valor 0x7F, lo que equivale a 25.4mA.

Antes de realizar estas 11 escrituras, se realiza una inicial en el registro de dirección 0x09 con el dato 0x40, es decir, activando únicamente el bit de reset, para realizar un reset inicial al sensor. Con esto se saca la conclusión de que es necesario realizar 12 operaciones de escritura.

7.4. Frecuencias a tener en cuenta

La frecuencia con la que la línea SCL debe variar está limitada por las especificaciones técnicas del MAX30102.

El datasheet de este sensor nos dice que la frecuencia máxima con la que puede variar este es de 400kHz, es decir, debe tener un periodo mínimo de 2.5μs. Además, en el diagrama

temporal se especifica que el tiempo mínimo que debe estar a nivel bajo es de 1.3μs y 0.6μs a nivel alto, lo que indica que la señal no tiene porqué estar el mismo tiempo a nivel alto como a nivel bajo.

Se ha decidido emplear una señal SCL de 100kHz, es decir, 10μs de período. El tiempo que esta señal está a nivel alto es ligeramente inferior que el que está a nivel bajo. Esta frecuencia es lo suficientemente alta para realizar las operaciones sin que lleven demasiado tiempo, pero sin aproximarse a la frecuencia máxima.

Puesto que las pulsaciones mínimas por minuto de una persona están entre 40, siendo el caso de un atleta entrenado, y las máximas se obtienen a partir de la fórmula 3.1.0.3, el valor calculado de pulsaciones que se podrá visualizar en la placa estará entre 30 y 220. Por debajo o por encima de este rango no se visualizará el resultado ya que no será correcto, sino que se verá el mensaje “DEDO” en la placa.

7.5. Número de muestras a evaluar

Se ha decidido emplear únicamente las muestras obtenidas por el led infrarrojo, puesto que a este le influyen menos las perturbaciones de la luz ambiente y temperatura.

Se configura el sensor para que lea 100 muestras por segundo y que haga una media de 4 muestras, por lo que la frecuencia con la que toma una muestra es de 100/4, es decir, 25 muestras por segundo.

Para realizar el cálculo de la frecuencia cardíaca, bastaría con medir la pulsación durante 10s, ya que en este tiempo se podrían obtener unos 12 máximos dependiendo de la pulsación de la persona. Puesto que la frecuencia de muestreo es de 25Hz, se toma una muestra cada 0.04s, por lo que se obtendrán 256 muestras para realizar el cálculo cada 10.24s.

La FIFO tiene una capacidad de 32 muestras, por lo que se leerá esta 8 veces para obtener las 256 muestras necesarias para el cálculo de la frecuencia cardíaca.

7.6. Estándar de comunicación I2C

El bus I2C (Inter-Integrated Circuit) para comunicación de circuitos integrados es un bus serie empleado generalmente para conectar circuitos integrados entre sí. Fue diseñado por Philips Semiconductors a principios de los 80 y tiene la principal característica que permite tener uno o varios maestros controlando uno o varios esclavos.

Igual que la UART y al contrario que en la comunicación SPI, el protocolo de comunicación I2C utiliza sólo 2 canales de comunicación: uno de reloj (SCL) y otro por donde se transmite la información (SDA). Este último es bidireccional, ya que desde este se envían los datos del maestro al esclavo y viceversa.

Ambas líneas son síncronas y de drenador abierto, esto es, similares a las de colector abierto pero en este caso se emplean transistores de tipo FET, por lo que es el drenador el que está abierto, permitiendo que las líneas puedan tener un tercer estado de alta impedancia y siendo necesario conectar una resistencia de pull-up para que puedan alcanzar nivel alto. La

transmisión de datos se realiza en paquetes de 8 bits, enviándose bit a bit y comenzando por el más significativo. En la figura 7.6.0.9 se muestra un ejemplo de conexión de comunicación I2C.

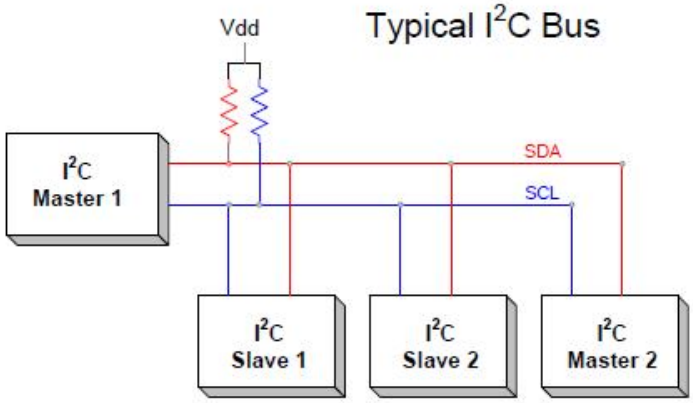


Figura 7.6.0.9 – Conexión maestro-esclavos en comunicación I2C [16]

Para realizar la comunicación mediante este protocolo, es necesario seguir una serie de pasos:

1. **Condición de start:** Inicialmente las dos líneas, tanto SDA como SCL están a nivel alto. En el momento en el que se desea comenzar la comunicación, el maestro debe poner SDA a nivel bajo mientras que SCL sigue a nivel alto. Un tiempo después el maestro deberá poner SCL a nivel bajo y comenzar así la señal de reloj periódica que sincronizará el maestro con el esclavo. En la figura 7.6.0.10 se representa esta operación.

2. **Dirección del esclavo:** una vez realizada la condición de inicio, el maestro debe enviar con cada ciclo a nivel alto de SCL cada bit que conforma la dirección del esclavo con la que desea comunicarse, estando compuesta esta por entre 7 y 10 bits dependiendo del esclavo, más el bit menos significativo, el cual indica si se desea realizar una operación de escritura o de lectura (a nivel bajo indica escritura), siendo lo más común que la dirección sea de 7 bits más el de operación, formando así un byte. Tanto en el envió como en la recepción de datos, primero se envía el bit más significativo.

3. **Reconocimiento (ACK):** Tras enviar la dirección del esclavo, en el noveno ciclo de SCL el esclavo debe realizar la acción de reconocimiento, es decir, poner la línea de SDA a nivel bajo para indicar que ha recibido correctamente el byte.

4. **Dirección del registro:** Tras enviar la dirección del esclavo y realizar el reconocimiento, el maestro debe enviar el byte con la dirección del registro del esclavo en el que se quiere leer o escribir, seguida nuevamente de una operación de reconocimiento del esclavo en el noveno ciclo de reloj.

5. **Operación de escritura:** en este punto es cuando difiere el procedimiento a seguir. Si se trata de una operación de escritura, tras enviar la dirección del registro, el maestro

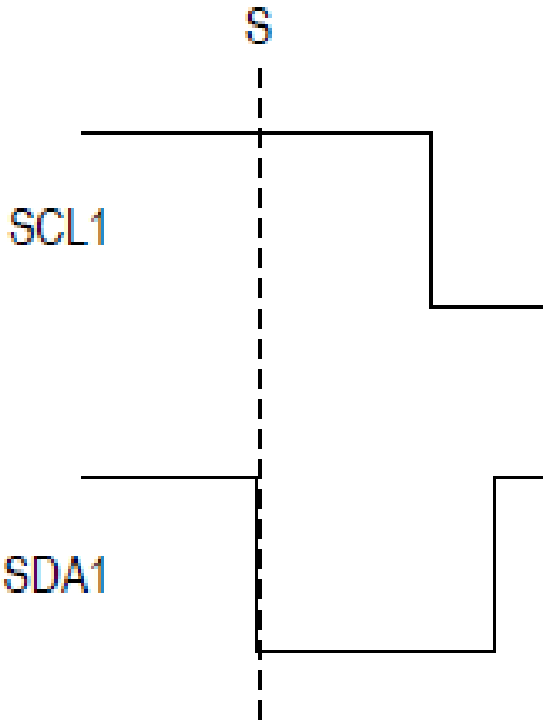


Figura 7.6.0.10 – Operación de START para comenzar la comunicación I2C.

enviará el byte del dato a escribir en dicho registro, y en el noveno ciclo de SCL deberá volver a esperar un reconocimiento del esclavo.

6. **Operación de lectura:** tras enviar la dirección del esclavo, para realizar una lectura es necesario hacer una operación de RESTART, es decir, durante el siguiente ciclo a nivel alto de SCL el maestro debe poner SDA a nivel bajo. Tras el restart, el maestro debe mandar nuevamente la dirección del esclavo a leer, ahora con el LSB de lectura a nivel alto. Tras realizar el ACK, el maestro debe poner SDA en alta impedancia para que el esclavo pueda enviar los 8 bits que conforman el dato contenido en el registro direccionado. Ahora es el maestro quien debe realizar la operación de reconocimiento, poniendo SDA a nivel bajo en el noveno ciclo de SCL para indicar que se ha realizado la comunicación correctamente. En caso de no querer realizar más lecturas, en lugar de realizar una operación de reconocimiento el maestro debe realizar una de no reconocimiento.

7. **No reconocimiento (NACK):** cuando no se desean leer más datos, el maestro debe poner SDA a nivel alto y permanecer constante durante el noveno ciclo de SCL para indicar un no reconocimiento.

8. **Operación de STOP:** tras el NACK, o tras escribir un byte, si no se desean leer ni escribir más bytes, es necesario realizar una operación de paro, la cual consiste en que el maestro hace cambiar de nivel a SDA pasando de nivel bajo a nivel alto mientras SCL permanece a nivel alto. En la figura 7.6.0.11 se representa esta operación.

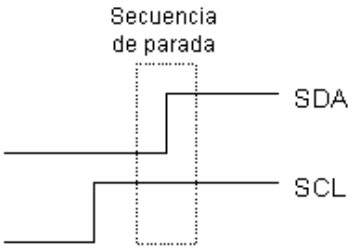


Figura 7.6.0.11 – Operación de STOP para finalizar la comunicación I2C [17]

En la figura 7.6.0.12 se muestra un ejemplo de escritura de un byte mediante la comunicación I2C, y en la figura 7.6.0.13 se puede ver un ejemplo de lectura de un byte siguiendo este protocolo.

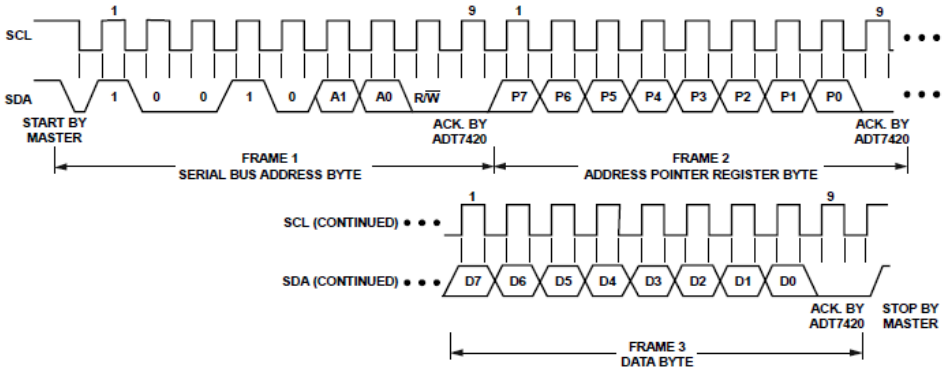


Figura 7.6.0.12 – Ejemplo de escritura de 1 byte mediante protocolo I2C [10]

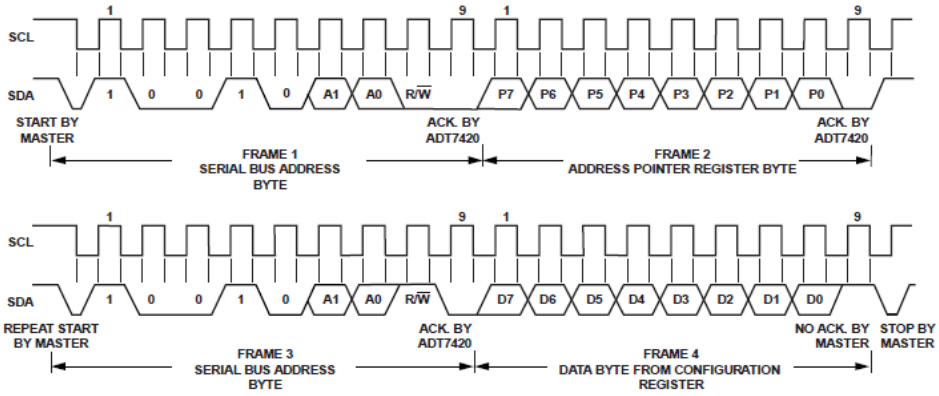


Figura 7.6.0.13 – Ejemplo de lectura de 1 byte mediante protocolo I2C [10]

Para realizar todas esta etapas de la comunicación, se ha optado por diseñar máquinas de estados, con la finalidad de respetar los tiempos establecidos en el datasheet, controlar la comunicación y realizar todas las operaciones de forma síncrona.

Existen dos tipos de máquinas de estados, de Mealy y de Moore. En el primer tipo las salidas dependen del estado actual y del valor de las entradas, en el segundo, en cambio, las salidas sólo dependen del estado actual. En el presente proyecto, todas las máquinas de estados siguen el modelo de Moore.

7.7. Comunicación mediante UART.

Una UART, de las siglas Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal), es un componente que sirve para controlar la comunicación a través del puerto serie, mediante el cual realizaremos la comunicación entre la Nexys4 DDR y el PC. Lo que hace este componente es convertir los datos de formato paralelo a serie para poder transmitirlo por el puerto serie.

Esta comunicación es posible ya que la Nexys4 DDR incluye el chip FTDI FT2232HQ, unido al conector J6, el cual sirve de puente para comunicarse con el puerto COM del PC. Este mismo integrado se emplea para controlar la circuitería USB-JTAG, pero sin que interfiera con la comunicación UART, de forma que permite que la placa sea programada, comunicada a través de una UART, y alimentada desde un único cable Micro USB.

La comunicación se realiza a través de un puerto serie de dos hilos, uno el TXD (transmisor) y otro el RXD (receptor) y da la opción de emplear 2 líneas de control de flujo (RTS y CTS). La conexión debe de realizarse de forma cruzada, es decir, el transmisor de uno de los dispositivos conectado al receptor del otro, y vicevercesa. La comunicación puede ser simple (sólo en una dirección), full dyplex (ambos dispositivos envían y reciben al mismo tiempo) o half duplex (los dispositivos se turnan para transmitir y recibir). Puesto que en este proyecto sólo es necesario enviar datos al PC, y no decibirlos, la comunicación a realizar será simple y utilizando únicamente una línea.

La tarea a realizar con la UART es separar los bits que componen cada byte y enviarlos de uno en uno, para lo que se suele emplear un registro de desplazamiento. En esta transmisión los datos se caracterizan por estar metidos en una trama, la cual está compuesta por un bit de inicio,luego los bits de datos, y 1 o más bits de parada. También se puede añadir un bit de paridad que irá tras los bits de datos, pero la trama empleada será la conocida como 8N1, es decir, un bit de start, 8bits de datos, ninguno de paridad y 1 bit de parada. El esquema de esta trama es el que se puede ver en la figura 7.7.0.14. Ahora, al contrario que en la comunicación I2C, se envía primero el bit menos significativo del byte.

Figura 7.7.0.14 – Trama de datos de la comunicación mediante UART [18]

7.8. Cálculo de la frecuencia cardíaca

Tras realizar la comunicación con el sensor, se tendrán guardadas las muestras que forman una señal pediódica con unos máximos cada cierto número de muestras.

Una vez se tiene esta señal, ya se puede proceder a la obtención de la frecuencia cardíaca. Una posible forma de calcularla sería obtener el número de muestras que hay entre dos máximos, y así consecutivamente, para realizar la media del número de muestras obtenido en cada caso. Pero este método tiene el inconveniente de que se podrían detectar falsos máximos, ya que la señal obtenida no es perfecta sino que contiene ruido.

Se ha optado por calcular la frecuencia cardíaca a partir de la función de autocorrelación de la señal obtenida del canal infrarrojo, según el procedimiento que se explica en [14].

Los pasos a seguir para calcular la frecuencia cardíaca son los siguientes:

1. **Restar la parte continua a la señal.** La señal obtenida del sensor tiene una parte continua, la cual conviene eliminar para así centrar la señal en el eje de abscisas y así facilitar los cálculos y operar con valores inferiores. Para realizar esto se sumarán todos los datos de la señal, se calculará la media y se restará esta a los datos.

2. **Centrar la señal en el eje de ordenadas.** Para facilitar los cálculos, se recomienda centrar la señal en el eje de ordenadas. Puesto que esto no se puede realizar en el ISE, ya que los vectores no pueden tener índices negativos, nos ahorraremos este paso, pero los cálculos se realizaran cómo si la señal estuviese centrada para facilitar las operaciones a realizar.

3. **Eliminar la pendiente.** Una vez se tiene la señal sin la parte continua, es decir, centrada en el eje de las abscisas, hay que nivelar la señal. La señal suele tener una tendencia, por lo que es conveniente eliminarla. Esto se realiza calculando la pendiente de la señal de la siguiente forma:

$$\beta = \frac{\sum_{t=-63,5}^{63,5} t \cdot y'_t}{\sum_{t=-63,5}^{63,5} t^2} = \frac{\sum_{t=-63,5}^{63,5} t \cdot y'_t}{174752} \tag{7.8.0.1}$$

   Siendo y'$_t$ el valor de la señal sin la parte continua.

   Para simplificar la operación de la división, se redondea la constante 174752 por el valor 131072 (2^{17}). Una vez obtenida la pendiente, se calcula la nueva señal restándole a la señal sin media la recta pendiente:

$$y_t = y'_t - \beta \cdot t \tag{7.8.0.2}$$

4. **Restar la media.** Nuevamente es necesario calcular la media de la señal sin tendencia y restársela para volver a tener la señal centrada en el eje de abscisas

5. **Cálculo de la función de autocorrelación.** La función de autocorrelación establece la relación existente entre la propia señal y esta desplazada k muestras. Esta función se calcula se la siguiente forma:

$$r_m = \sum_{t=1}^{n-m} y_t \cdot y_{t+m} \tag{7.8.0.3}$$

Siendo y_t la señal centrada y sin tendencia.

Esta función se suele emplear en señales con ruido para obtener, por ejemplo, la frecuencia fundamental de esta.

6. **Obtención del máximo.** Una vez se ha calculado la función de autocorrelación, el primer máximo local de esta función corresponde con el número de muestras (m) que equivalen a 1 período de la señal.

7. **Cálculo de la frecuencia cardíaca.** Tal y como se ha configurado el sensor, se obtienen las muestras con una frecuencia de 25Hz, por lo que la frecuencia cardíaca se obtendría dividiendo el punto donde se obtuvo el primer máximo local (m) entre 25. Si se desea obtener el resultado en pulsaciones por minuto, sólo hay que realizar la siguiente operación:

$$HR = \frac{60 \cdot 25}{m} \tag{7.8.0.4}$$

Es decir, obtener el periodo (inversa de m/25) y multiplicarlo por 60 para obtenerlo en minutos.

8 Resultados finales

Para realizar la comunicación entre el sensor y el PC, se ha elegido la Nexys 4 DDR, puesto que aunque en la asignatura de Sistemas Digitales I se empleaba la Nexys 2 DDR, el modelo 4 aporta ciertas ventajas frente a la 2, ya que tenemos una frecuencia del oscilador superior siendo en el caso de esta última de 100MHz, además de que dispone de más elementos integrados y mayor número de salidas como leds y switches, lo que nos facilitará las pruebas.

Otra ventaja por la que se ha elegido la Nexys 4 DDR, es que esta trae integrado el sensor de temperatura ADT7420, el cual utiliza el protocolo I2C para comunicarse, con lo que inicialmente se podrán realizar pruebas de comunicación con este sensor, puesto que es más sencillo que el MAX30102.

A la hora de realizar la comunicación con el sensor, se debe tener en cuenta el diagrama de tiempos que figura en su datasheet, puesto que es importante respetarlo para que la comunicación se realice correctamente.

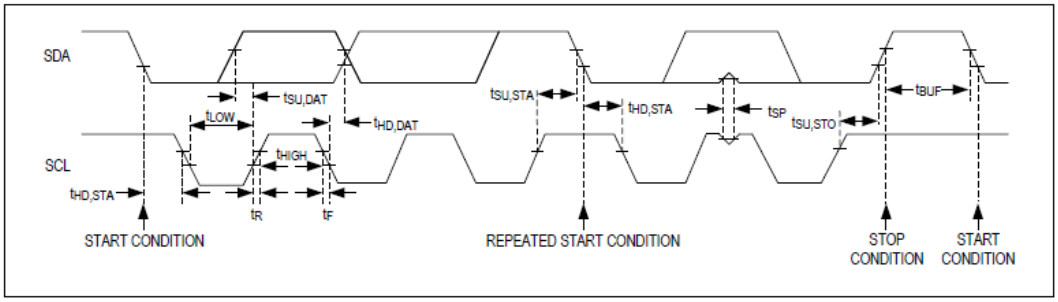


Figura 8.0.0.1 – Diagrama de tiempos en la comunicación I2C del sensor MAX30102 [10]

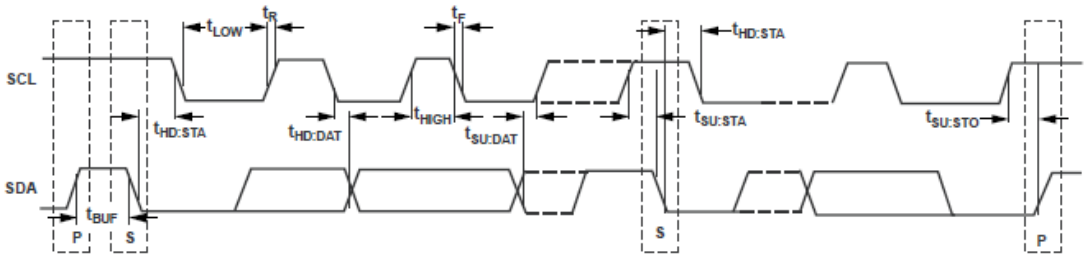


Figura 8.0.0.2 – Diagrama de tiempos en la comunicación I2C del sensor ADT7420 [10]

En el caso del MAX30102, el diagrama de tiempos tiene la forma que se ve en la figura 8.0.0.1 y para el sensor de temperatura ADT7420 la de la figura 8.0.0.2. En la tabla 8.0.0.1 se recoge una comparativa de los tiempos a respetar en cada uno de estos sensores.

	ADT7420		**MAX30102**		
	Min	**Max**	**Min**	**Max**	**Unidades**
Frecuencia SCL		400		400	kHz
$\mathbf{t}_{HIGH}$	0.6		0.6		μs
$\mathbf{t}_{LOW}$	1.3		1.3		μs
$\mathbf{t}_{R}$		0.3		0.3	μs
$\mathbf{t}_{F}$		0.3		0.3	μs
$\mathbf{t}_{HD:STA}$	0.6		0.6		μs
$\mathbf{t}_{SU:STA}$	0.6		0.6		μs
$\mathbf{t}_{SU:DAT}$	0.02		0.1		μs
$\mathbf{t}_{SU:STO}$	0.6		0.6		μs
$\mathbf{t}_{HD:DAT}$	0.03			0.9	μs
$\mathbf{t}_{BUF}$	1.3		1.3		μs

Tabla 8.0.0.1 – Tabla de especificaciones temporales del ADT7420 y el MAX30102.

Siendo:

- $\mathbf{t}_{HIGH}$**:** tiempo que debe estar la señal SCL a nivel bajo.
- $\mathbf{t}_{LOW}$**:** tiempo que debe estar la señal SCL a nivel alto.
- $\mathbf{t}_{R}$**:** tiempo de subida de SCL y de SDA.
- $\mathbf{t}_{F}$**:** tiempo de bajada de SCL y de SDA.
- $\mathbf{t}_{HD:STA}$**:** tiempo de mantenimiento de la operación de start, es decir, tiempo que debe estar SDA a nivel alto antes de que se produzca el flanco de baja de SCL para que se produzca el comienzo de la comunicación.
- $\mathbf{t}_{SU:STA}$**:** tiempo de establecimiento de la operación de start, es decir, tiempo que debe pasar desde el flanco de subida de SCL hasta que SDA se pone a nivel bajo generando un start.
- $\mathbf{t}_{SU:DAT}$**:** tiempo de establecimiento del dato en SDA, es decir, tiempo que debe estar el dato preparado en SDA antes de que se produzca el flanco de subida de SCL.
- $\mathbf{t}_{SU:STO}$**:** tiempo de establecimiento de la operación de stop, es decir, tiempo que debe estar SDA a nivel bajo desde que se produce el flanco de subida de SCL hasta que cambia a nivel alto para generar un stop.
- $\mathbf{t}_{HD:DAT}$**:** tiempo de mantenimiento del dato en SDA, es decir, tiempo que debe mantenerse el dato en SDA después del flanco de bajada de SCL.
- $\mathbf{t}_{BUF}$**:** tiempo que debe pasar entre una condición de stop y otra de start.

Se puede observar que las especificaciones temporales de ambos sensores son prácticamente iguales a excepción de $t_{SU:DAT}$, en cuyo caso se cogerá el valor más restrictivo, es decir, 0.1μs, y para $t_{HD:DAT}$ se respetará un valor intermedio entre ambos. De esta forma, si se programa la comunicación respetando estos valores, se podrán realizar pruebas en ambos sensores.

8.1. Esquema general

Tal y como se menciona en el apartado 7.6, la comunicación y obtención de la frecuencia cardíaca se lleva a cabo empleando máquinas de estado.

Se ha diseñado una máquina de estados para cada parte del proceso. Para tener una idea general de qué máquinas se activan desde dónde, y cual es la secuencia a seguir, se puede ver en el esquema 8.1.0.3 un gráfico general, en el que se ve como la máquina de estados del módulo ME_Repetidas_Acciones es quien activa y controla el resto de máquinas de estados.

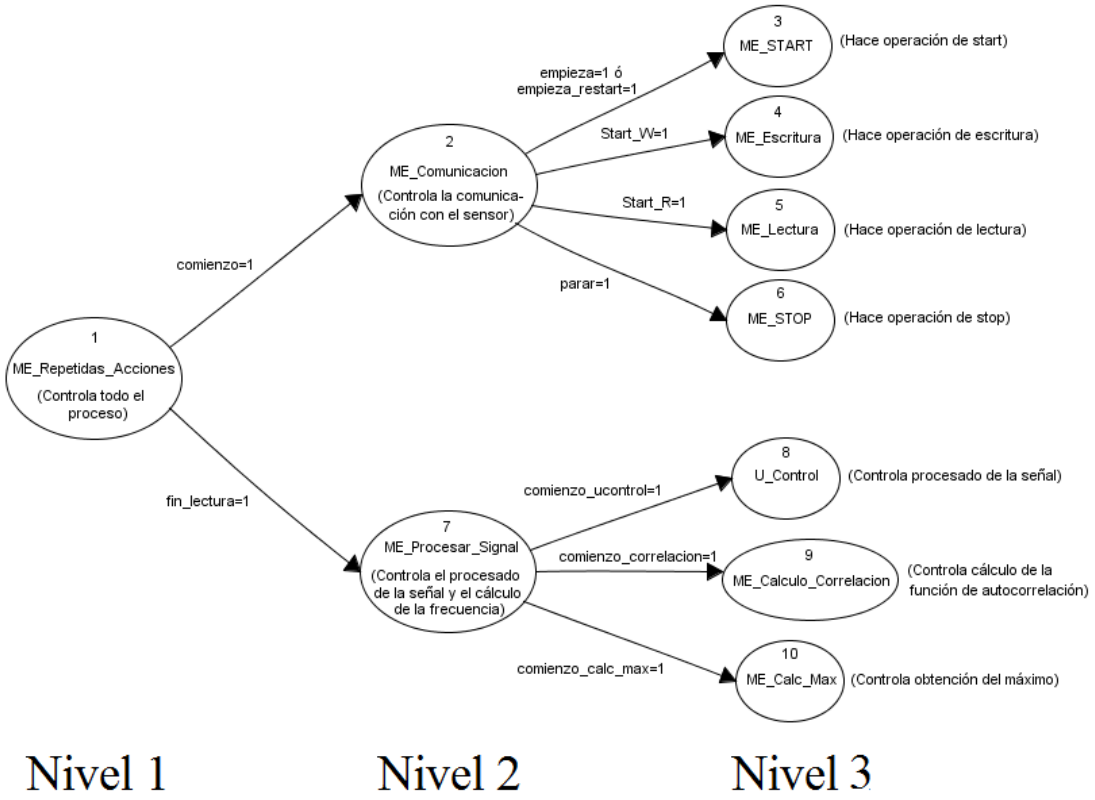


Figura 8.1.0.3 – Esquema general de unión de las máquinas de estados.

En ningún momento se ejecutan dos máquinas de estados del mismo nivel simultáneamente, sino que estas se ejecutan secuencialmente siguiendo el esquema de la figura 8.1.0.3, de forma que la siguiente máquina de estados no comienza hasta que finaliza la anterior

En los siguientes apartados se explicarán con más detalle estas máquinas y las operaciones que realizan.

8.2. Programación comunicación I2C

Tal y como se comentó anteriormente, la comunicación con el sensor comienza realizando una operación de start, seguida de otra serie de operaciones dependiendo de la acción que se quiera efectuar. A continuación, se va a detallar la programación realizada para llevar a cabo cada una de estas operaciones que en conjunto conforman la comunicación entre la Nexys y el sensor.

Las máquinas de estados presentadas a continuación y que tienen como objetivo controlar la comunicación y generar los datos de la línea SDA, tienen en común que todas aquellas en las que se generan valores a enviar por la línea SDA tienen como salida del módulo la señal *SDAo*, de forma que en un módulo superior se unificarán todas estas señales *SDAo* para dar un único valor a SDA. Por lo tanto, cuando al explicar estos módulos aparece la salida *SDAo* se hablará de darle valor a SDA, puesto que posteriormente SDA tomará el valor de estas señales.

8.2.1. Operación de START

Antes de comenzar la comunicación, tanto SDA como SCL permanecen a nivel alto a la espera de realizar alguna operación. Para comenzar, es necesario realizar la operación de start, la cual consiste en forzar SDA a nivel bajo mientras SCL está a nivel alto. Esto debe suceder al menos 0.6μs antes del flanco de bajada de SCL, es decir, hay que respetar el tiempo $t_{HD:STA}$ explicado antes. Esta acción está representada en la figura 7.6.0.10.

Para realizar esto, se ha programado un módulo en VHDL denominado como “ME_START” mediante el cual se ha implementado una máquina de estados finitos.

En la figura 8.2.1.1 se puede observar el diagrama de estados de esta máquina.

En este módulo están instanciados un contador que se habilita con la señal *ce_setup* y un detector de flanco de SCL.

En el primer estado *Inicio*, se espera a que se active alguna de las señales provenientes del módulo superior ME_Comunicación (*empieza* o *va_restart*) para comenzar la máquina de estados.

En el estado *E2* se espera un tiempo para respetar los 0.6μs de tiempo de mantenimiento del start tras el flanco de subida de SCL. Es en *E3* dónde se produce el start al pasar SDA a nivel bajo. Finalmente, en *E4* se activa a nivel alto la señal *START*, la cual indica que se ha finalizado esta operación.

En la figura 8.2.1.2 se puede ver la simulación funcional de este módulo, en la que se ve cómo al activarse la señal *empieza* se produce la operación de start y cómo se supera el tiempo mínimo de mantenimiento y de establecimiento. Se puede apreciar también que la señal *START* permanece a nivel alto sólo un ciclo de reloj de la placa .

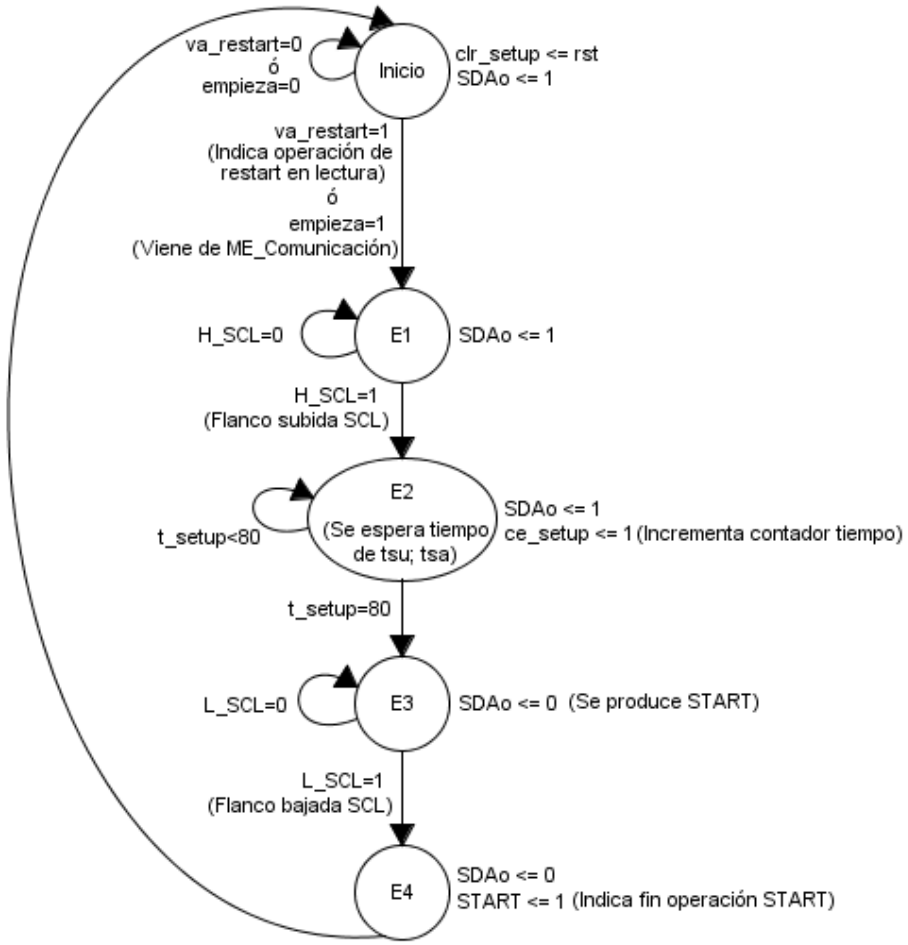


Figura 8.2.1.1 – Diagrama de estados de la máquina de START.

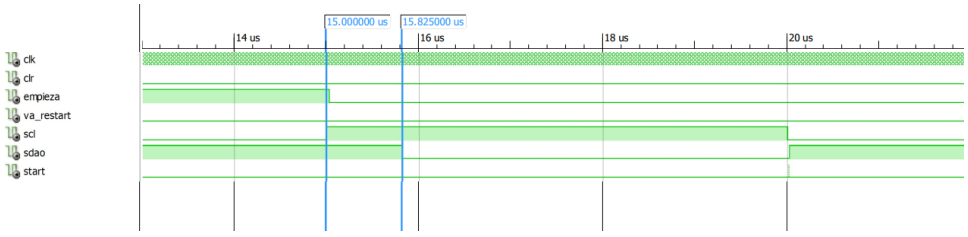


Figura 8.2.1.2 – Simulación funcional del módulo ME_START.

8.2.2. Operación de escritura

Tras realizar el start, es necesario enviar la dirección del esclavo con el que se quiere realizar la comunicación mediante una operación de escritura, proceso que se realiza en el módulo ME_ESCRITURA.

Esta acción debe realizarse tal y como se explica en el apartado 7.6 y se respetarán los tiempos de la tabla 8.0.0.1.

En la figura 8.2.2.2 se puede visualizar el diagrama de estados diseñado para realizar esta operación de escritura.

En el estado *Inicio* se espera a que la señal *START_W*, la cual vendrá de la máquina de estados ME_Comunicacion, se active a nivel alto. En este primer estado se activan las señales de clear de los distintos módulos implementados dentro de este. Estos módulos instanciados son dos contadores (uno del número de bits enviados y otro para esperar un tiempo), un registro de desplazamiento y un detector de flancos de SCL.

Una vez se pasa al estado *Desplazar* se activa el registro de desplazamiento, el cual ya se ha cargado con el dato en el estado anterior, y el contador de bits escritos. En el estado *WDATO* dónde se le da a SDA el valor del bit menos significativo de la señal de salida del registro de desplazamiento.

Desde el estado *Impedancia* a *Espera_Flanco* se realiza el ACK, poniendo SDA en alta impedancia para que el sensor sepa que puede escribir en la línea. Si el sensor envía un 1 durante el ACK, se pasa por el estado *ERROR* para indicar que se produjo mal la comunicación. En caso contrario, se pasa por el estado *FIN_ACK*. En el estado *FIN* se activa la señal *FIN_W* para indicarle a otros módulos que ya se finalizó la operación de escritura.

En la figura 8.2.2.1 se puede ver la simulación de este módulo en la que se escribiría el dato “10010110”, el cual corresponde a la dirección del sensor de temperatura integrado en la Nexys.

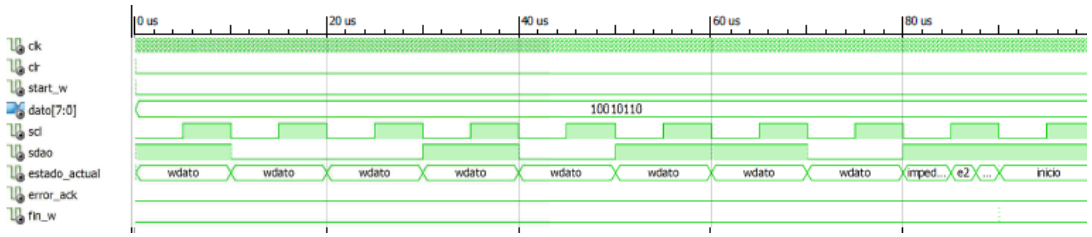


Figura 8.2.2.1 – Simulación funcional del módulo ME_ESCRITURA

8.2.3. Operación de lectura

Para realizar la operación de lectura, primero es necesario haber ejecutado un start, posteriormente haber escrito la dirección del esclavo y la dirección del registro a leer, volver a hacer una operación de start y reenviar la dirección del esclavo, tal y como aparece en la figura 7.6.0.13. Para la operación de lectura se ha creado el módulo ME_LECTURA, en el que se ha diseñado una máquina de estados con el diagrama de la figura 8.2.3.1.

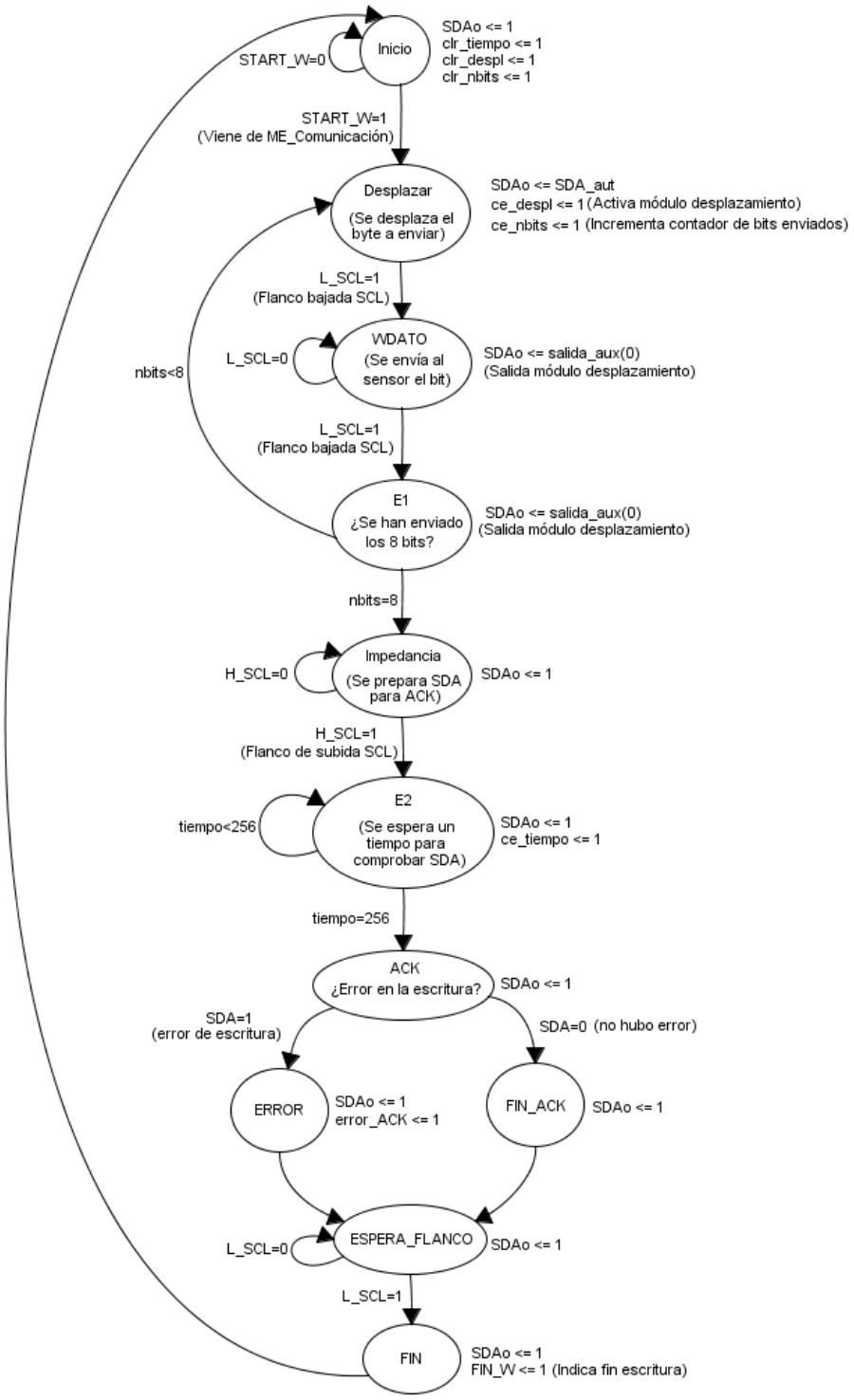


Figura 8.2.2.2 – Diagrama de estados de la operación de escritura.

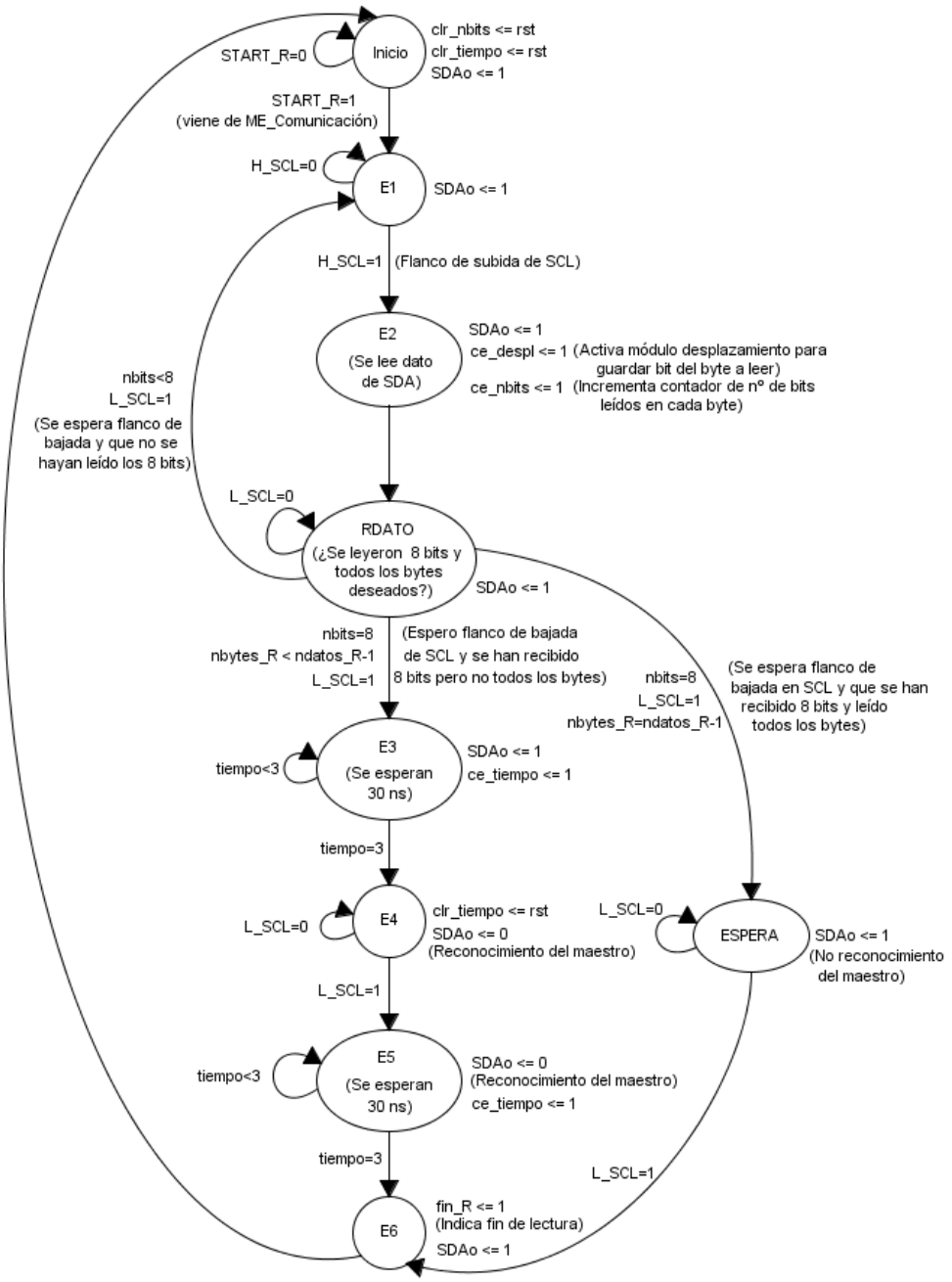


Figura 8.2.3.1 – Diagrama de estados de la operación de lectura.

Este módulo consta de 2 contadores (uno para el tiempo y otro para los bits leídos), un registro de desplazamiento y un detector de flancos de SCL.

En el estado *Inicio* se inicializan los contadores instanciados en este módulo. Se espera a que la variable *START_R* sea puesta a nivel alto por la máquina de estados ME_Comunicacion para comenzar la lectura.

Es en el estado *E2* en dónde se leer el valor de la línea SDA y se guarda en un registro de desplazamiento, de forma que el primer bit recibido sea el más significativo de un vector de 8 bits. En el estado *RDATO* se comprueba el número de bits leídos. En el caso de que se hayan leído los 8 bits, caben dos opciones: que no se hayan leído todos los bytes deseados, por lo que se pasará al estado *E3* para realizar un ACK, o que por el contrario se hayan leído todos los bytes deseados, pasando así al estado *ESPERA*.

La variable que indica los datos deseados es *ndatos_R*, y la variable *nbytes_R* indica los bytes leídos, es decir, el número de veces que se ejecutó la operación de lectura completa

En los estados anteriores, la línea SDA se mantuvo a nivel alto, puesto que el maestro no necesitaba mandar ningún dato, sino recibirlos del esclavo. Por el contrario, en *E4* se va a realizar el ACK, así que de forma similar al realizado por el esclavo, ahora es el maestro el que pone la línea SDA a nivel bajo durante el noveno ciclo de SCL.

Finalmente en *E6* se activa la señal *fin_R* para indicar que se terminó la operación de lectura.

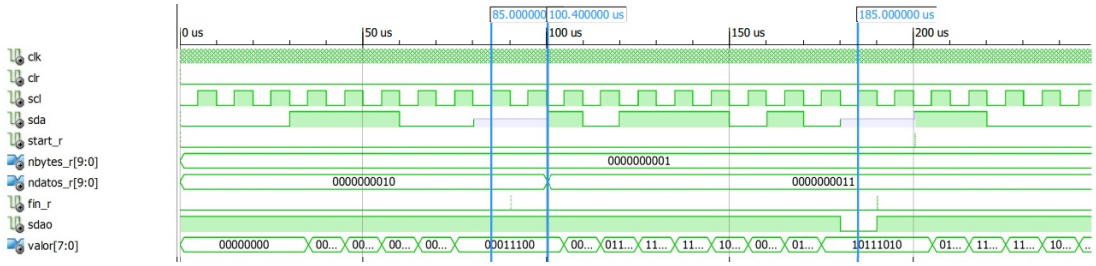


Figura 8.2.3.2 – Simulación funcional 1 del módulo ME_LECTURA.

En la figura 8.2.3.2 se puede ver un primer ejemplo de simulación de este módulo en el que se quiere leer 2 bytes (*ndatos_R=2*) pero ya se ha leído uno. Se puede ver como SDAo está a nivel alto todo el tiempo hasta leer los 8 bits (primer marcador) y permanece así en el noveno ciclo de SCL para realizar un NACK.

En el segundo marcador de la figura 8.2.3.2 se puede ver un segundo ejemplo en el que ahora se quiere leer 3 bytes. El primer dato comienza en este segundo marcador, y se ve como se va guardando en la señal *VALOR* en cada periodo de SCL. En el noveno ciclo de SCL (tercer marcador), se ve como ahora *SDAo* toma el valor 0 para indicarle al sensor el ACK, y tras activarse *START_R* se realiza la lectura del segundo dato.

8.2.4. Operación de STOP.

Por último, tras realizar una operación de escritura o de lectura, si no se desea continuar la comunicación es necesario ejecutar una operación de stop. Esto se hace mediante una

máquina de estados implementada en el módulo ME_STOP. Esta máquina sigue el diagrama de la figura 8.2.4.1.

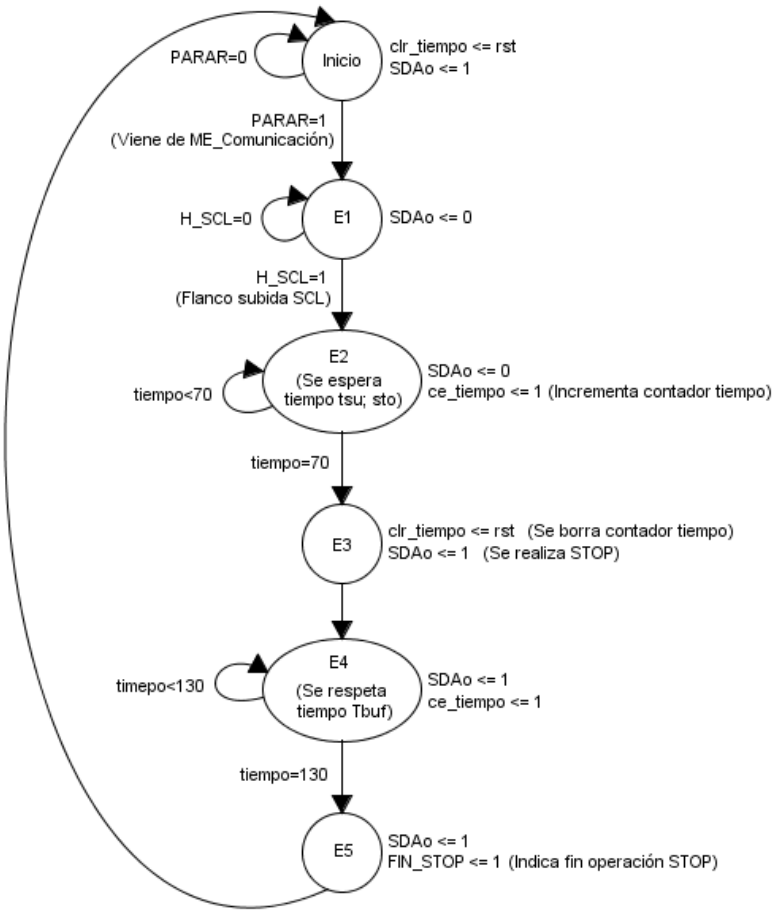


Figura 8.2.4.1 – Diagrama de estados de la operación de stop.

En este módulo se instancian un contador para esperar 2 tiempos, y un detector de flancos de SCL.

Se pasa al estado *E1* cuando la máquina de estados ME_comunicación pone a nivel alto la señal *PARAR* indicando que debe comenzar esta máquina de estados.

La línea SDA permanece a nivel bajo hasta que pasa un tiempo tras el flanco de subida de SCL, momento en el que SDA pasa a nivel alto y se produce la operación de stop (estado *E3*)

En *E4* se espera 1.3μs para respetar el tiempo de buffer. En cuanto se cumple ese tiempo, se pasa al estado *E5*, donde se activa la señal *FIN_STOP* a nivel alto para indicar que se finalizó la operación de stop, y se vuelve al estado *Inicio*.

En la figura 8.2.4.2 se puede ver la simulación funcional de este bloque, en la que se aprecia cómo tras activarse la señal *PARAR*, SDA pasa de nivel bajo a nivel alto tras el primer flanco de subida de SCL, respetando los 0.6μs de tiempo de establecimiento mínimo, y manteniéndose luego SDA a nivel alto.

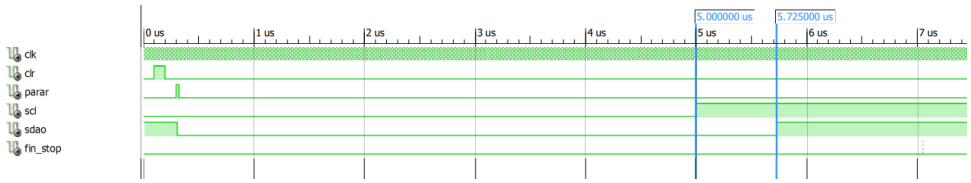


Figura 8.2.4.2 – Simulación funcional del módulo ME_STOP.

8.2.5. Máquina de estados ME_Comunicacion.

Una vez listas las máquinas de estados que generan cada una de las distintas operaciones que se pueden realizar al comunicarse mediante protocolo I2C con el esclavo, es necesario diseñar una máquina general que unifique las restantes para que se ejecuten siguiendo la secuencia correcta. Con este propósito se creó el módulo ME_Comunicación, en el cual se implementó una máquina de estados desde la que se generan las señales que activarán las anteriores máquinas de estados presentadas, de forma que se pueda realizar una comunicación general con cualquier esclavo mediante protocolo I2C. El diagrama de esados es el de la figura 8.2.5.1

En esta máquina de estados se activan las anteriores para realizar la secuencia a seguir en la comunicación I2C.

En el estado *ESCRIBIR* se realiza la operación de escritura. La primera vez que se pasa por este estado, la señal *DATO* tendrá el valor de la dirección del esclavo (señal *DIR*) más el bit de escritura 0. Dependiendo del número de escrituras realizadas, la señal *DATO* tendrá el valor de la dirección del esclavo más el bit de escritura o de lectura, la dirección del registro o el dato que se quiera escribir. La señal *error_ack_aux* indicará si que produjo un error en la comunicación, que de ser así, se borrará esta señal y se volverá a realizar la escritura.

En *Comparo* se comprueba el valor de la variable *nbyte_W*, la cual es la salida del contador de número de veces que se ha realizado una operación de escritura. Dependiendo del valor de esta variable y de la variable *RW*,indicando esta última si se quiere realizar una escritura o una lectura, se puede saber en qué punto de la comunicación se está y cual es la siguiente operación a realizar.

- **Si nbyte_W<2**: quiere decir que sólo se ha realizado la escritura de la dirección del esclavo y por lo tanto falta escribir la dirección del registro, con lo que es necesario volver al estado *ESCRIBIR*.

- **Si nbyte_W=2 y RW=1**: se ha realizado la escritura de la dirección del esclavo y del registro. Al estar *RW* a nivel alto indica que se quiere leer, por lo que en este punto es necesario hacer un restart, para lo que se vuelve al estado *GO*.

- **Si nbyte_W>2 y RW=1**: si se ha realizado la escritura de la dirección del esclavo y del registro, el restart y de nuevo la escritura de la dirección del esclavo, es decir, 3 escrituras, por esto *nbyte_W>2* y al querer realizar una lectura (RW=1), se pasa al estado *LEER* para obtener los datos del sensor.

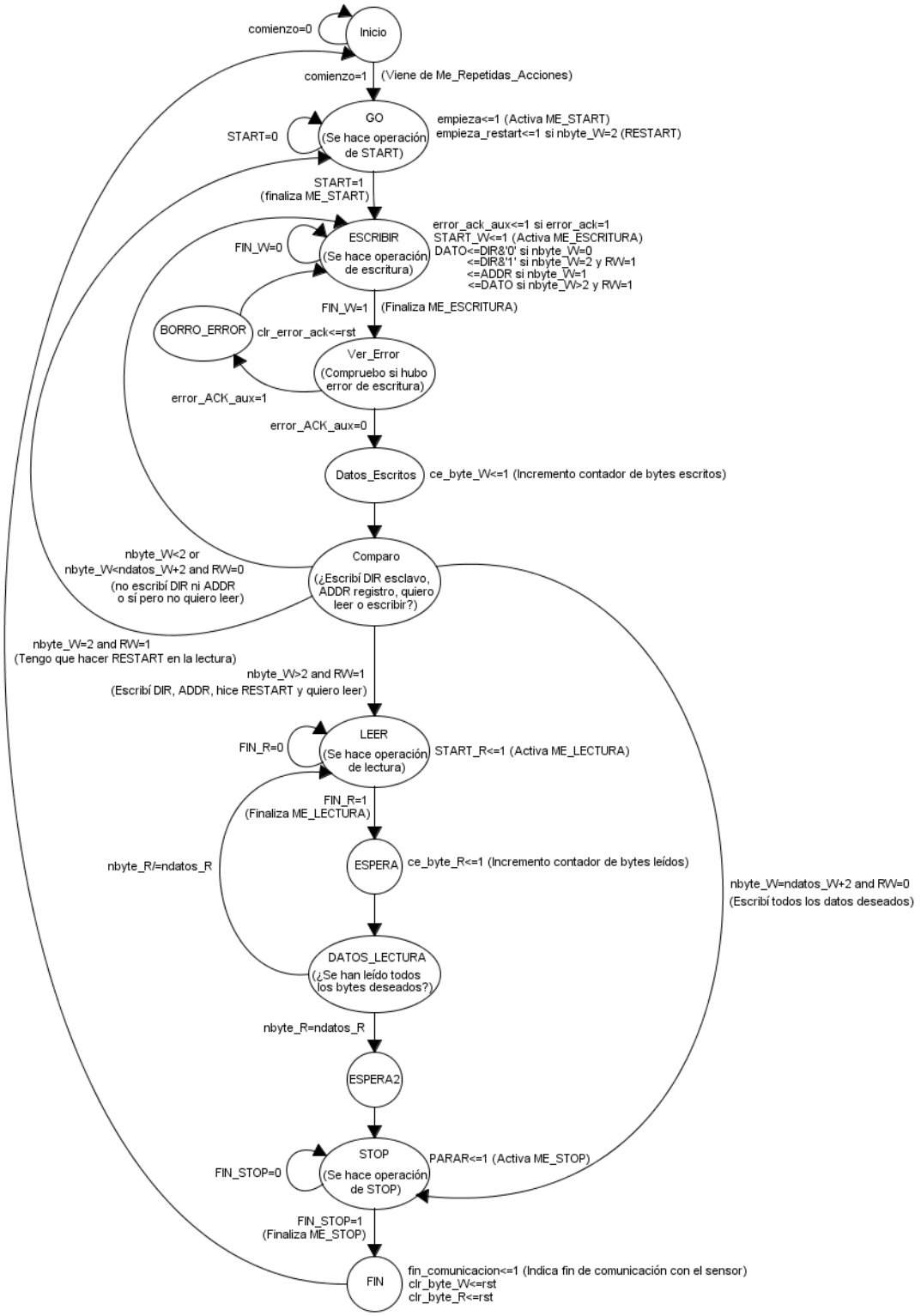


Figura 8.2.5.1 – Diagrama de estados del módulo ME_Comunicación.

- **Si nbyte_W<ndatos_W+2 y RW=0**: en el caso de querer escribir, se vuelve al estado *ESCRIBIR* para realizar la escritura de todos los datos que se desee una vez ya se han escrito la dirección del esclavo y del registro (por eso *ndatos_W+2*, es decir, el número de datos que se quiera escribir más las 2 operaciones de escritura de las direcciones).

- **Si nbyte_W=ndatos_W+2 y RW=0**: se pasa al estado *STOP* cuando, al querer escribir, ya se han escrito todos los datos.

Al pasar al estado *LEER*, se realizarán el número de operaciones de lectura que se deseen, y finalmente se realizará la operación de stop.

Por último, en el estado *FIN* se activa la señal *fin_comunicación* para indicar que se finaliza la comunicación con el esclavo. En este estado las señales *clr_byte_W* y *clr_byte_R* se activan para borrar los contadores.

8.2.6. Módulo TOP_Comunicacion

Una vez creadas todas las máquinas de estados necesarias para realizar la comunicación I2C, es necesario realizar las conexiones entre ellas. Para ello se crea el módulo TOP_Comunicación, en el que de instanciarán los módulos comentados hasta el momento, es decir, ME_START, ME_Escritura, ME_Lectura, ME_STOP y ME_Comunicacion.

Además de las anteriores máquinas de estados, se instancia el módulo Divisor_SCL en el que se genera la señal de reloj SCL que sincroniza la comunicación entre el maestro y el esclavo.

En el anexo 10 se puede ver la conexión de los 6 módulos instanciados en TOP_Comunicación.

Todas las máquinas de estados descritas tienen una salida denominada SDAo, la cual determina el valor que debe tomar SDA en cada momento de esa máquina de estados. Ahora es necesario unificar estos valores en una sola variable, puesto que sólo tenemos la línea inout SDA. Como ninguna de las máquinas de estados se solapan, lo que se hace es simplemente decirle a la línea SDA que tome el valor 0 cuando alguna de las líneas SDAo de los distintos módulos toma el valor 0 y sino que esté en alta impedancia. No es necesario ponerla a nivel alto, ya que tanto la propia Nexys para el sensor de temperatura como la placa MAXREFDES117 incluyen la resistencia de pull-up necesaria para interpretar este tercer estado. Se hace lo mismo con la línea SCL.

8.3. Prueba comunicación con el sensor MAX30102

El sensor MAX30102 es más completo que el de temperatura empleado para realizar pruebas, puesto que tiene muchos más registros para configurar y modos de funcionamiento.

Primero se va a realizar la prueba de escribir en un registro y luego leerlo para comprobar que se está realizando la comunicación con el sensor correctamente, tanto al escribir como al leer un dato.

Para esta primera prueba, por ejemplo, se va a escribir en el registro de configuración 0x08 el dato 0xA3. Se van a realizar dos operaciones, primero la escritura del registro y luego la lectura de este, por lo que se ejecutará dos veces la comunicación, te tal forma que en la máquina de estados ME_Repetidas_Acciones únicamente se pasará por el estado *Escritura* y por el estado *Lectura_S* una vez.

Es necesario realizar la asignación de SDA y SCL en el fichero UCF con los pines correspondientes del puerto PMOD y asignar a la señal correspondiente la dirección de este sensor (“1010111”).

Se implementa el proyecto en la placa y se puede ver en la figura 8.3.0.1 el dato 0xA3 (“10100011” en binario) en los leds de la placa.

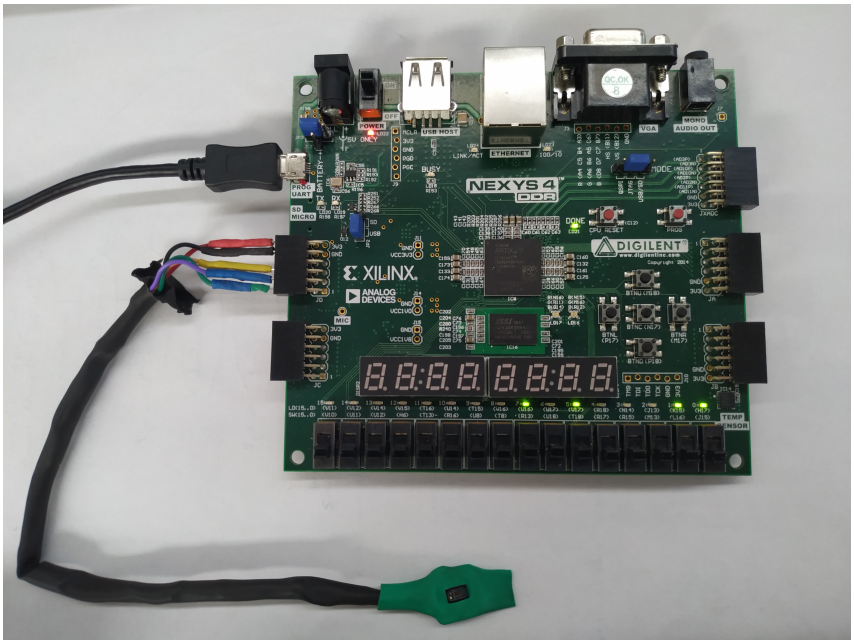


Figura 8.3.0.1 – Lectura del registro 0x08 del sensor MAX30102.

Tras haber logrado realizar con éxito la anterior prueba, se va describir a continuación el procedimiento necesario para realizar la lectura de las muestras:

1. Realizar las 12 operaciones de escritura para configurar el sensor tal y como se explica en el apartado 7.3.

2. Leer el registro de interrupción 0x00 para determinar cuándo está llena la FIFO .

3. Cuando la FIFO está llena, se realiza la lectura de esta.

4. Volver a leer el registro 0x00 para comprobar que se ha borrado la interrupción. Luego se espera a que esta se active nuevamente para volver a leer la FIFO.

Inicialmente, se van a realizar 10 lecturas de la FIFO para comprobar que los datos son correctos.

Para realizar la secuencia de pasos descrita, es necesario crear otra máquina de estados de forma que active la máquina ME_Comunicacion las veces necesarias. Con este propósito se crea el módulo ME_Repetidas_Acciones.

8.3.1. Máquina de estados ME_Repetidas_Acciones para prueba de lectura.

En este módulo se genera la máquina de estados representada en las figuras 8.3.1.1 y 8.3.1.2.

En el estado *Inicio* se espera a que se pulse el botón de comienzo en la placa, momento en el que se pasa al estado *Empieza* para dar comienzo a la comunicación.

Primero, se realizarán las 12 escrituras en el sensor para configurarlo, dándole los valores de la dirección del registro y el dato a escribir correctos para cada escritura.

Una vez configurado el sensor, se desea comprobar si se ha activado la interrupción de FIFO llena, por lo que se pasa al estado *Empieza1* para volver a comenzar la comunicación y pasar al estado *Lectura_INT*, en el que ahora *RW* vale 1 ya que se quiere realizar una lectura, y *ADDR* vale 0x00 puesto que es la dirección del registro a leer. Luego se comprueba el valor del bit 7 del dato leído. Este bit corresponde con la interrupción A_FULL, por lo que si está a nivel bajo quiere decir que la FIFO está llena y se puede pasar al siguiente estado.

Ahora, se va realizar la lectura de la FIFO, por lo que se van a leer los 32 datos guardados en ella. Se leen los 192 bytes (32 muestras x 3 bytes x 2 canales), almacenando cada vez que se lee un byte en una posición de una memoria RAM. En este caso sólo se quiere realizar una única operación de lectura en la que se leerán los 192 bytes.

En *Empieza3*, *Lectura_INT2* y *Comparo_INT2* se realiza el mismo proceso que antes para leer el registro de interrupción 0x00. Ahora esta interrupción debe estar desactivada, y por lo tanto el bit más significativo debe valer 1. Si es así se pasa al estado *E1*, sino se leerá el registro hasta que se haya borrado la interrupción.

En *E1* se incrementa un contador que indica el número de veces que se ha leído la FIFO. Inicialmente se va a leer esta 10 veces. Si aun no se ha completado el número de lecturas deseadas, se vuelve al estado *Empieza1* para esperar a que se llene la FIFO y volver a leerla nuevamente.

En el estado *FIN* se borran los contadores y se pasa al estado *Inicio* en el momento en que la variable *fin_RAM* vale 1.

Como se ha mencionado, se han instanciado 4 contadores en este módulo para contar el número de operaciones de lectura, de escritura, el número de lecturas de la FIFO y un tiempo de espera.

La señal *lee_32*, que se activa a nivel alto en el estado *Lectura_S*, sirve para indicar que se quiere leer la FIFO y por lo tanto es necesario leer 192 bytes. Cuando esta señal vale 0, se leerá únicamente un byte (lectura del registro de interrupción).

Puesto que ahora se van a guardar un total de 320 muestras en una RAM, lo que equivale a 1920 bytes, aunque se visualicen estos valores en el display o en los leds de la placa no se va a poder ver de forma clara si se está leyendo correctamente, por lo que es necesario enviar

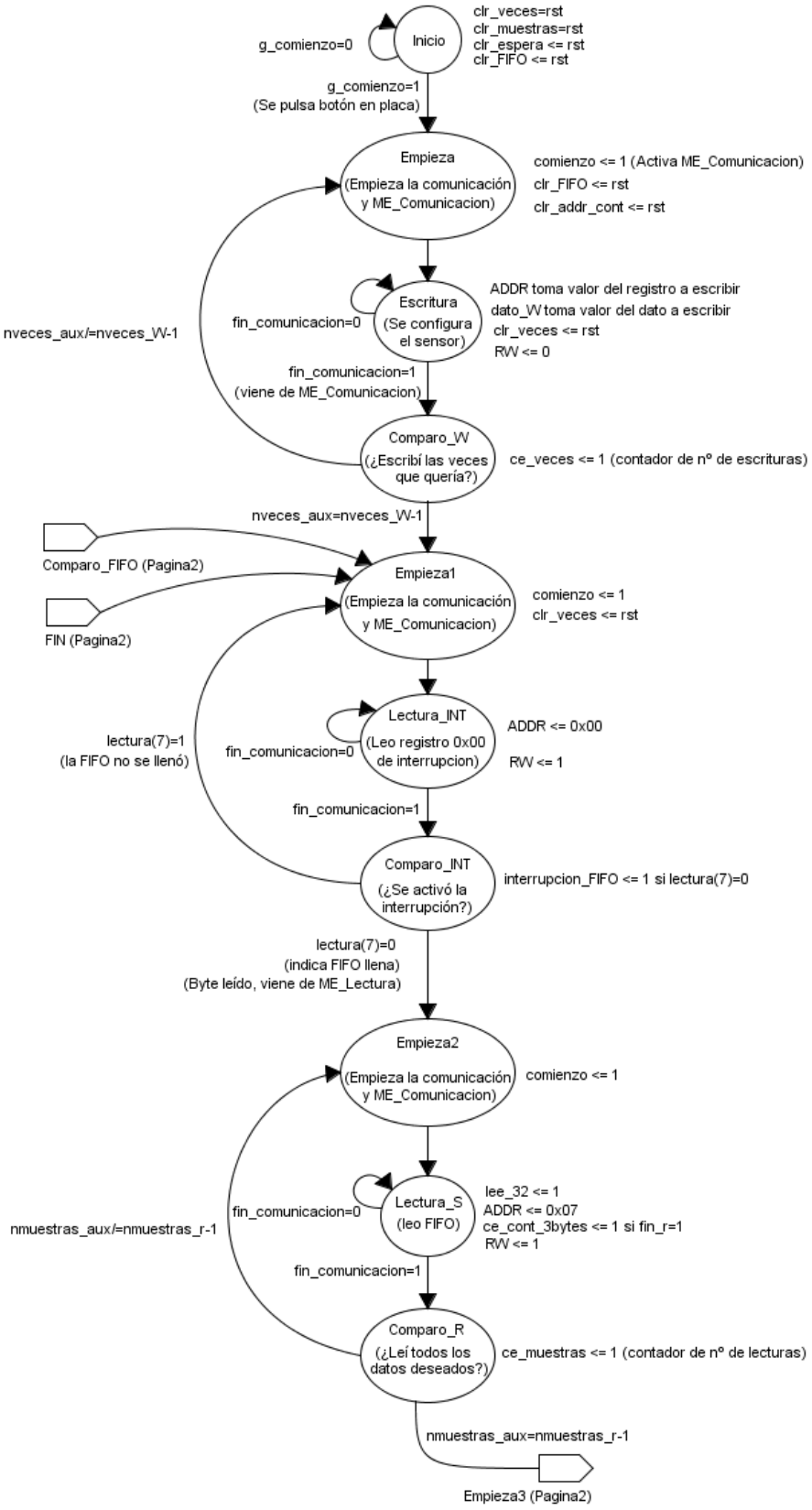


Figura 8.3.1.1 – Parte 1 del diagrama de estados de ME_Repetidas_Acciones para prueba de lectura.

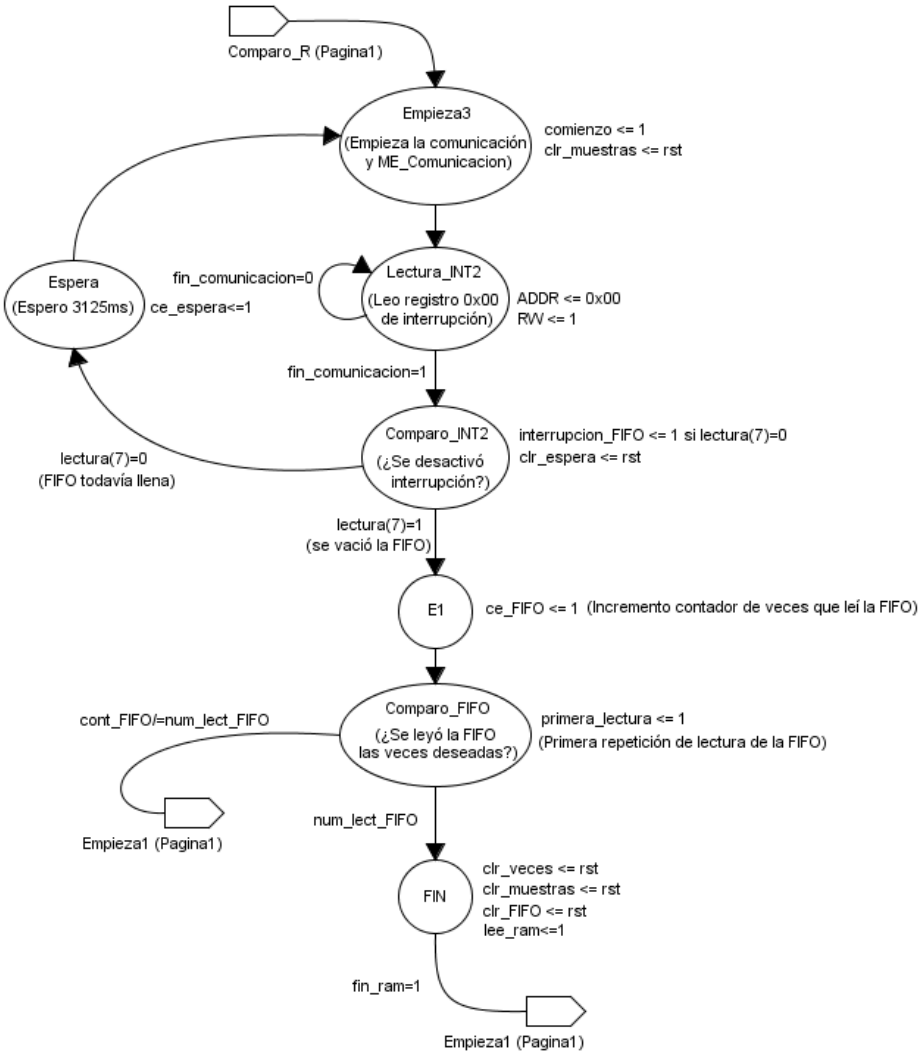


Figura 8.3.1.2 – Parte 2 del diagrama de estados de ME_Repetidas_Acciones para prueba de lectura.

estos datos al ordenador para poder visualizarlos en él de una forma más sencilla y gráfica, y así obtener la información necesaria.

Para esto, es necesario comunicarse con el PC mediante una UART. La señal de salida *lee_ram* que se activa a nivel alto en el estado *FIN* hace que comience esta comunicación. La señal *fin_ram* empleada como condición entre el estado *FIN* e *Inicio* viene de otro módulo e indica que se ha finalizado el envío de datos al PC.

8.3.2. Módulo Memoria_RAM

Para poder guardar los datos leídos del sensor, que como se ha mencionado antes serán 1920 bytes, se crea una memoria RAM en el módulo denominado como Memoria_Ram.

La RAM tendrá 2048 direcciones para poder almacenar todos los bytes. Los datos que se almacenarán serán de 8 bits.

Esta memoria se instancia en el módulo SUPER_TOP. Su señal de habilitación de escritura (*WE*) se activará cuando se finalice la lectura de cada byte leído de la FIFO.

8.3.3. Módulo Contador

Contador del módulo SUPER_TOP que genera la dirección de la RAM antes mencionada para guardar los datos leídos del sensor.

Este contador se activa con la misma señal de habilitación de escritura de la RAM.

8.4. Comunicación con el PC

Para poder obtener información de los datos leídos del sensor, es conveniente enviarlos al ordenador para poder trabajar con estos datos de forma más sencilla. La comunicación con el PC se realizará únicamente a través de la línea TX, es decir, será una comunicación simple, puesto que sólo nos interesa enviar datos al ordenador, no recibirlos. Para esto se creó el módulo TOP_UART_TX.

Dentro de este módulo hay otros 4 módulos instanciados que se explicarán a continuación. Se puede ver su esquema de conexionado en el anexo 10.

Dentro de este módulo, se aplica una máscara al byte recibido de la RAM, de forma que sólo se guardan los dos bits menos significativos si el dato recibido es el primero de una muestra, y se guardará todo el dato si es el segundo o tercer byte de la muestra. Es necesario hacer esto ya que como se explicó antes, al seleccionar una resolución del AD de 18 bits, los primeros 6 MSB del primer byte de cada muestra no aportan información.

8.4.1. Módulo EdgeDetector

Este módulo consiste en un detector de flanco de subida de la señal de entrada *x_0*. Sirve para detectar el momento en el que se ha activado la señal que comienza la comunicación mediante la UART. La salida de este módulo se activa únicamente durante un ciclo de reloj y hace que comience la máquina de estados del módulo ME_ComMatlab.

8.4.2. Módulo ME_ComMatlab

En este módulo se implementa la máquina de estados de la figura 8.4.2.1 para controlar la comunicación con el PC.

En el momento en que se activa la señal que da comienzo a la máquina de estados, se pasa al estado *transmitir*, en el que se activa la salida *Transmite* para que comience la transmisión de un dato al el PC. Acto seguido se pasa al estado *TX_Ocupado*, en el que se espera a que se termine de enviar el byte. En cuanto se ha enviado el byte (*TX_ocupado=0*) se pasa al estado *Incr_Dir*, dónde se activa la señal de habilitación del módulo Contador_ascendente, para que aumente la dirección de la RAM y así pasar a enviar el siguiente dato. En el siguiente flanco de reloj se pasa al estado *Comprueba_DIR*, en el que se comprueba si la dirección de la RAM ha llegado a 0, lo que quiere decir que se han enviado todos los datos, por lo que se pasa al estado *FIN* en el que se activa la señal *Fin_matlab*. En el caso de que la dirección sea mayor que 0, quiere decir que aun no se han enviado todos los datos y se vuelve al estado *Transmitir* para comenzar una nueva transmisión.

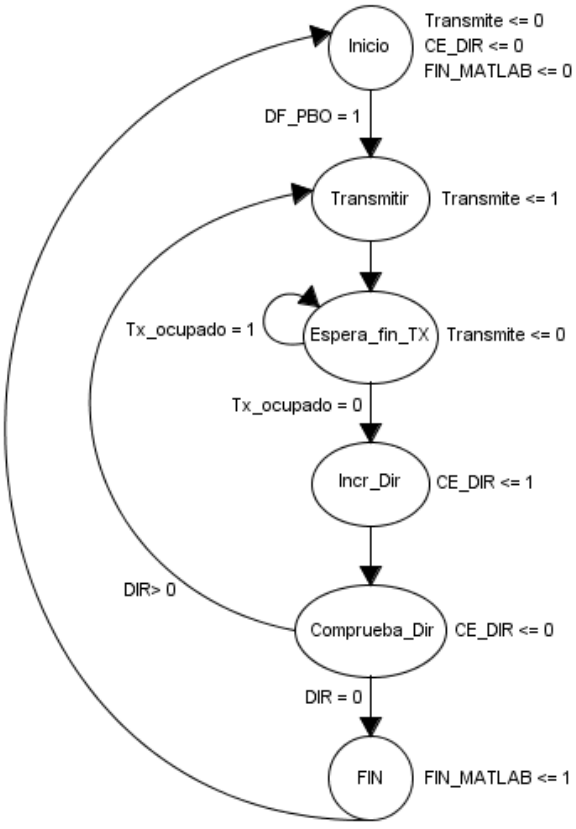


Figura 8.4.2.1 – Diagrama de estados del módulo ME_ComMatlab.

8.4.3. Módulo UART_TX

Puesto que no se necesita recibir ningún dato del ordenador, la UART diseñada emplea únicamente la línea Tx para transmitir los datos desde la FPGA al PC. La tarea a realizar es separar los bits que componen cada byte y enviarlos de uno en uno. Ahora, al contrario que en la comunicación I2C, se envía primero el LSB del byte.

Este módulo consta de otros 4 para realizar exitosamente la comunicación:

1. **Divisor:** la tasa de transferencia de datos mediante el puerto serie se mide en baudios. Los baudios determinan la cantidad de bits enviados en un segundo. Para realizar la comunicación se emplea una tasa de 115200 baudios, para lo que se crea un divisor de frecuencia.

   Si se dividide 1/115200Hz se obtiene que un dato se transfiere en 8.68x10^{-6}s, es decir, que se debe transferir un bit cada 8.68μs. Puesto que la frecuencia de CLK es de 100MHz, si dividimos 100Mhz entre 115200Hz, se obtiene que el contador del divisor debe llegar hasta 868. La señal de reloj de salida *baud* va a estar a nivel alto únicamente un periodo de CLK. La señal de entrada *CE* habilita el contador, y será activada desde el módulo U_Control_TX.

2. **Registro_desplazamiento:** en este módulo se carga mediante la señal *LOAD* el dato que se quiere enviar al PC, y cada vez que se activa la señal de habilitación se desplazan los bits hacia la derecha. Por la salida se saca el bit menos significativo, de forma que el más significativo es el último en salir.

3. **Mux:** se emplea un multiplexor que selecciona entre 3 entradas, siendo la salida de este módulo el dato que se enviará por la línea Tx. Se elige entre tres entradas, siendo *A* el bit de stop de la comunicación (valor 1), *B* el bit correspondiente del dato a enviar y *C* el bit de comienzo de comunicación (valor 0).

4. **U_Control_Tx:** en este módulo se implementa la máquina de estados que controla la transmisión mediante la UART. La máquina de estados diseñada tiene el diagrama que se puede ver en la figura 8.4.3.1.

   Se espera en el estado *Einicial* a que se active la entrada *Transmite* que da comienzo a la comunicación con el PC.

   En el estado *E_Bit_inicial* la señal de salida *SEL* toma el valor “10” para que se envíe el bit 0 de inicio de comunicación. Se espera a que se active a nivel alto la señal *baud* que indica que han pasado los 8.68μs y se puede enviar el dato siguiente. Del estado *E_bit_dato0* al estado *E_bit_dato7* la salida *sel* toma el valor “01” para que la salida del multiplexor sea el bit correspondiente a enviar.

   En el estado final *E_bit_final* la señal *sel* toma el valor “00” para que la salida del multiplexor sea el bit de parada, es decir, un 1, y así finalizar el envío de 1 byte.

   Se vuelve al estado *Einicial* para a esperar a que se active la señal *transmite* nuevamente y repetir todo el proceso hasta que se envíen todos los datos de la memoria RAM.

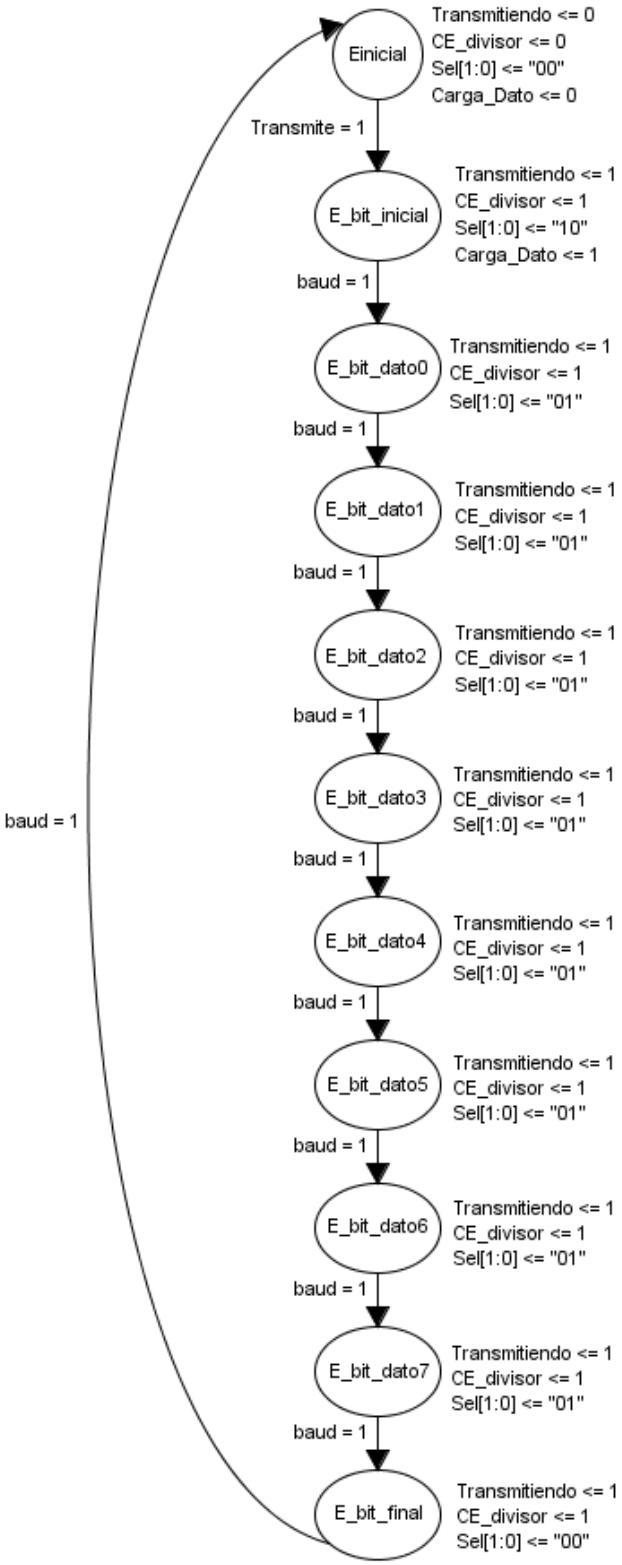


Figura 8.4.3.1 – Diagrama de estados del módulo U_Control_Tx

8.5. Recepción de datos en el PC.

Para recibir los datos enviados desde la FPGA al PC se ha elegido el entorno Matlab, ya que dispone de numerosas herramientas que nos permitirán realizar pruebas con los datos obtenidos del sensor.

8.5.1. Script en Matlab para recepción de datos.

El script desarrollado para la recepción de los datos es el que se puede ver en el código 8.1.

Código 8.1: Script para recibir datos por el puerto serie en Matlab.

```
%%MODIFICAR SI ES NECESARIO EL PUERTO DE CONEXIÓN%%
Port='COM4';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FPGA_COM=serial(Port,'BaudRate',115200); %%velocidad de 115200 baudios
set(FPGA_COM,'StopBits',1);  %%un bit de parada
set(FPGA_COM,'InputBufferSize',2048); %%número de datos del buffer
set(FPGA_COM,'Timeout',60); %%tiempo de espera del puerto
fopen(FPGA_COM);  %%se abre el puerto
out = fread(FPGA_COM,2048); %%se deben leer 2048 datos de 8 bits cada uno
FPGA_COM.ValuesReceived  %%devuelve el número de datos recibidos

%%Inicialización de señales para unir los bytes%%
i=0;
cont=0;
led_R=[];
led_IR=[];
xR=[];
xIR=[];
R=0;
IR=0;

while i<=2047
i=i+1;
cont=cont+1;
%%Tres bytes que conforman una muestra del led rojo%%
if cont==3
    R=R+1;
    led_R(R)=bin2dec(strcat(dec2bin(out(i-2),8),dec2bin(out(i-1),8),dec2bin(out(i),8)));
    xR(R)=R;
end
    %%Tres bytes que conforman una muestra del led IR%%
if cont==6
    IR=IR+1;
    led_IR(IR)=bin2dec(strcat(dec2bin(out(i-2),8),dec2bin(out(i-1),8),dec2bin(out(i),8)));
```

```
    xIR(IR)=IR;
    cont=0;
end
end

%%Se grafican y de gradan los datos%%
subplot(3,1,1);
plot (xR,led_R,'r')
subplot(3,1,2);
plot (xIR,led_IR,'b')
filename = 'datos.xlsx';
savefig ('datos10.fig')
xlswrite(filename,led_R,'A1')
%%Se cierra el puerto%%
fclose(FPGA_COM);
```

Lo primero que se hace es determinar el puerto a emplear, que dependiendo del ordenador puede variar, de forma que lo primero que se debe hacer es comprobar en el administrador de dispositivos del PC a qué puerto COM aparece conectada la Nexys.

A continuación se configuran las características de este puerto, estableciéndose la velocidad de transferencia en 11500 baudios, se especifica que el bit de stop, el número de datos a recibir que será de 2048 (tamaño de la memoría RAM empleada en el ISE) y un tiempo de espera de 60s, es decir, el puerto esperará un minuto a recibir los datos, una vez pasado este tiempo si no los ha recibido la comunicación será insatisfactoria.

Luego se abre el puerto y se lee. Si no se especifica, el tamaño de los datos a recibir por defecto es de 8 bits. Puesto que en Matlab existe la limitación de que los tamaños de los datos a recibir es limitado, se enviará byte a byte y luego se unirán los 3 bytes que conforman una muestra en este Script.

Tras recibir los datos, lo que se hace es unir los 3 primeros bytes como una cadena de caracteres, y luego se pasa a decimal para guardar la muestra del led rojo ya completa. Se realiza lo mismo con los 3 datos siguientes, que serán los que compongan una muestra del led infrarrojo, y se continuará alternando de 3 en 3 hasta el final.

Finalmente se grafican estos datos para poder visualizarlos y se guardan en un Excel.

8.5.2. Envío de datos desde la FPGA a Matlab.

Ahora en el módulo SUPER_TOP además del módulo TOP_Comunicacion, se han instanciado los módulos ME_Repetidas_Acciones, Memoria_Ram, el contador Dir_W_RAM y TOP_UART_TX.

Se realizan las conexiones entre estos módulos. Además de las señales necesarias para realizar estas conexiones, existen estas 4 señales:

- **DIR_AUX(6:0)** se conecta a la entrada *DIR* del módulo TOP_Comunicación. Esta señal tiene siempre el valor “1010111” correspondiente a la dirección del esclavo.
- **N_DATOSR_AUX(9:0)** se conecta a la entrada *N_DATOSR* del módulo TOP_Comunicación.

Esta señal, que indica el número de bytes a leer, valdrá 192 cuando se quiera leer la FIFO del sensor, y valdrá 1 cuando se vaya a leer el registro de interrupción.

- **N_DATOSW_AUX(7:0)** se conecta a la entrada *N_DATOSW* del módulo TOP_Comunicación. Esta señal, que indica el número de bytes a escribir, tendrá siempre el valor 1, ya que sólo se quiere escribir 1 byte en cada operación.

- **nveces_W(4:0)** se conecta a la entrada *nveces_W* del módulo ME_Repetidas_Acciones. Esta señal valdrá siempre 12, puesto que indica el número de operaciones de escritura que se quieren realizar.

- **nmuestras_R(4:0)** se conecta a la entrada *nmuestras_R* del módulo ME_Repetidas_Acciones. Esta señal valdrá siempre 1, puesto que indica el número de operaciones de lectura que se quieren realizar cada vez que se hace una comunicación.

La salida *uart_txd_out* del módulo SUPER_TOP se conecta al pin D4 de la placa, el cual es el receptor (RXD) del chip FTDI FT2232HQ de la Nexys.

Se implementa el proyecto VHDL en la placa, se ejecuta el programa de Matlab y se pulsa el botón de comienzo en la placa. Inicialmente podría parecer que los datos obtenidos son erróneos, ya que parece que la señal no varía, pero al eliminar la parte continua de la señal, se puede ver una señal que varía con un cierto período, es decir, se está leyendo bien.

Se realizan varias pruebas y se guardan estos datos. En las figuras 8.5.2.1 y 8.5.2.2 se pueden ver dos ejemplos de lecturas realizadas.

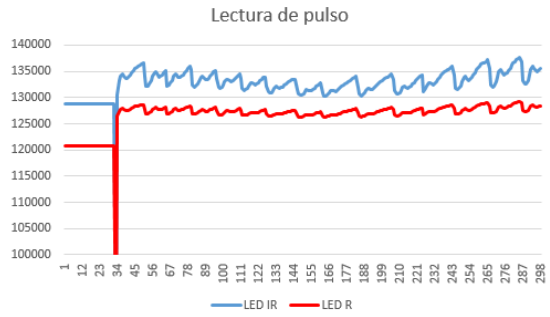


Figura 8.5.2.1 – Ejemplo 1 de lectura de 320 muestras del led R e IR.

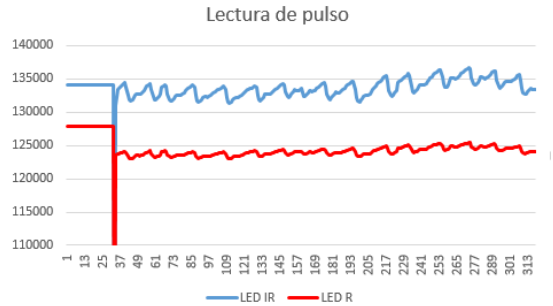


Figura 8.5.2.2 – Ejemplo 2 de lectura de 320 muestras del led R e IR.

Estas pruebas sirven para darse cuenta de que la primera lectura de la FIFO siempre es errónea ya que las primeras 32 muestras siempre son constantes, por lo que habrá que descartarlas.

Además, se puede ver como el led rojo presenta unos picos menos visibles que el led infrarrojo, es decir, la señal está más deformada y sería más complicado realizar los cálculos con ella, por lo que únicamente se trabajará con los datos obtenidos por el led infrarrojo. Esto es normal, ya que el led rojo se ve más afectado por la temperatura ambiente y por las posibles perturbaciones.

También se puede ver que los datos tienen una parte continua debido al tejido de la piel, músculos, etc. tal y cómo se había mencionado. Además,se obtiene una señal aparentemente periódica, en la que sus valores máximos están dentro del rango máximo ($140000 < 2^{18}$). Con esto se saca la conclusión de que se está realizando correctamente la captura y lectura de la pulsación.

8.6. Procesado de la señal en VHDL.

Para realizar los pasos descritos en el apartado 7.8 para obtener la frecuencia cardíaca, se ha diseñado el módulo PRUEBA_PROCESADO.

Este módulo consta de otros 7, siendo 4 de ellos máquinas de estados que controlan el procesado de la señal, uno donde se realizan las operaciones, y 2 memorias. Estos se explican a continuación.

8.6.1. Módulo RAM2

Memoria de 256 direcciones en la que se guarda la señal a medida que se va procesando hasta que se obtiene sin tendencia ni parte continua. Su señal de habilitación de escritura se activa desde la máquina de estados U_Control, ya que es la que controla esta parte del proceso. De este mismo módulo viene la señal de entrada de datos de la RAM2.

8.6.2. RAM3

Memoria de 256 direcciones en la que se guarda la función de autocorrelación. Su señal de habilitación de escritura se activa desde la máquina de estados ME_Calculo_Correlacion, ya que es la que controla esta parte del proceso. La entrada de datos proviene del módulo U_Operativa, puesto que es donde se realizan los cálculos con las muestras.

8.6.3. Módulo ME_Procesar_Signal

En este módulo se crea una máquina de estados para controlar los distintos pasos a realizar para obtener la frecuencia cardíaca, ya que esta se obtiene a partir de la ejecución secuencial de varias máquinas de estado.

Se ha decidido realizar 8 lecturas de la FIFO para obtener un total de 256 muestras del led rojo y 256 del led infrarrojo.

Para su cálculo, como ya se ha mencionado, sólo se va a emplear el led infrarrojo.

Tras realizar varias pruebas de grupos de 256 muestras y visualización de estas, se ha podido observar que en numerosas ocasiones la señal cambia de tendencia aproximadamente en el medio de esta, por lo que para evitar esto se ha decidido realizar el procesado y obtención de la frecuencia en dos grupos de 128 muestras, de forma que se calculará la frecuencia cardíaca de las 128 primeras muestras, luego de las 128 siguientes, y se obtendrá la media de estas dos. De esta forma, al evitar ese cambio brusco de tendencia el cálculo de la función de autocorrelación ofrece mejores resultados.

La máquina de estados diseñada para este módulo sigue el diagrama de la figura 8.6.3.1.

Cuando la máquina de estados ME_Repetidas_Acciones da la orden de que comience el procesado (*comienzo=1*) se pasa al siguiente estado. Lo que se hace en esta máquina es ir activando las diversas máquinas de estados hasta obtener la frecuencia cardíada final. La señal num_proces_aux a nivel bajo indica que se está realizando el cálculo con las primeras 128 muestras, y al pasar a nivel alto indica que se está haciendo con las segundas.

8.6.4. Módulo U_Control.

Este módulo se encarga de controlar las operaciones necesarias para obtener la señal sin la parte continua y sin tendencia.

En este módulo hay tres contadores instanciados. Dos de ellos generan la dirección de la memoria RAM del módulo SUPER_TOP. Uno de ellos genera la dirección durante el primer cálculo de frecuencia, y por lo tanto, se inicializa con el valor 1. Este se va incrementando de dos en dos, ya que sólo queremos coger los valores impares de la memoria correspondientes con la lectura del led infrarrojo. El otro contador genera la dirección cuando se está procesando el segundo grupo de 128 muestras, por lo que este módulo se inicializa con el valor 257 para obtener la segunda tanda de 128 muestras.

La máquina de estados tiene el diagrama de la figura 8.6.4.1.

En esta máquina de estados se realizan las operaciones de los pasos 1, 2, 3 y 4 del apartado 7.8. En el estado *FIN* tenemos los valores de la señal sin parte continua y sin tendencia guardados en la RAM2, y se activa la señal *fin_acondicionado* para pasar a realizar el cálculo de la f.a.

8.6.5. Módulo ME_Calculo_Correlacion.

Una vez tenemos la señal procesada, hay que pasar a calcular su función de autocorrelación, proceso que se controla desde la máquina de estados de este módulo, la cual aparece representada en la figura 8.6.5.1.

Para obtener la función de autocorrelación es necesario realizar las operaciones especificadas en la ecuación 7.8.0.3. Esto consiste en que para un punto, por ejemplo el primer valor de la función ya procesada, hay que ir multiplicando cada valor de esta señal por el valor de la señal desplazada una unidad, e ir sumando estos valores. Así obtendríamos el primer dato de la función de autocorrelación. El segundo valor se obtendría multiplicando cada dato de la

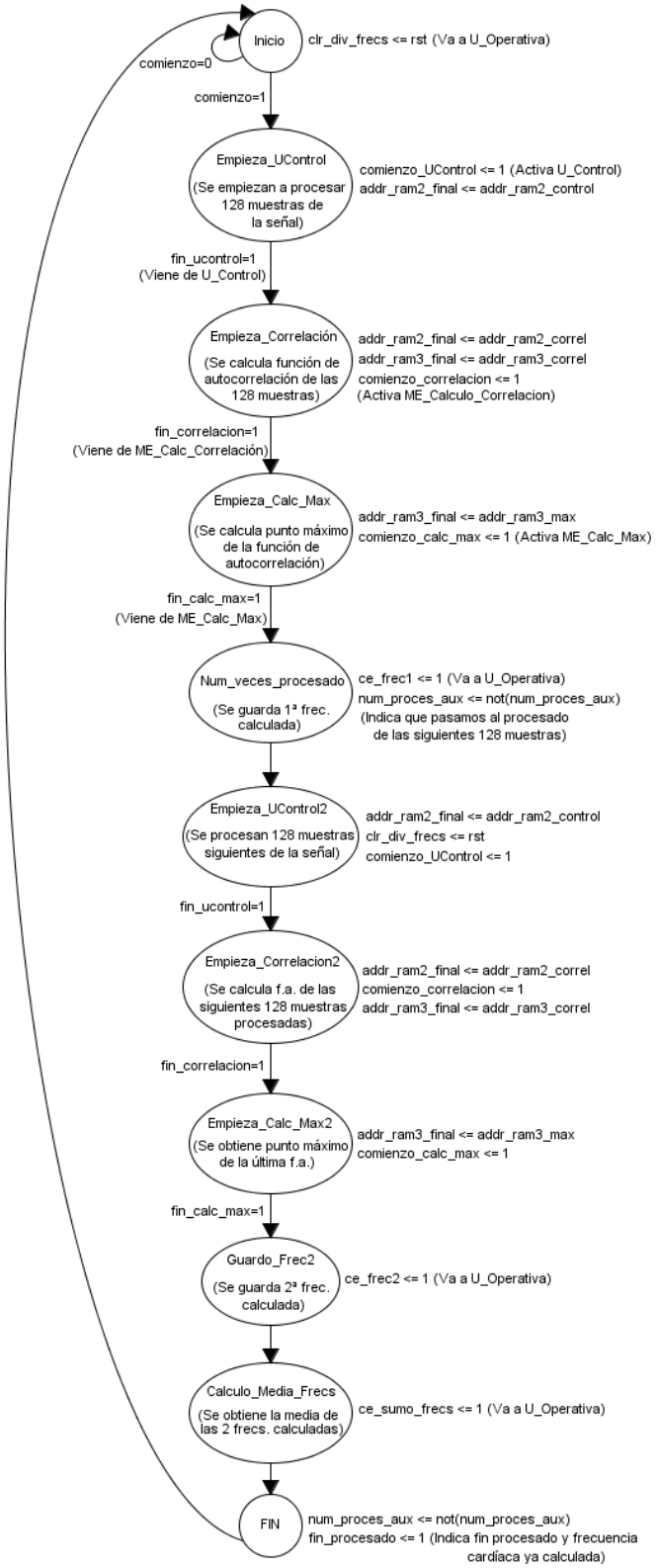


Figura 8.6.3.1 – Diagrama de estados de la máquina del módulo ME_Procesar_Signal

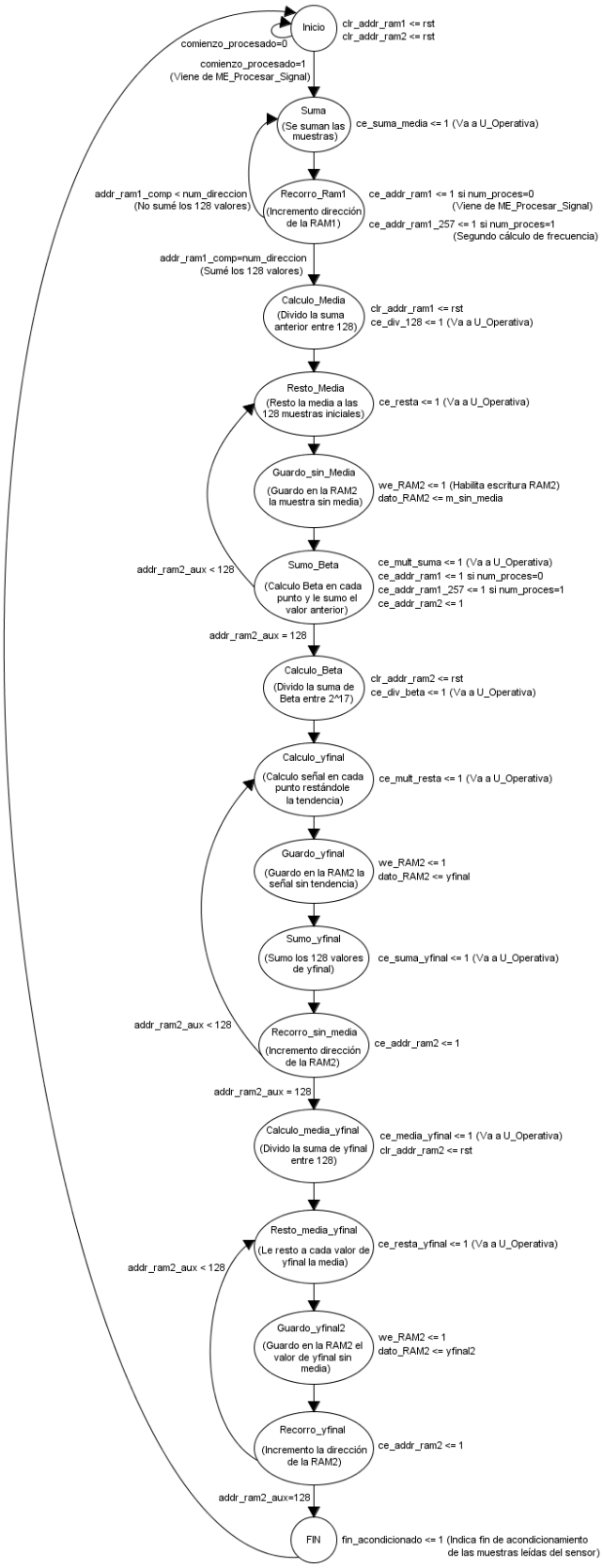


Figura 8.6.4.1 – Diagrama de estados de la máquina del módulo U_Control.

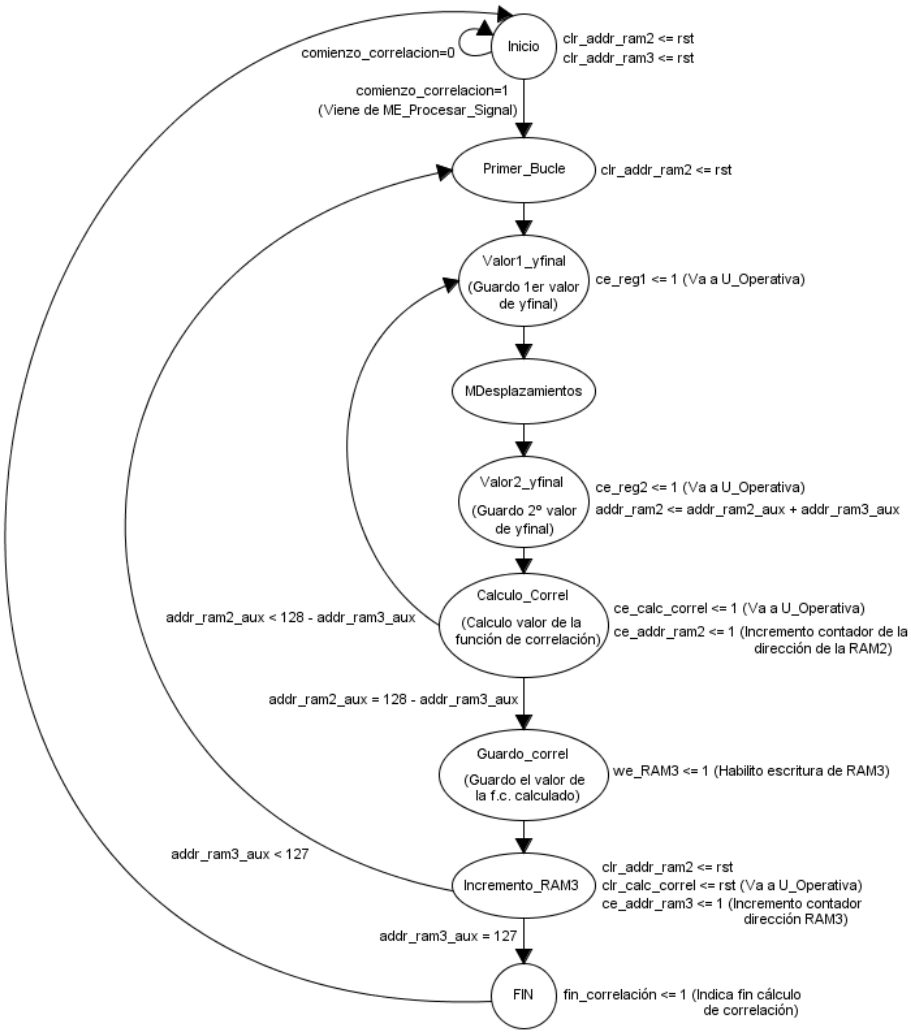


Figura 8.6.5.1 – Diagrama de estados de la máquina del módulo ME_Calculo_Correlacion.

señal por el que esté 2 posiciones más a la derecha e ir sumando estas multiplicaciones. Y así con cada uno de los 128 valores de la señal procesada. La dirección de la RAM3 equivale al valor de desplazamiento a realizar.

Este proceso no se realiza con el dato 128, ya que el último valor de la función de autocorrelación siempre será 0, puesto que al ser el último dato de la señal ya no se pueden realizar más desplazamientos.

Una vez guardados todos los datos en la RAM3, se pasa al estado FIN, en el que se indica que se ha finalizado el cálculo de la f.a. y se vuelve al estado Inicio.

8.6.6. Módulo ME_Calc_Max.

Una vez tenemos la función de autocorrelación, hay que obtener el primer máximo local. Esta operación se controla desde este módulo.

El diagrama de estados de esta máquina es el de la figura 8.6.6.1.

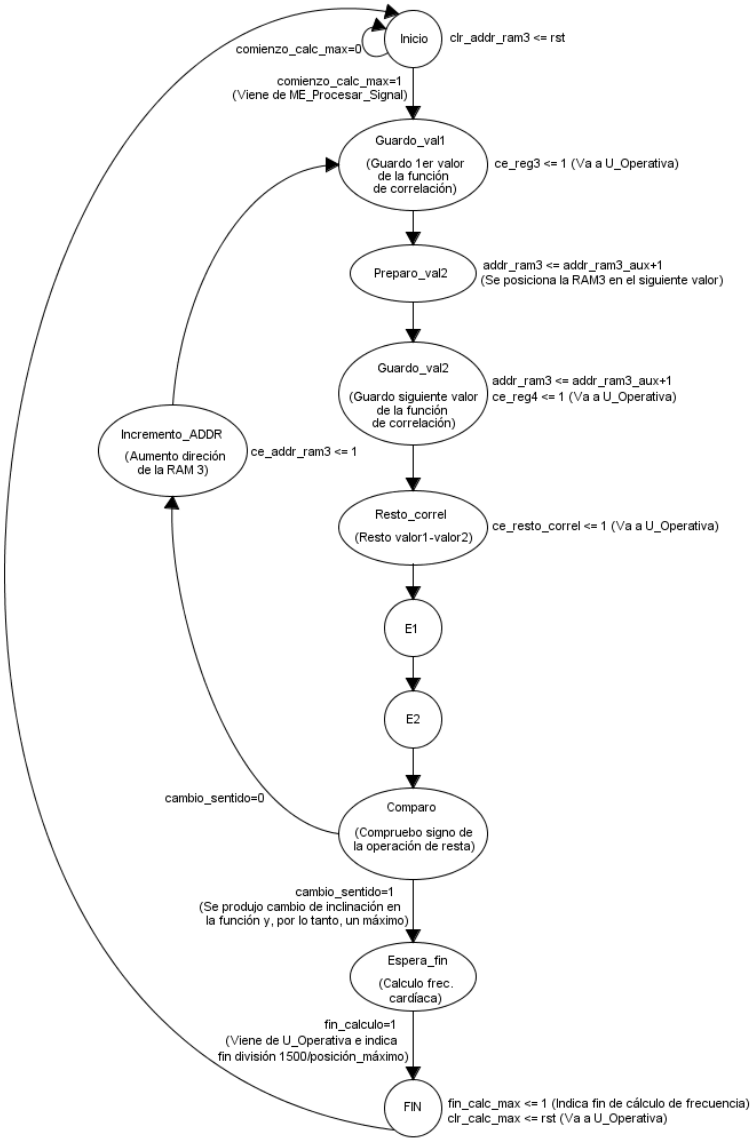


Figura 8.6.6.1 – Diagrama de estados de la máquina del módulo ME_Calc_Max.

Este módulo tiene instanciado un contador que genera la dirección de la posición que queremos leer de la RAM3.

Hay varias formas de obtener el primer máximo local de la función de autocorrelación, pero se ha optado por restarle al valor de la función que se está evaluando el valor siguiente. El bit de signo de esta resta es el que nos indica la pendiente de la señal. Se obtendrá el primer máximo en el momento en que la resta pase de valor negativo a positivo, es decir, con el primer flanco de bajada del bit de signo de la resta.

El punto de la función de correlación en el que se encuentra el máximo coincide con el valor de la dirección de la RAM3. Este valor equivale al número de muestras que hay entre cada pico de la señal original recibida del sensor.

En el estado *Espera_fin* se está esperando a que el módulo Divisor de U_Control divida

1500 entre la dirección actual de la RAM3, que será la que contiene el primer máximo local. El resultado obtenido es la frecuencia cardíaca.

Al finalizar la máquina de estados se activa la señal *fin_calc_max* para indicar el fin del cálculo de la frecuencia cardíaca.

8.6.7. U_Operativa

Es en este módulo dónde se realizan las operaciones necesarias para obtener la frecuencia cardíaca. En él están instanciados otros 9 módulos, algunos de ellos varias veces.

1. **Módulo SUMADOR:** este módulo suma el dato A+B y saca el resultado por SUMA. Está instanciado 3 veces: Suma_sinprocesar (suma dato a dato la señal obtenida del sensor), SUMADOR_YFINAL (suma la señal sin tendencia dato a dato) y SUMADOR_frecs (suma las 2 frecuencias calculadas).

2. **Módulo DIVISOR_128:** divide la entrada entre 128, es decir, elimina los 7 bits menos significativos del dato de entrada. Se instancia como OBTENGO_Media (divide la suma de los datos de la señal sin procesar entre 128 para calcular la media) y como OBTENGO_Media_YFINAL (divide la suma de los datos de la señal sin tendencia entre 128 para obtener la media).

3. **Módulo RESTADOR:** resta las entradas A-B. Se instancia como Resta_Media (resta la media de la señal sin procesar a esta para tener la señal sin parte continua) y como RESTA_MEDIA_YFINAL (resta la media de la señal sin tendencia a esta para tener la señal nivelada sin parte continua).

4. **Módulo MULT_Y_SUMA:** este módulo multiplica sus entradas A y B y al resultado le suma la entrada Result_ANT. Se instancia como Tendencia_MULT_Y_SUMA (calcula una parte de la obtención de β) y como Calculo_Correlacion (realiza la operación de multiplicación para obtener la función de autocorrelación).

5. **Módulo DIVISOR_2_17:** divide la entrada entre 2^{17}, es decir, le quita los 17 bit menos significativos a la señal de entrada. Se emplea para calcular β.

6. **Módulo MULT_Y_RESTA:** este módulo multiplica sus entradas A y B y le resta a la entrada M_Sin_Media el resultado de esta multiplicación. Se utiliza para obtener la señal sin tendencia.

7. **Módulo Registro:** guarda el dato de la entrada cuando se activa la entrada de habilitación *CE*. Está instanciado como Registro_Yfinal2_t (guarda el primer valor de la multiplicación para obtener la función de autocorrelación), Registro_Yfinal2_t (guarda el segundo valor de la multiplicación para obtener la función de autocorrelación), Registro_Frec1 (guarda la primera frecuencia cardíaca calculada) y como Registro_Frec2 (guarda la segunda frecuencia cardíaca calculada).

8. **Módulo Calculo_Max:** en este módulo, mediante un restador, un detector de flanco y un contador, se va restando el primer valor de la función de autocorrelación con el siguiente, y si el resultado tiene el mismo signo que antes, es que no se ha producido ningún máximo. Si el resultado de la operación pasa de negativo a positivo, es decir, el bit de signo pasa de 1 a 0, es que hemos obtenido un máximo y *cambio_sentido* pasa a valer 1.

9. **Módulo Division instanciado como Calculo_Frec:** en este módulo se realiza la división de 1500 entre el punto en el que encontramos el máximo.

8.7. Proyecto final

Ahora no se empleará el módulo en el que se realizaba la comunicación con el PC, puesto que no será necesario, sino que en su lugar está el módulo que engloba todas las partes del procesado. Se puede ver el esquema de conexionado del módulo SUPER_TOP de este proyecto final en el anexo 10.

8.7.1. MEMORIA_RAM

Módulo explicado anteriormente en el que guardamos las lecturas del sensor. Para realizar la comunicación con Matlab, en esta memoria se guardaban 2048 datos de 8 bits cada uno, ya que teníamos que enviar al PC los datos en bytes. Cómo ahora no es necesario enviar los datos, se pueden guardar las muestras completas y no byte a byte, de forma que los datos guardados en la memoria tendrán un tamaño de 18 bits y esta memoria tendrá 1024 direcciones.

8.7.2. Divisor_1Hz

Divisor de frecuencia que genera una señal de 190Hz para introducirla en el módulo *BCD7SEG*. El pulso a nivel alto dura únicamente 1 periodo de reloj. La frecuencia de esta señal no puede ser muy baja para evitar parpadeos en los segmentos del display.

8.7.3. Contador_2

Generar las direcciones de la RAM cuando se quiere escribir los datos en ella. Ahora se activa la señal de habilitación de escritura cada 3 lecturas del sensor, es decir, cada vez que leemos una muestra.

8.7.4. BIN_BCD

Este módulo es un conversor de binario a BCD (decimal codificado en binario). Lo que se hace es pasar el dato de la frecuencia cardíaca calculada, que está en binario, a código BCD para poder representarlo en los displays.

8.7.5. BCD7SEG

Una vez tenemos el dato en BCD, en este módulo se realiza la programación necesaria para representar el dato en los displays. De este módulo salen las salidas AN (ánodo), SEG (segmentos) y DP (punto decimal) que se conectan a los pines correspondientes de la placa.

8.7.6. ME_Repetidas_Acciones

El diagrama de estados de este módulo representado en las figuras 8.3.1.1 y 8.3.1.1 sufre algunos cambios.

El diagrama final de esta máquina es el de las figuras 8.7.6.1 y 8.7.6.2.

La diferencia entre este diagrama y el anterior, es que ahora tenemos un contador que se incrementa cada vez que se realiza la lectura de un byte, de forma que se activa la señal *CT_3bytes* cuando se han leído 3 bytes para así indicarle a la memoria que puede guardar un dato.

Además, ahora en vez del estado en el que esperábamos a que se realizase la comunicación con Matlab, ahora tenemos el estado *Comienza_procesado*, en el que se indica comience el procesado de la señal. En *Espera_Fin* se espera a que se termine el cálculo de la frecuencia cardíaca y se vuelve a leer el sensor.

Tenemos también la señal *primera_lectura*, la cual indica si se está realizando la primera lectura de la FIFO. Si esta está a nivel alto, no se guardarán los datos leídos de la FIFO en la RAM, pues que como ya se ha mencionado, es necesario descartar las primeras 32 muestras. De esta forma, la primera secuencia de lecturas de la FIFO se realizará 9 veces, para tener los 256 bytes de cada led. Luego, se leerá la FIFO 8 veces, ya que nos quedaremos con todos los datos.

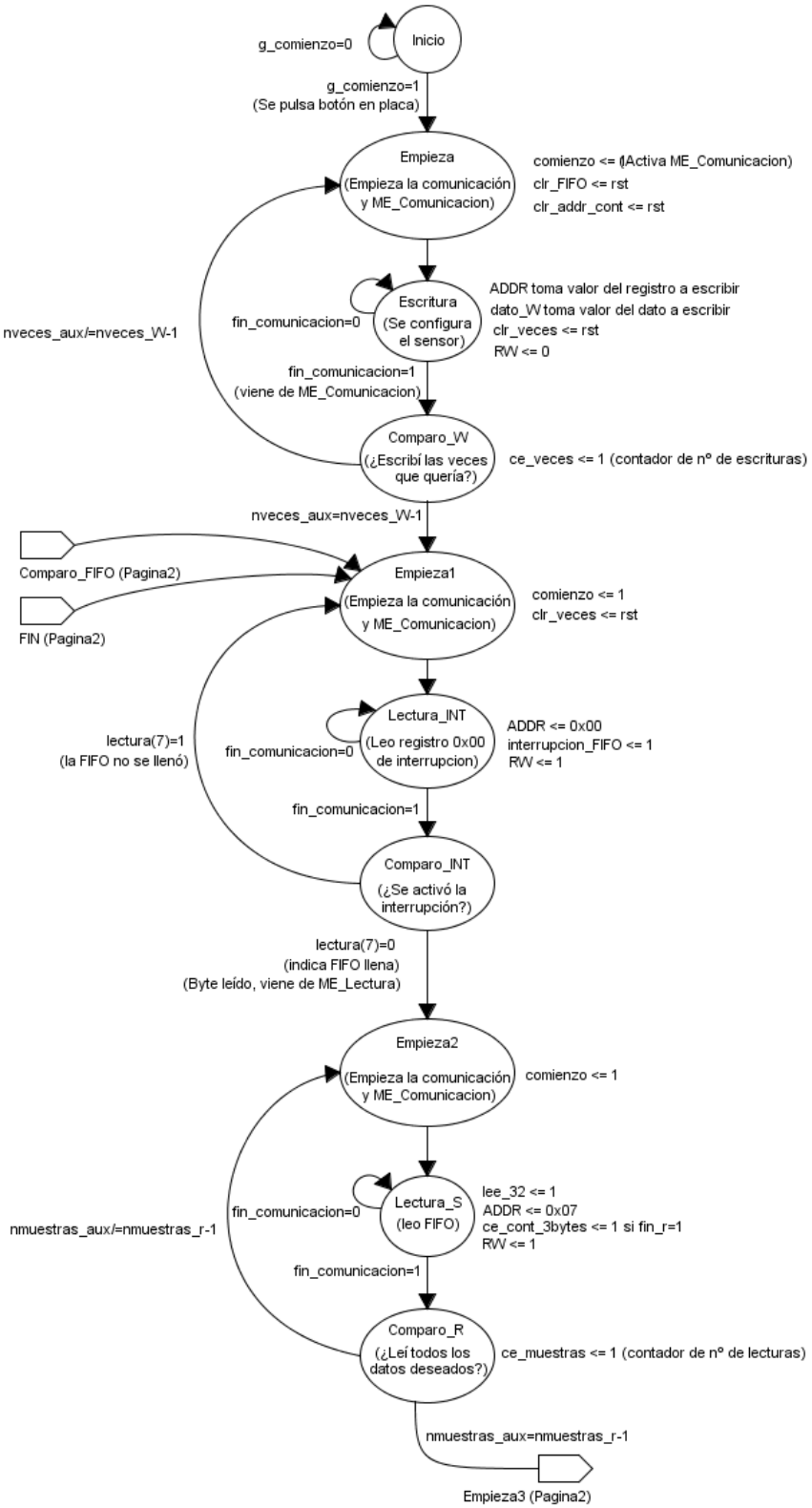


Figura 8.7.6.1 – Parte 1 del diagrama de estados final del módulo ME_Repetidas_Acciones.

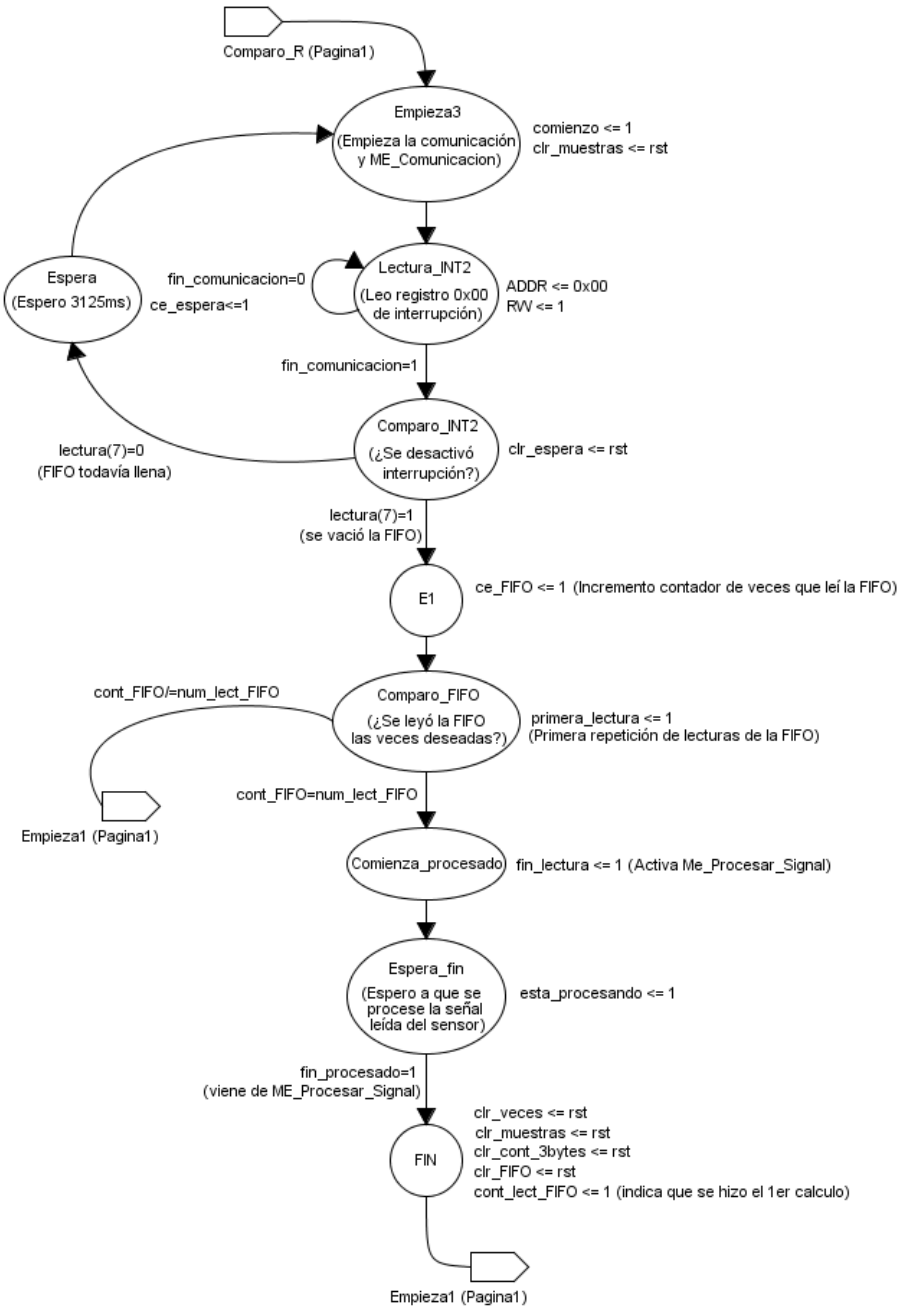


Figura 8.7.6.2 – Parte 1 del diagrama de estados final del módulo ME_Repetidas_Acciones.

8.7.7. TOP_Comunicacion

Este módulo permanece igual que cómo se explicó en el apartado 8.2.6. Se puede ver el esquema de conexiones entre sus distintos módulos en el anexo 10.

8.7.8. Prueba_Procesar

Este módulo permanece igual que cómo se explicó en el apartado 8.6. Se puede ver el esquema de conexiones entre sus distintos módulos en el anexo 10.

8.8. Pruebas del proyecto final.

Antes de implementar el proyecto en la placa, se prueba a generar una ROM de 512 datos, en la que se guardan las muestras recogidas en alguna de las pruebas realizadas con Matlab. Lo que se hace es que cuando estamos en el procesado de la señal, se cogerán los datos de la ROM y no los de la RAM. Se realiza alguna prueba y se ve que se realiza el procesado de forma correcta.

Se implementa el proyecto en la placa, realizando la asignaciones correspondientes en el ucf entre las entradas y salidas y los pines de la placa.

Para comenzar la lectura de la placa, se debe pulsar el botón ubicado en el pin N17. El valor del display se actualizará cada 12 segundos aproximadamente.

En las imágenes 8.8.0.1 y 8.8.0.2 se pueden ver varios ejemplos de mediciones correctas realizadas con el sensor.

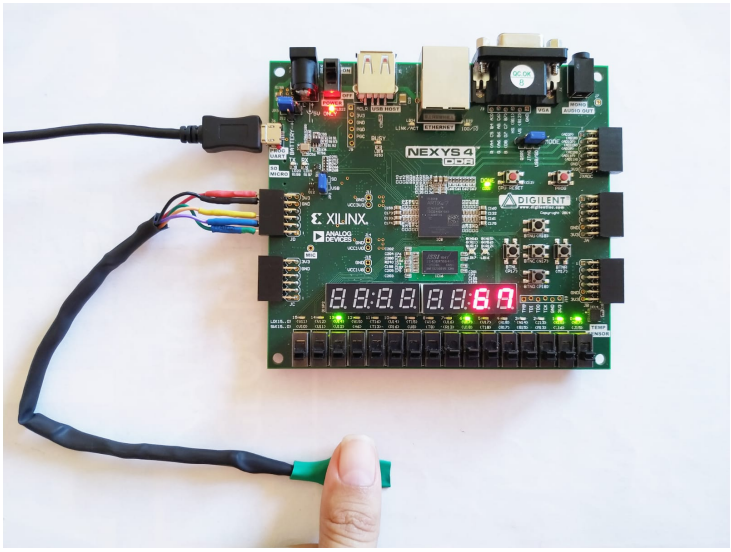


Figura 8.8.0.1 – Ejemplo1 de visualización de frecuencia cardíaca.

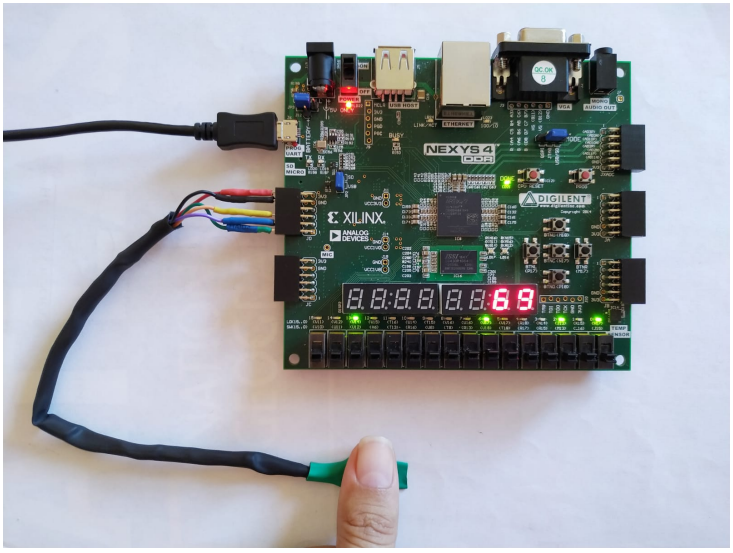


Figura 8.8.0.2 – Ejemplo2 de visualización de frecuencia cardíaca.

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice del documento ANEXOS

9 Documentación de partida

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Pérez García,Natalia*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** Feb2019
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Adquisición y procesado de datos de sensores con interfaz I2C usando FPGA

Número TFG: *770G01A149*

TUTOR:(Titor) *Meizoso Lopez,Maria Del Carmen*

COTUTOR/CODIRECTOR: *Esteban Jove Pérez*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Jueves, 5 de Septiembre del 2019

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Pérez García,Natalia

DESCRIPCIÓN Y OBJETIVO:OBJETO:Esta propuesta consiste en utilizar la plataforma Nexys4 DDR para la adquisición de datos de sensores que utilicen la interfaz I2C, procesarlos y mostrarlos en un display. Se completará el diseño añadiendo una UART para enviar los datos recogidos a un PC.

ALCANCE:

Se pretende obtener un sistema totalmente funcional, que se desarrolle según las etapas siguientes:

1. Análisis y selección del sensor o sensores a emplear
2. Desarrollo en VHDL de la interfaz de comunicación I2C.
3. Desarrollo en VHDL de la UART.
4. Simulación y pruebas sobre el sistema físico.

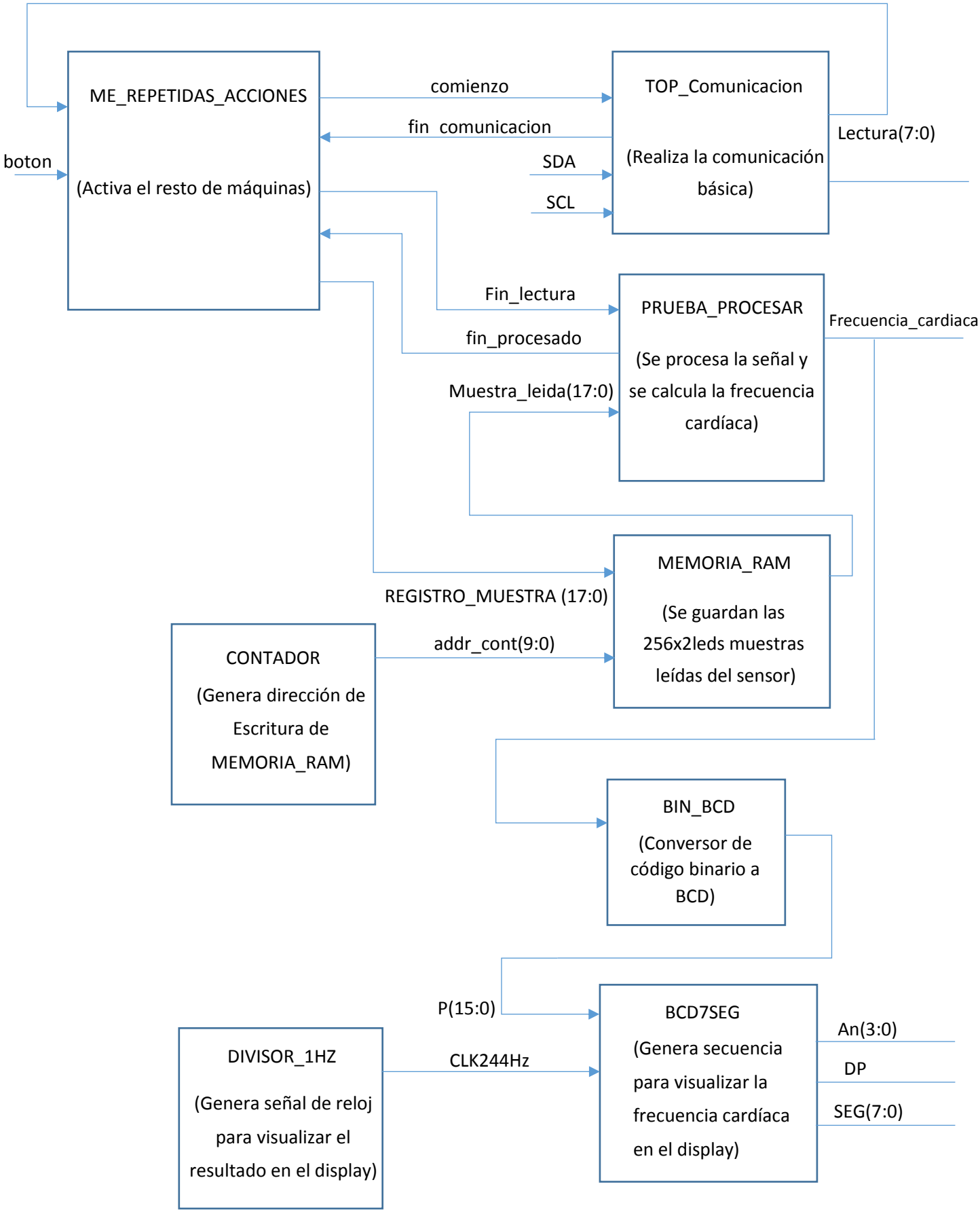
10 Esquemas de conexionado

En este capítulo figuran los esquemas de conexionado de los módulos principales en los que se instancian los módulos que se han ido explicado hasta el momento.

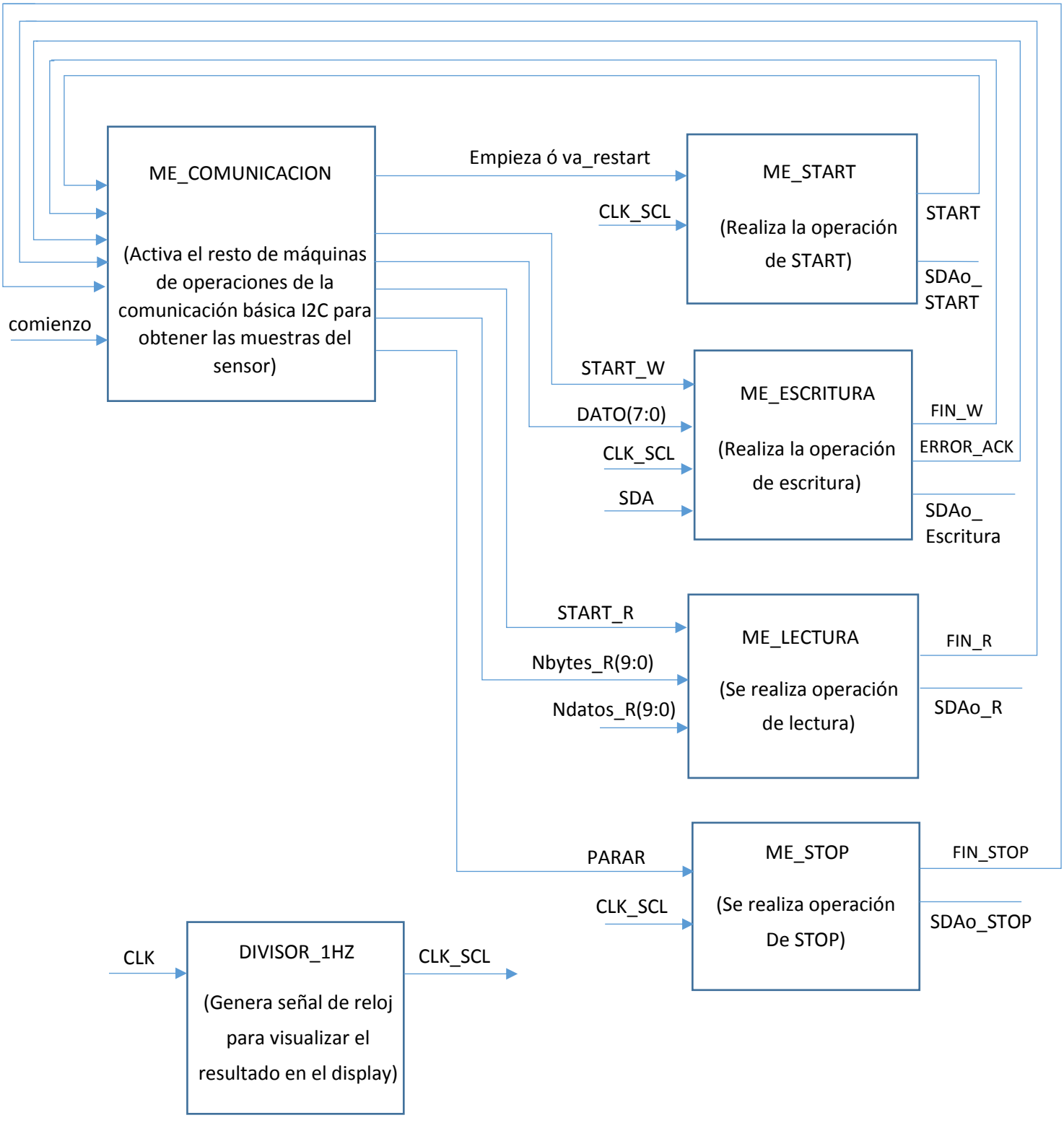
En estos esquemas no aparecen todas las señales de entrada y salida de los módulos, puesto que sería complicado visualizar de forma clara el funcionamiento del proyecto, sino que figuran algunas de las señales principales para poder establecer una conexión general entre los distintos módulos presentados.

1. **Esquema de conexionado del módulo SUPER_TOP**
2. **Esquema de conexionado del módulo TOP_COMUNICACION**
3. **Esquema de conexionado del módulo PRUEBA_PROCESAR**
4. **Esquema de conexionado del módulo TOP_UART_TX**
5. **Esquema de conexionado del módulo UART_TX**

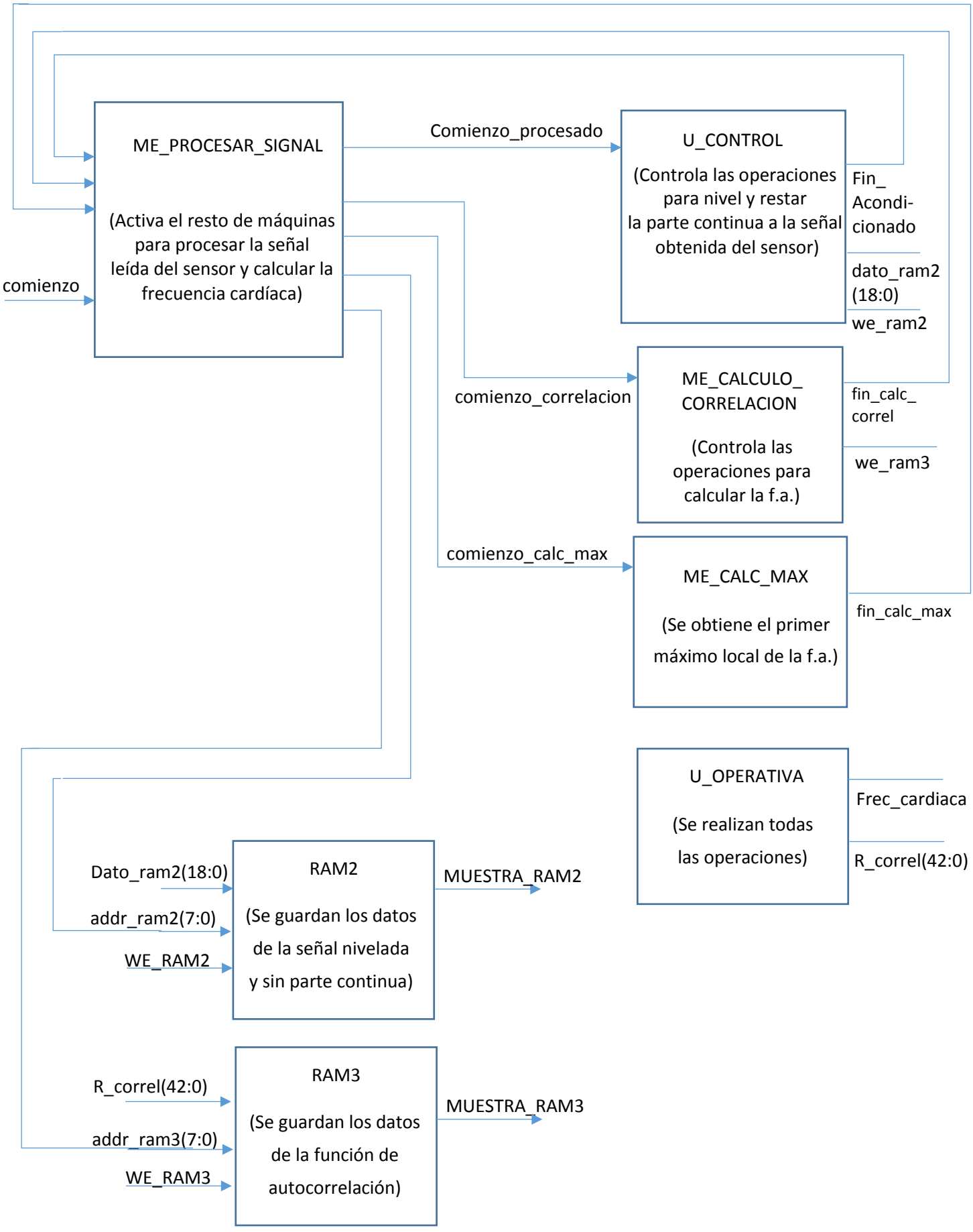
SUPER_TOP


ME_REPETIDAS_ACCIONES
(Activa el resto de máquinas)
boton
comienzo
fin_comunicacion
SDA
SCL
TOP_Comunicacion
(Realiza la comunicación básica)
Lectura(7:0)
Fin_lectura
fin_procesado
PRUEBA_PROCESAR
(Se procesa la señal y se calcula la frecuencia cardíaca)
Frecuencia_cardiaca
Muestra_leida(17:0)
REGISTRO_MUESTRA (17:0)
MEMORIA_RAM
(Se guardan las 256x2leds muestras leídas del sensor)
CONTADOR
(Genera dirección de Escritura de MEMORIA_RAM)
addr_cont(9:0)
BIN_BCD
(Conversor de código binario a BCD)
P(15:0)
DIVISOR_1HZ
(Genera señal de reloj para visualizar el resultado en el display)
CLK244Hz
BCD7SEG
(Genera secuencia para visualizar la frecuencia cardíaca en el display)
An(3:0)
DP
SEG(7:0)

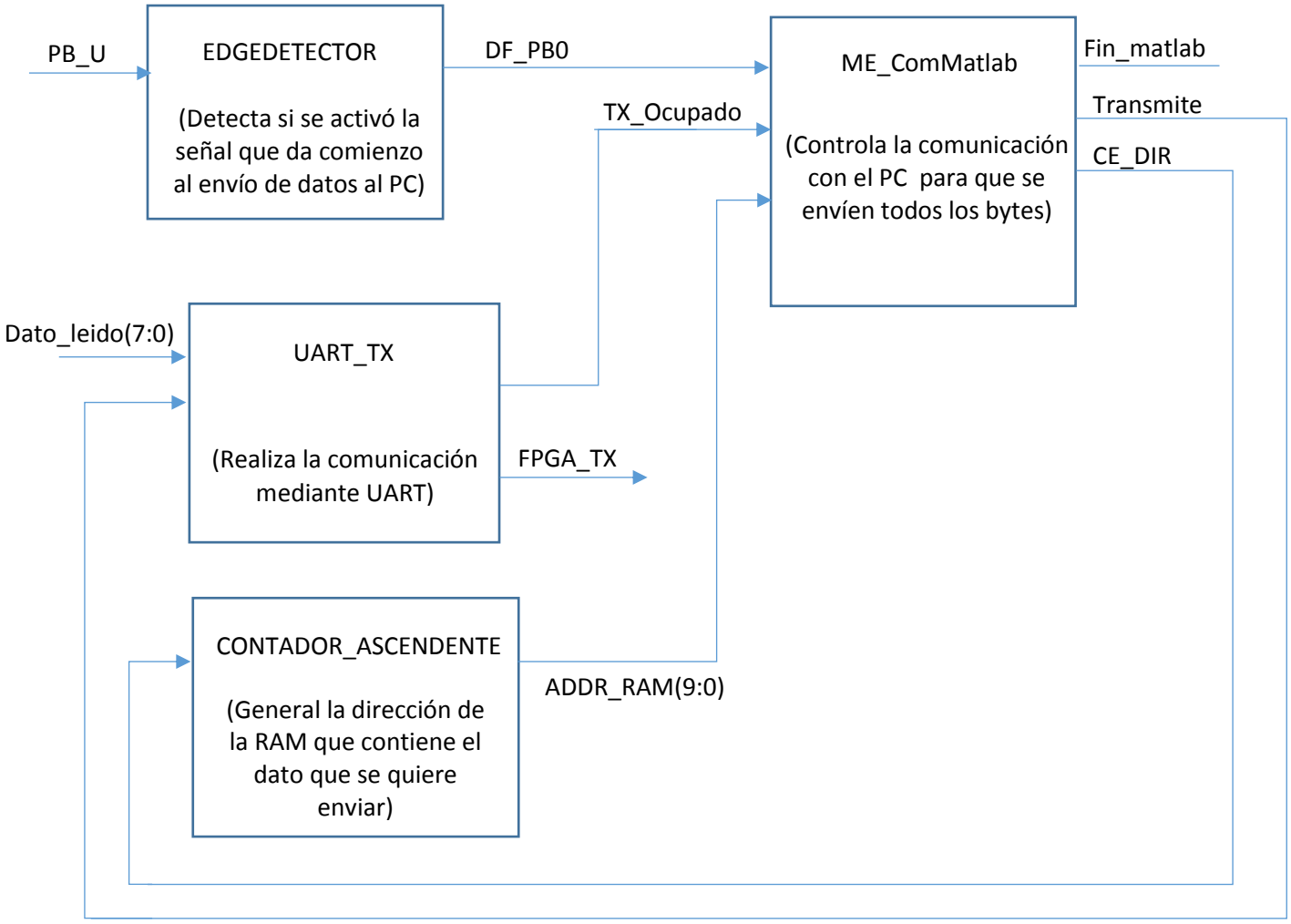
TOP_COMUNICACION



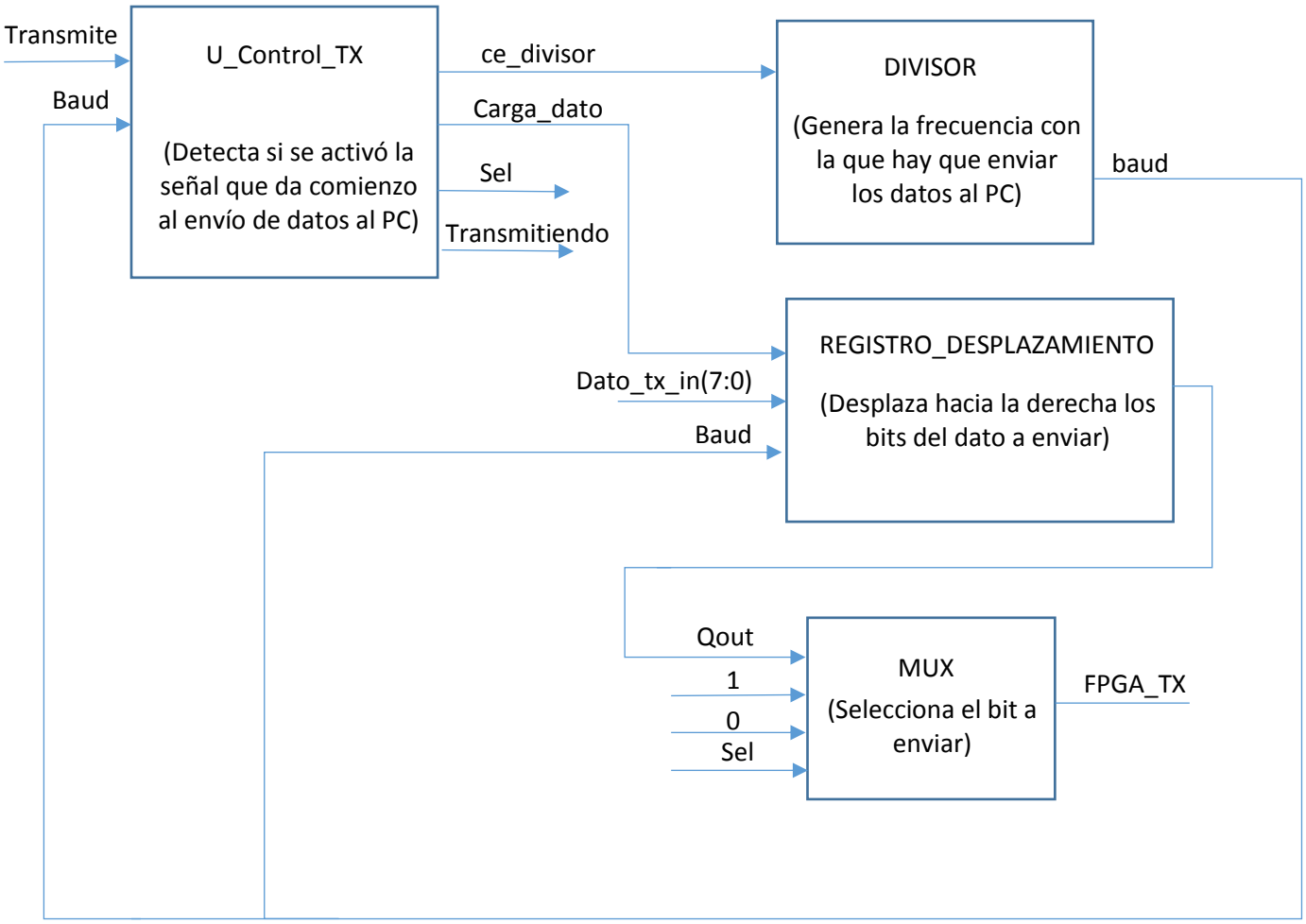
PRUEBA_PROCESAR



TOP_UART_TX


PB_U
EDGEDETECTOR
(Detecta si se activó la señal que da comienzo al envío de datos al PC)
DF_PB0
ME_ComMatlab
(Controla la comunicación con el PC para que se envíen todos los bytes)
Fin_matlab
Transmite
CE_DIR
TX_Ocupado
Dato_leido(7:0)
UART_TX
(Realiza la comunicación mediante UART)
FPGA_TX
CONTADOR_ASCENDENTE
(General la dirección de la RAM que contiene el dato que se quiere enviar)
ADDR_RAM(9:0)

UART_TX


Transmite
Baud
U_Control_TX
(Detecta si se activó la señal que da comienzo al envío de datos al PC)
ce_divisor
Carga_dato
Sel
Transmitiendo
DIVISOR
(Genera la frecuencia con la que hay que enviar los datos al PC)
baud
REGISTRO_DESPLAZAMIENTO
(Desplaza hacia la derecha los bits del dato a enviar)
Dato_tx_in(7:0)
Baud
Qout
1
0
Sel
MUX
(Selecciona el bit a enviar)
FPGA_TX

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice del documento PLIEGO DE CONDICIONES

11 Pliego de condiciones

Disponemos, por un lado, de la Nexys4 DDR y por otro de la placa MAXREFDES117 con los cables ya conectados y acondicionados para realizar su conexionado de forma más sencilla. A continuación, se detallará el conexionado a realizar para su utilización, y los pasos a seguir para su correcto uso.

11.1. Conexionado sensor-Nexys

La placa MAXREFDES117 consta de 8 pines para conectar los cables a los 5 entradas y salidas de esta placa, estando 3 de ellos repetidos (SDA,SCL,INT) tal y cómo se ve en la figura 11.1.0.1.

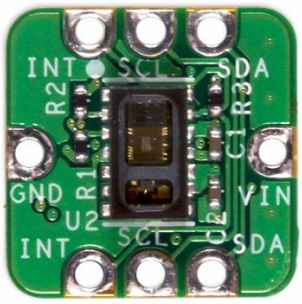


Figura 11.1.0.1 – Pines de la placa MAXREFDES117.

Se puede realizar la conexión indistintamente en los pines repetidos, por lo que realmente sólo son necesarias 5 conexiones.

Se ha asignado a cada uno de los pines un cable de un color distinto para poder diferenciarlos. Además, la placa de acondicionamiento se ha recubierto de material termo retráctil con la finalidad de eliminar así posibles perturbaciones del ambiente o producidas al tocar el dedo los circuitos de la placa impresa.

- **Vin:** cable rojo.
- **GND:** cable negro.
- **INT:** cable verde.

- **SCL:** cable violeta.
- **SDA:** cable naranja.

Es necesario conectar cada uno de estos cables a uno de los pines del puerto PMOD JD de la Nexys4 DDR. La asignación de cada uno de estos cables respecto a los pines de este conector es la siguiente:

- **Cable rojo:** pin 6.
- **Cable negro:** pin 5.
- **Cable naranja:** pin 3.
- **Cable violeta:** pin 2.
- **Cable verde:** pin 1.

En la figura 11.1.0.2 se puede ver la numeración de los pines de este tipo de conectores para la Nexys4 DDR.

Figura 11.1.0.2 – Numeración de los pines del puerto PMOD de la Nexys4 DDR.

Además, es necesario conectar el cable Micro B-USB al puerto Micro-USB del pin J6 de la Nexys y a un puerto USB del PC.El conexionado final es el que se puede ver en la figura 11.1.0.3.

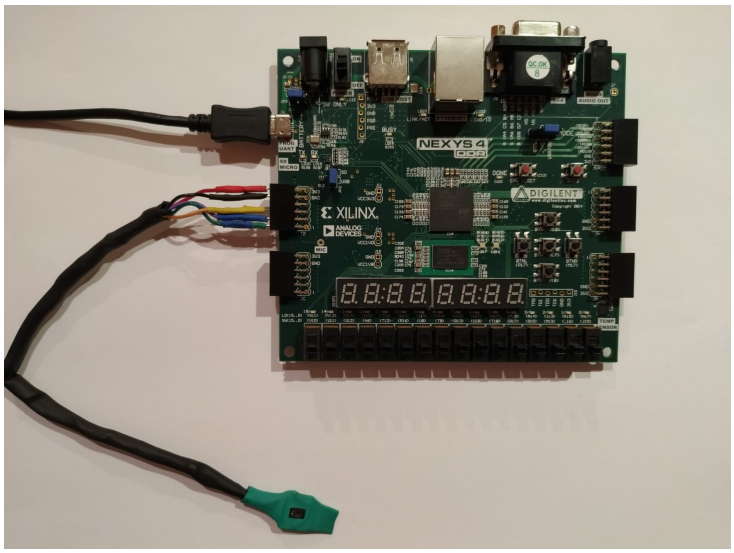


Figura 11.1.0.3 – Conexionado entre el sensor y la Nexys4 DDR.

11.2. Guía para su correcto uso

Una vez se ha realizado el conexionado necesario, hay que implementar el programa en la placa.

Con el programa ya cargado en la Nexys, el usuario debe colocar el dedo encima del sensor, sin realizar mucha presión y de forma que sea la yema del dedo la que cubre el cristal del sensor.

Luego, bastará con pulsar el botón conectado al pin N17 de la placa para que el sensor comience a realizar la captura de datos. Se visualizará una nueva medición del pulso cada 11 segundos aproximadamente.

En caso de que no esté el dedo sobre el sensor, o se obtenga una señal fuera del rango de [20,220], aparecerá sobre el display el mensaje “DEDO”.

Si se desea parar las mediciones, hay que cambiar de estado el switch J15 de la placa, de forma que si este está a nivel alto se realiza el reseteo de las operaciones. Para volver a comenzar las mediciones hay que volver a pulsar el mismo botón mencionado antes.

En la figura 11.2.0.4 se ven marcadas con una etiqueta las partes a utilizar en la placa. A continuación, se detalla a qué corresponde cada uno de los elementos marcados.

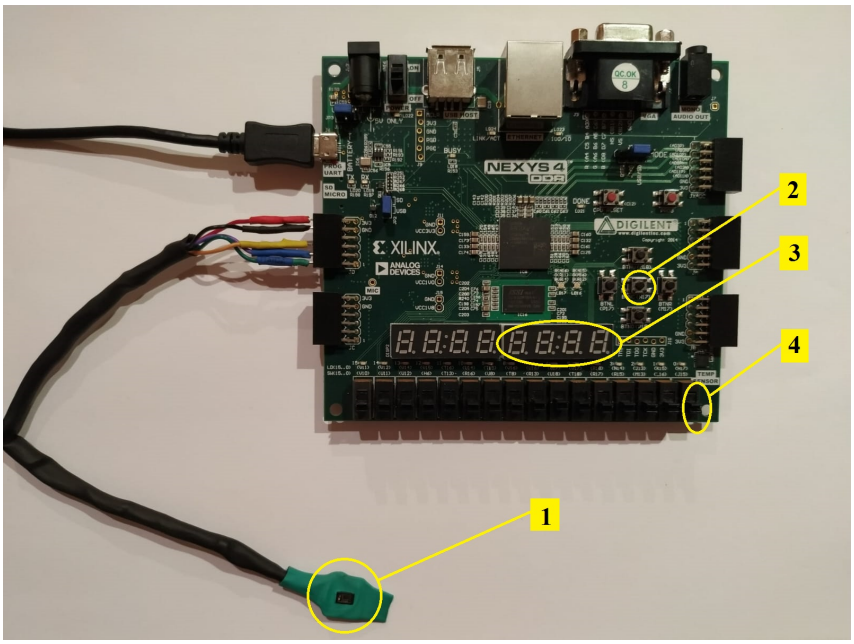


Figura 11.2.0.4 – Identificación de elementos de la Nexys a utilizar por el usuario [13]

1. Sensor
2. Botón de comienzo
3. Display en el que se visualiza la frecuencia cardíaca.
4. Switch de reset actualmente en la posición de reposo.

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

ESTADO DE MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice del documento ESTADO DE MEDICIONES

12 Estado de mediciones

Puesto que se ha optado por incluir la placa MAXREFDES117 en lugar de únicamente el sensor de pulso MAX30102, ya que esta incluye todos los circuitos de acondicionamiento del sensor, se han reducido considerablemente el número de componentes a emplear.

Los componentes empleados para la realización de este proyecto son los que figuran en la tabla 12.0.0.1

Elementos	Referencia	Unidades
Nexys4 DD Artix-7	Farnell: 2490174	1
MAXREFDES117	Farnell: 2627165	1
Cable Micro B-USB	Farnell: 2907930	1
Cable de puente macho-macho 300mm	Farnell: 2762508	5
Tubo termorretractil de 8mm,20cm de longitud	Farnell: 1256846	1
Tubo termorretractil de 9.5mm,4cm de longitud	Farnell: 1187637	1

Tabla 12.0.0.1 – Tabla de componentes empleados.

TÍTULO: **ADQUISICIÓN Y PROCESADO DE DATOS DE SENSORES CON INTERFAZ I2C USANDO FPGA**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: NATALIA PÉREZ GARCÍA

Índice del documento PRESUPUESTO

13 Presupuesto

13.1. Materiales

Concepto	Unidades	Unidades mínimas a pedir	Precio unitario (€)	Total (€)
Nexys4 DD Artix-7	1	1	287.00	287.00
MAXREFDES117	1	1	15.01	15.01
Cable Micro B-USB	1	1	4.66	4.66
Cable de puente macho-macho 300mm	5	10	0.414	4.14
Tubo termorretractil de 8mm,20cm de longitud	1	1200mm	0.0045	5.39
Tubo termorretractil de 9.5mm,4cm de longitud	1	1200mm	0,0031	3.73
			TOTAL (€)	319.93

Tabla 13.1.0.1 – Presupuesto de materiales.

13.2. Mano de obra

Concepto	Unidades	Precio unitario (€)	Total (€)
Mano de obra	300	15	4500
		TOTAL (€)	4500

Tabla 13.2.0.2 – Presupuesto de mano de obra.

13.3. Total

Concepto	Precio Total (€)
Material	319.93
Mano de obra	4500
Total	4819.93

Tabla 13.3.0.3 – Presupuesto total.