Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Sistema multisensorial para la obtención de parámetros significativos del ritmo circadiano

Autor: Aitor Gómez-Pimpollo Fernández

Tutor/es: Juan Jesús García Domínguez

2018

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Sistema multisensorial para la obtención de parámetros significativos del ritmo circadiano

Autor: Aitor Gómez-Pimpollo Fernández

Tutor: Juan Jesús García Domínguez

TRIBUNAL:

Presidente: Juan Carlos García García

Vocal 1º: José Luis Martín Sánchez

Vocal 2º: Juan Jesús García Domínguez

Calificación: ……..

Fecha: ……

ÍNDICE

1. RESUMEN

En este proyecto se pretende diseñar un sistema multisensorial capaz de aportar datos influyentes que afectan directamente en el desarrollo del ciclo circadiano de las personas.

Los datos significativos que presentan menos dificultades a la hora de su obtención y que permiten evaluar y realizar estudios fiables en este ámbito son la actividad física, la temperatura corporal periférica, la temperatura ambiente y la luz ambiente.

Dichos datos serán aportados por sensores y con ayuda de un microcontrolador serán procesados para posteriormente transferirlos a un ordenador y poder representarlos.

Para la aplicación de representación de datos se utilizará el programa MATLAB®.

1.1. PALABRAS CLAVE

Ciclo circadiano, microcontrolador, MATLAB®.

2. ABSTRACT

The purpose of this project is to design a multisensorial system capable of providing meaningful data that have a direct impact on the circadian rythm function.

Meaningful data gathered more easily and that enable the realization and evaluation of reliable studies are the following: physical activity, body temperature and light and room temperature.

Such data are provided by sensors. Then, with the help of a microcontroller, they are processed, transferred to a computer and represented.

The app used for the data representation is MATLAB®.

2.1. KEYWORDS

Circadian rythm, microcontroller y MATLAB®.

3. ABREVIATURAS

- NSQ: Núcleo supraquiasmático.
- ADC: Convertidor analógico digital.
- DAC: Convertidor digital analógico.
- PC: Personal computer, ordenador
- PWM: Modulación por ancho de pulsos.
- SCL: Línea del “System Clock” del I2C.
- SDA: Línea del “System Data” del I2C.
- GND: Punto del circuito con una tensión de 0 voltios.
- $\mathrm{Vcc}_{3.3\ \mathrm{V}}$: Punto del circuito con una tensión de 3.3 voltios.
- $\mathrm{Vcc}_{5\ \mathrm{V}}$: Punto del circuito con una tensión de 5 voltios.
- $\mathrm{I_{ADC}}$: Corriente de malla del sensor de temperatura corporal.
- $\mathrm{V_{ADC}}$: Tensión en el punto de entrada del ADC.
- $\mathrm{R_{324\Omega}}$: Resistencia auxiliar de montaje de valor 324 Ohmios.
- $\mathrm{R_{PT100}}$: Resistencia variable asociada al sensor de temperatura corporal.
- $\mathrm{V_{R_{PT100}}}$: Tensión que soporta el sensor de temperatura corporal.
- $\mathrm{R_{4.7K\Omega}}$: Resistencia de 4.7 KΩ necesaria para configuración del puerto I2C.
- MSB: Siglas inglesas de “*Most Significant Byte*”, o byte de mayor peso.
- LSB: Siglas inglesas de “*Low Significant Byte*”, o byte de menor peso.

4. RESUMEN EXTENDIDO

Es habitual hoy en día escuchar a científicos y personas relacionadas con el ámbito de la salud hablar sobre la importancia que tienen las buenas rutinas de descanso en las personas. En un primer pensamiento personal podemos decir si descansamos bien o no, pero, ¿verdaderamente somos conscientes de esta importancia y de la gran cantidad de factores que influyen en que nuestra rutina de descanso sea buena?

Desde hace aproximadamente dos décadas, el ámbito de la cronobiología está tomando fuerza, y nos podemos preguntar: "¿Qué es esto de la cronobiología?". Pues bien, la cronobiología se define como la disciplina científica que estudia los cambios rítmicos (ritmos biológicos) en los distintos niveles de organización de los organismos, siendo un ritmo biológico la recurrencia de un fenómeno biológico en intervalos regulares. Uno de estos ritmos biológicos es el ciclo del descanso, también llamado ciclo circadiano.

La principal causa por la que la cronobiología está en una tendencia ascendente es porque múltiples estudios han demostrado que existe una relación entre la disrupción del sistema circadiano y patologías como el cáncer, la obesidad, alteraciones cognitivas y afectivas, y el envejecimiento prematuro entre otras. Por este motivo, surge la necesidad de crear sistemas multisensoriales capaces de recopilar datos significativos influyentes de una manera fiable, para poder establecer técnicas que permitan evaluar el funcionamiento del sistema circadiano de una manera objetiva. Los datos que se recopilan para estos estudios son los de temperatura ambiente, luz ambiente, temperatura corporal y actividad física del individuo.

En este trabajo se quiere desarrollar un prototipo de prestaciones fiables que contenga sensores que aporten datos de la temperatura corporal de la persona en cuestión, su actividad física y la luz y la temperatura ambiente a las que está expuestas durante el transcurso del ciclo. Todos los sensores se conectan a un microcontrolador que agrupa estos datos en una trama y los envía a un ordenador.

Una vez el sistema prepara la trama y la transmite al PC, con la ayuda de una aplicación generada en MATLAB®, se reciben los datos y se representan en tiempo real en cuatro gráficas independientes.

5. MEMORIA DESCRIPTIVA

5.1. INTRODUCCIÓN: CICLOS CIRCADIANOS Y CONCEPTOS BIOLÓGICOS

Todos los seres vivos presentan funciones en sus organismos en las que se producen variaciones medibles dependiendo de los cambios lumínicos. Por este motivo, esta oscilación entre luz y oscuridad demuestra que existe una periodicidad biológica durante el transcurso de un día completo.

Estas variaciones regulares de las funciones orgánicas reciben el nombre de ritmos biológicos, los cuales pueden tener duraciones diversas.

De acuerdo a estas variaciones, existen tres ritmos biológicos principales, que son el ritmo circadiano, ultradiano (ritmo cuyo período es menor de 24 horas, es decir, ocurre más de una vez al día) e infradiano (ritmo cuyo período es mayor de 24 horas, es decir, que ocurre menos de una vez al día). [1]

El ritmo circadiano es un ciclo de 24 horas, en el que se generan y sincronizan oscilaciones de las variables biológicas, de tal forma que permite reconocer los fenómenos de vigilia-sueño y su acción homeostática en los procesos sistémicos de todo ser vivo. La homeostasis es el modo en el que los organismos procuran mantener el equilibrio bioquímico entre todas sus partes para que el conjunto funcione bien. Por ello es importante conocer bien y mantener constante el ciclo circadiano, ya que alteraciones en él pueden provocar alteraciones del equilibrio bioquímico en los organismos y por tanto patologías.

Otro concepto importante en este ámbito es la cronobiología, ya que es la disciplina científica encargada de estudiar los ritmos biológicos en los distintos niveles de organización de los organismos. [2]

Para entender el ciclo circadiano en los mamíferos y poder aplicar las diferentes variables a un sistema electrónico de medición, primero debemos comprender las bases moleculares de este ciclo, principalmente compuestas por tres elementos: relojes circadianos, vías de entrada y vías de salida. Los mismos se describen a continuación.

5.1.1. RELOJES CIRCADIANOS

Hay un marcapasos principal que se encuentra en el núcleo supraquiasmático (NSQ) del hipotálamo y es el responsable de la mayoría de los ritmos biológicos de los mamíferos.

Una lesión en este núcleo sería responsable de arritmicidad, la cual podría desencadenar graves problemas.

Por otro lado, pero no menos importante, se encuentran los marcapasos periféricos que están distribuidos por el organismo, sobretodo en el córtex cerebral, pero también en otros órganos y tejidos como el hígado, y se encargan de regular el núcleo supraquiasmático.

5.1.2. VÍAS DE ENTRADA

Como hemos dicho, los ciclos circadianos aparecen gracias a los núcleos supraquiasmáticos (NSQ), pero además deben de estar sincronizados con los ciclos ambientales, es decir, el reloj debe de "ponerse en hora" con unos sincronizadores.

Uno de los sincronizadores más importante es el ciclo luz-oscuridad, pero existen otros como el ejercicio, los horarios de las comidas, el patrón del sueño…

5.1.3. VÍAS DE SALIDA

El NSQ envía proyecciones nerviosas y mediadores humorales como la melatonina y el cortisol, a zonas del cerebro que intervienen en la regulación de los patrones de sueño-vigilia, comportamiento, temperatura corporal y a los centros neuroendocrinos y órganos periféricos. [1]

5.2. FACTORES QUE INTERVIENEN EN EL CICLO CIRCADIANO Y REPERCUSIÓN

Fisiológicamente, una de las particularidades del ritmo circadiano es que tiene una naturaleza endógena, es decir, que está bajo la influencia de un reloj marcapasos interno, que produce las actividades rítmicas autónomas.

A su vez, este marcapasos está influenciado por señales provenientes del medio externo; por tanto, en caso de que no haya señales externas que lo alteren, el ritmo circadiano se mantiene dentro de su parámetro normal de 24 horas de periodo de vigilia-sueño.

Existen diversos factores que pueden influir en el ciclo circadiano, y dependiendo de su expresión van a tener lugar diversas repercusiones en el organismo. Alteraciones en la luz, trabajos nocturnos o cambios rápidos de zona horaria conocidos como “jet lag”, pueden provocar alteraciones en el ciclo acarreando consecuencias tales como hipertensión, aumento del riesgo de cáncer, problemas de sueño, obesidad o alteraciones en el embarazo, entre otras.

Para la monitorización del funcionamiento del sistema circadiano, es necesario desarrollar técnicas que permitan su evaluación de forma objetiva; por ello, lo más conveniente es tomar las medidas de factores como la temperatura corporal, la actividad motora, la exposición a la luz ambiente, la temperatura ambiental, la secreción de melatonina o cortisol y la expresión de los genes reloj. [1], [2]

5.2.1. CICLO CIRCADIANO, LUZ Y TEMPERATURA AMBIENTAL

La luz es captada por los ojos, los cuales contienen en la retina una capa de células nerviosas especializadas (conos y bastones) para captar la luz y permitir la visión.

Sin embargo, la luz no sirve únicamente para la visión. Existen otro tipo de células fotoreceptoras retinales, que contiennen el fotopigmento melanopsina, que se estimula con longitudes de onda de la banda azul del espectro (oscila entre los 460 y 480 nm) . Estas células forman el sistema denominado “No formador de Imágenes”, el cual activa una serie de estructuras cerebrales que producen respuestas fisiológicas como la sincronización circadiana, interviniendo en la modulación del sueño.

Por tanto, en condiciones normales, los centros reguladores del sueño se activan al acabar el día y disminuir la señal luminosa, de manera que notemos la sensación de somnolencia una hora más tarde, y se desactivan progresivamente por la mañana. La transcendencia en la actualidad viene dada porque la luz artificial del mundo moderno

(pantallas de televisión, móviles y ordenadores) puede contribuir a que los centros cerebrales que regulan el sueño no se activen hasta horas más tarde.

La exposición a la luz está en relación con la temperatura ambiental, y ambos constituyen los dos principales factores que afectan a los ritmos marcadores del ciclo circadiano. [3]

5.2.2. CICLO CIRCADIANO Y MELATONINA

La melatonina es una hormona que forma parte de una de las principales vías de salida de nuestro reloj biológico. En los mamíferos es producida por la glándula pineal durante la noche, de manera que prepara al organismo para dormir.

Su síntesis está controlada por el marcapasos circadiano central, es decir, el NSQ. Su función está unida al ciclo luz-oscuridad a través de las células retinohipotalámicas y células ganglionares provistas del pigmento de la melanopsina.

Durante la tarde-noche, los niveles de luz azul son bajos, por tanto el NSQ libera noradrenalina, que es un neurotransmisor que permita la síntesis de melatonina y así se inducirá el sueño. Los valores plasmáticos máximos de melatonina se alcanzan entre las 02:00 y las 04:00h. Por la mañana, los niveles de luz azul serán máximos, lo cual inhibirá la síntesis de melatonina por parte de los marcapasos circadianos centrales y tendrá lugar la vigilia y la contracción de la pupila. [3]

Su perfil de secreción puede verse afectado por la luz principalmente, pero también por la actividad, la cafeína y algunos fármacos antiinflamatorios y beta bloqueantes.

Además de participar en el ciclo del sueño, esta hormona tiene otra serie de efectos importantes sobre el organismo que podrían verse alterados si no funciona bien el ciclo circadiano como son:

- Función antioxidante y eliminación de radicales libres.
- Efectos sobre el sistema inmunológico: Propiedades inmunoestimulantes y antiinflamatorias y regulación del equilibrio entre las reacciones normales y su excesiva manifestación.
- Efecto antitumoral. Esto es debido a la suma de los dos efectos anteriores.

Por todo ello, la buena regulación de esta hormona tiene un efecto primordial en el organismo. [4]

5.2.3. CICLO CIRCADIANO Y CORTISOL

El cortisol es un glucocorticoide secretado por la glándula adrenal con un ritmo muy marcado, elevándose al momento del despertar y disminuyendo a lo largo del día hasta alcanzar su mínimo un par de horas después de iniciar el sueño. Por tanto, lleva un patrón opuesto al de la melatonina.

Su patrón está fuertemente influenciado por la luz y por el propio ciclo de sueño-vigilia, de manera que la privación de este, situaciones de insomnio o múltiples despertares nocturnos pueden aumentar sus niveles. [3]

5.2.4. CICLO CIRCADIANO, PRESIÓN ARTERIAL, ACTIVIDAD Y TEMPERATURA CORPORAL

La presión arterial mantiene un ritmo circadiano con niveles más bajos durante la noche debido a la inactividad física y los cambios de postura al dormir. En cambio, al despertar, estos valores sistólicos y diastólicos aumentan y varían notablemente la frecuencia cardíaca. [5]

Un inadecuado descanso nocturno puede dar lugar a alteraciones en la presión arterial y dar lugar a consecuencias negativas sobre todo en personas con diabetes o hipertensión.

La actividad física y la temperatura corporal también siguen un ciclo circadiano. Tenemos dos picos de rendimiento máximo, que son entre las 10 y las 13h y posteriormente entre las 16 y 19h. Después de las 20h no se recomienda hacer ejercicio ya que nuestro organismo, coincidiendo con la bajada de luz y aumento de la secreción de melatonina se prepara para el descanso. Por ello muchos expertos recomiendan realizar mayor actividad física y entrenamientos exhaustivos por la mañana.

El ritmo de la actividad es uno de los más utilizados en la práctica clínica dada su facilidad de recogida de datos y comodidad. Y además está muy en relación con el patrón de sueño-vigilia, de manera que cuando el individuo está despierto los valores son más elevados que durante el descanso.

Así mismo, el pico de temperatura corporal coincide aproximadamente entre las 15 y 21h, coincidiendo con los picos de consumo de oxígeno, ventilación y nivel metabólico. La temperatura corporal es un marcador endógeno muy estable y por ello útil para el estudio de las alteraciones circadianas. La temperatura corporal inicia su descenso en la franja horaria del inicio del periodo del sueño, un mínimo en la madrugada y un aumento

leve en la mañana coincidiendo con el despertar, hasta llegar a su pico máximo mencionado anteriormente. [3]

5.2.5. CICLO CIRCADIANO, SUEÑO E INTEGRACIÓN DE VARIABLES

Como hemos ido comentando en cada uno de los apartados, el patrón de sueño-vigilia es un patrón integrador que afecta a todos los anteriores. Es uno de los ritmos más importantes, y este, a su vez, está determinado por dos componentes:

- Un componente homeostático que depende del tiempo que llevamos despiertos.
- Un componente circadiano dependiente del ritmo de temperatura corporal central y mediado por la secreción de melatonina.

Alteraciones de este ciclo tienen importantes consecuencias para la salud. Su magnitud es tal que aparecen patologías como enfermedades cardiovasculares, demencias u otros problemas sanitarios más importantes. [2], [3]

5.3. OBJETIVOS

El objetivo principal de este proyecto consiste en crear un sistema multisensorial que sea capaz de obtener y proporcionar datos significativos de una persona para evaluar su ciclo circadiano, para que a partir de éstos se pueda estudiar y analizar conductas beneficiosas o perjudiciales en relación dicho ciclo.

Este sistema se puede dividir en tres etapas principales:

- Sistema sensorial: Es la etapa compuesta por todos los sensores encargados de aportar los datos requeridos para la evaluación del ciclo circadiano.
- Adquisición y pre-procesamiento: Esta etapa está basada en un microcontrolador que gestiona el funcionamiento del sistema, recibe los datos de los sensores y los prepara para el envío a un ordenador.
- Envío y visualización: En esta etapa el PC recibe los datos previamente procesados por el microcontrolador y mediante la creación de una aplicación en MATLAB® los mostrará en pantalla en 4 gráficas que se generan y muestran su evolución en tiempo real.

El diagrama de bloques asociado al sistema se puede ver en la ilustración 1.

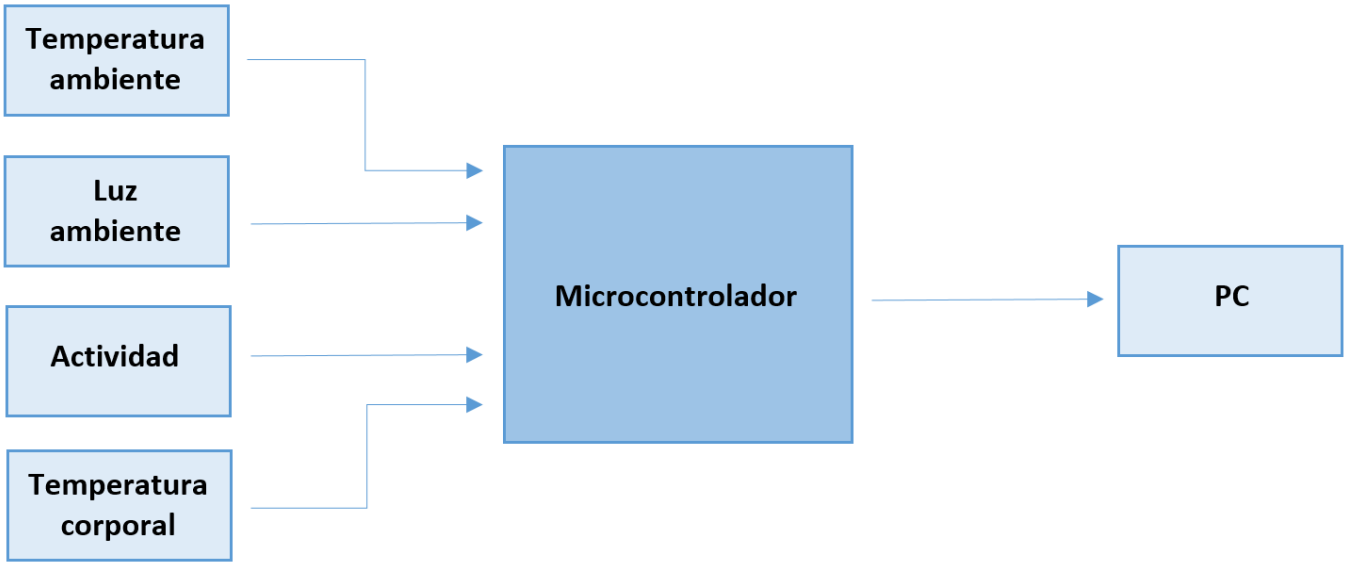

Ilustración 1 Diagrama de bloques del sistema

6. MEMORIA JUSTIFICATIVA

6.1. PARÁMETROS A MEDIR POR EL SISTEMA Y JUSTIFICACIÓN

6.1.1. TEMPERATURA CORPORAL

Para la medición de la temperatura corporal central, los medios que se han utilizado hasta la actualidad eran sondas rectales portadas por el sujeto varios días, lo cual suponía gran incomodidad. Estas sondas medían la temperatura corporal central, que se caracteriza por un ritmo de niveles elevados durante el día y un descenso durante la noche.

Para evitar esto, se está estudiando el ritmo de la temperatura de la piel como un nuevo ritmo marcador, y esto es lo que va a medir nuestro sistema.

La temperatura de la piel, también llamada temperatura corporal periférica, está asociada al eje simpático-parasimpático, de forma que durante el día, predomina el sistema simpático y desciende la temperatura de la piel, mientras que por la noche, se inhibe el simpático y predomina la actividad parasimpática, teniendo una mayor temperatura en la piel. Es decir, sigue un patrón inverso a la temperatura corporal central.

En la ilustración 2 se puede ver la evolución de la temperatura corporal central (línea roja) y la temperatura corporal periférica (línea violeta) durante un periodo circadiano.

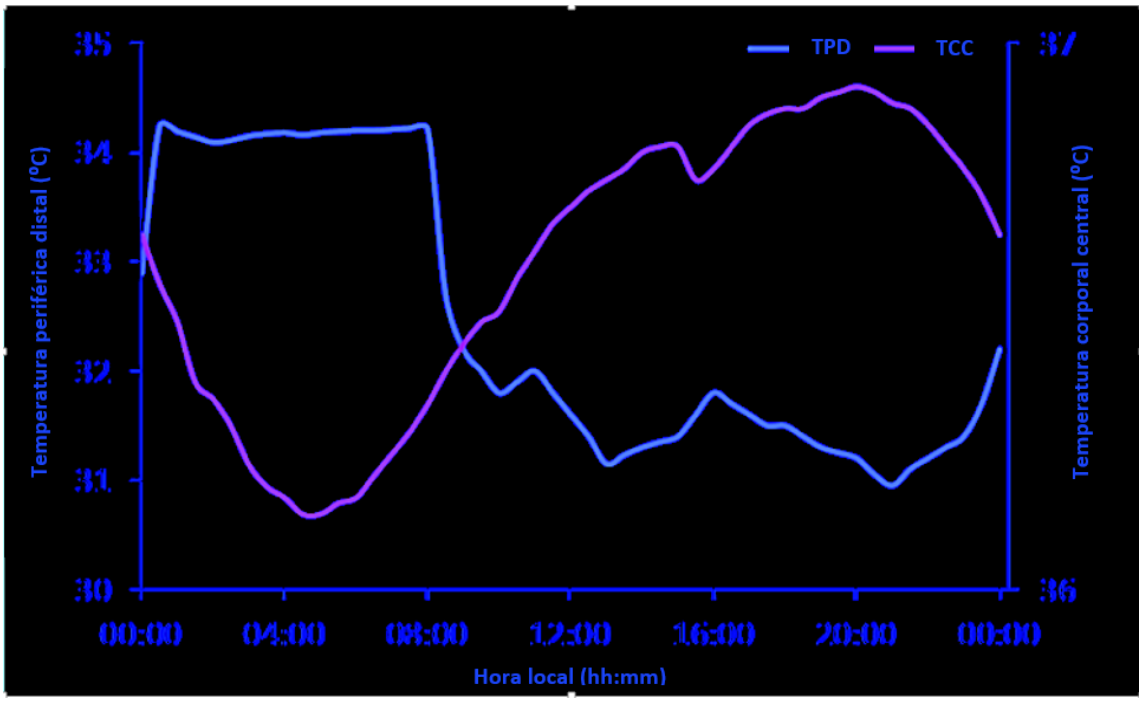

Ilustración 2 Temperatura corporal central (TCC) y temperatura periférica distal (TPD) [3]

6.1.2. LUZ AMBIENTE

Los datos de exposición a la luz son de gran importancia ya que a partir de ellos se puede sacar bastante información de interés a la hora de estudiar el desarrollo del ciclo circadiano. Se pueden obtener parámetros como el periodo horario entre la salida y puesta de sol, los momentos de carencia de luz en los que se empieza a segregar la melatonina y por tanto empieza a acentuarse la sensación y necesidad del sueño, secreción del cortisol…

Se debe prestar especial atención a posibles picos producidos por la luz artificial en momentos de teórica oscuridad ya que pueden causar alteraciones notables y, por tanto, podrían ser puntos de estudio y análisis de afección negativa al desarrollo del ciclo.

En la siguiente tabla de muestran en intervalos los niveles de luz de situaciones corrientes a las que puede estar expuesta una persona durante el desarrollo del día.

Situación	Luxes
Noche	0.001-0.02
Luna llena	0.2-0.6
Día nublado, en interior	5-50
Día nublado, en exterior	50-500
Día soleado, en interior	100-1.000
Bajo luz directa de sol	100.000
Habitación, salón	150-300
Mesa oficina/lectura	500-700
Supermercados/exposiciones	750-1.000
Mesa dibujo/trabajo	1.000-1.500

6.1.3. TEMPERATURA AMBIENTE

El patrón de temperatura ambiental oscila unos 15-20 grados centígrados entre el día y la noche; aunque en la actualidad, debido a que se pasa mucho tiempo en espacios interiores, esta diferencia de temperatura se reduce a 5-10 grados.

Los datos de temperatura ambiente también ayudan a establecer el horario de salida y puesta de sol. Esto sirve para contrastar los datos de exposición a la luz y poder filtrar momentos de exposición a luz artificial.

Se puede ver en la ilustración 3 la relación existente entre la temperatura ambiente y el ciclo de luz-oscuridad en el transcurso de un día completo.

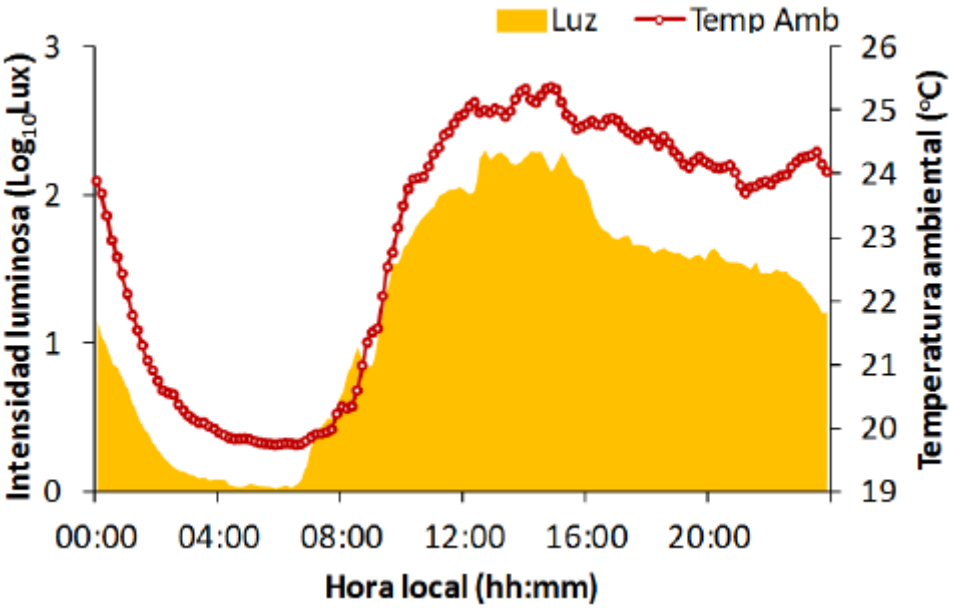

Ilustración 3 Relación ciclo de luz-oscuridad y temperatura ambiente [3]

6.1.4. ACTIVIDAD

Aunque algunos expertos en la materia no catalogan este parámetro como marcador del funcionamiento del sistema circadiano, es uno de los parámetros más utilizados para el estudio del mismo debido a la comodidad y al fácil registro de sus datos. Simplemente con el acelerómetro de un dispositivo móvil se puede hacer una aproximación de la actividad de la persona portadora y, de hecho, existen aplicaciones que actualmente lo realizan.

Este parámetro aporta información influyente en el ciclo circadiano debido a que la actividad está directamente relacionada con el ciclo sueño-vigilia, ya que en los

momentos en los que el individuo está durmiendo la actividad disminuye, llegando a puntos de inactividad plena en lo que a movimientos corporales se refiere.

Respecto a la obtención de datos, se debe tener en cuenta que el aparato que aporta los datos puede estar expuesto a vibraciones normales de la vida cotidiana o incluso que la persona bajo estudio pueda quitarse el dispositivo, lo que provoca datos que desvirtúan la realidad. Por este motivo, el estudio de estos datos debe ser prolongado en el tiempo, entorno a una semana como mínimo, para que se puedan establecer conclusiones fiables con los datos obtenidos. [3]

6.2. MICROCONTROLADOR Y CONCEPTOS GENERALES

Para el enlace y la comunicación entre el PC y los sensores donantes de datos es necesario un elemento de proceso, en este caso se propone un microcontrolador, que actúe como un puente entre ellos. Este microcontrolador es el encargado de tener en memoria el programa que controla todos los sensores y envía los datos al PC para almacenarlos y representarlos.

El microcontrolador elegido para el desarrollo del sistema es el LPC1768-Mini-DK2 de la familia ARM Cortex-M3. Es un dispositivo versátil que nos ofrece unas posibilidades adecuadas para el control de los sensores que se necesitan.

Algunas de las características de la Mini-DK2 son las siguientes:

- 8 canales de ADC con 12 bits para expresar el resultado.
- Un DAC de 10 bits para expresar el resultado.
- 70 pines de entrada/salida de propósito general.
- 4 UARTs para comunicación con PC vía USB.
- Protocolo de comunicación I2C para conexión de esclavos.
- 4 timers de propósito general con diferentes modos de funcionamiento.
- 6 salidas de PWM para control de motores.

La adquisición de los datos del sistema multisensorial se realizarán o bien a través del canal 0 del ADC disponible, o a través del bus I2C. En el caso del sensor de temperatura corporal se utilizará el ADC; y el bus I2C para las conexiones de los sensores de temperatura ambiente, luz ambiente y actividad. Finalmente se requerirá la UART0 para la transmisión de los datos desde el microcontrolador al PC, para su posterior tratamiento.

En los siguientes puntos se detallan los recursos utilizados de la placa, aportando las partes del código referentes a la configuración de los mismos.

6.2.1. ADC

El convertidor analógico digital se define como un dispositivo electrónico capaz de recibir una señal analógica y convertirla en una señal digital. La señal analógica entrante puede ser tanto tensión como corriente.

Este proceso de digitalización de la señal se realiza mediante un escalado del rango máximo de valores analógicos (3.3 voltios) que puede recibir el ADC en función del rango máximo de valores digitales que dispone. En el caso de la Mini-DK2, el rango digital es

[0, 4095] ya que dispone de 12 bits para expresar el resultado. El registro en el que el ADC expresa el resultado digitalizado de cada conversión es LPC_ADC->ADGDR de los bits 15 al 4.

6.2.2. BUS I2C

El I2C es un bus serie de datos que se utiliza para la comunicación entre diferentes partes de un circuito. En esta comunicación existen dos jerarquías, la de maestro y la de esclavo.

Este bus tiene dos líneas donde se conectan los dispositivos:

- SLC (System Clock): Es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (System Data): Es la línea por donde circulan los datos entre maestro y esclavos.

Además estas líneas deben tener un pull-up a una tensión de 5V con unas resistencias de 4.7 KΩ. Esto hace que se polaricen en estado alto y permite conectar en paralelo varias entradas y salidas.

El esquema del bus se muestra en la ilustración 4.

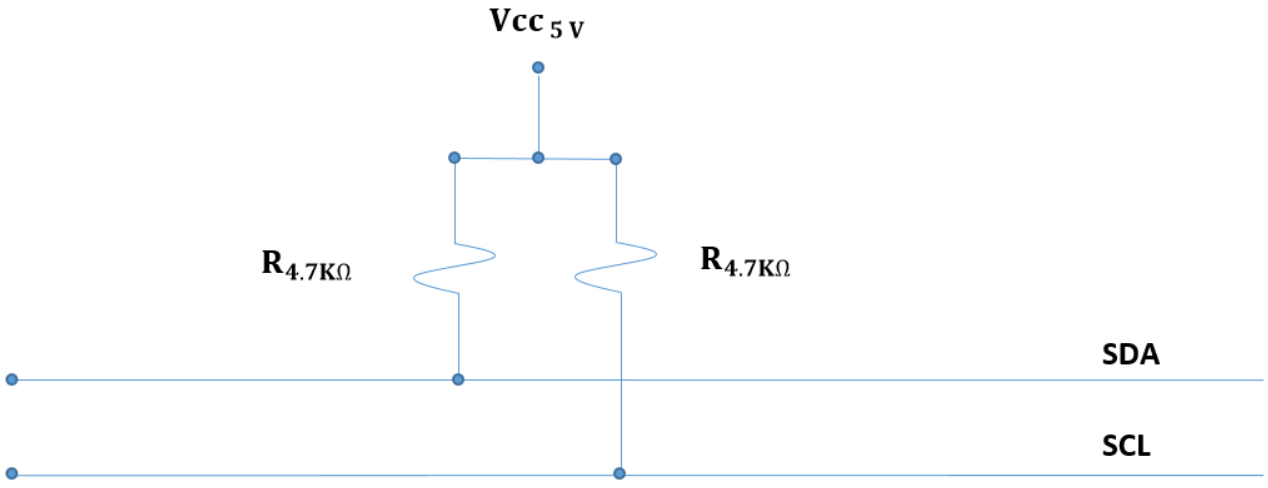

Ilustración 4 Esquema del bus I2C

Para establecer la conexión entre el microcontrolador (MASTER) y los sensores (SLAVES) se debe tener en cuenta una serie de parámetros que estructuran la comunicación I2C que se detallan a continuación.

La estructura característica de esta comunicación se muestra en la ilustración 5.

Ilustración 5 Estructura general de comunicación I2C

1. Condición de inicio (START) y fin (STOP) de la comunicación.
   - **START**: se tiene que producir una transición de nivel alto a nivel bajo del pin SDA mientras el pin SCL permanece a nivel alto.
   - **STOP**: se tiene que producir una transición de nivel bajo a nivel alto del pin SDA mientras el pin SCL permanece a nivel alto.

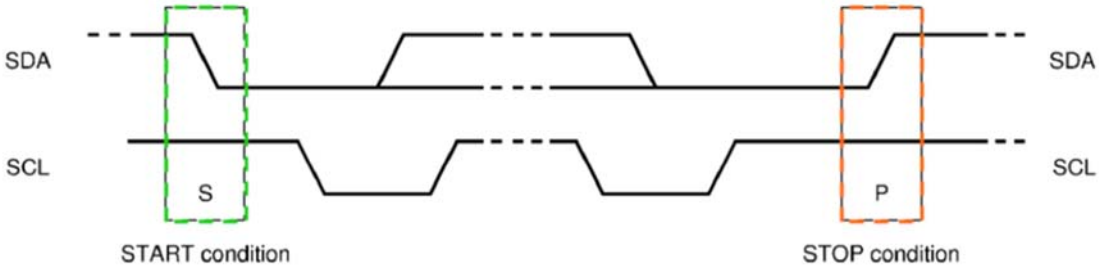

Ilustración 6 Condición START y STOP

2. Dirección de esclavo (SLAVE ADDRESS).
   - SLAVE ADDRESS: Este dato es una estructura de 7 bits que permite acceder al dispositivo que se desea. Cada sensor tiene una dirección característica y diferente a la de otros. Estas direcciones suelen ser configurables para que no coincidan.

3. Condición de lectura (READ) y escritura (WRITE).
   - READ: Para recibir del sensor, se debe poner el bit reservado para la indicación lectura/escritura a nivel alto (“1”).

Ilustración 7 Recepción I2C

   - WRITE: Para transmitir al sensor, se debe poner el bit reservado para la indicación lectura/escritura a nivel bajo (“0”).

Ilustración 8 Transmisión I2C

4. Bit de reconocimiento (ACK).
    - ACK: Este bit puede ser enviado tanto por el maestro como por el esclavo, y se utiliza para indicar si se trata del último byte de la transmisión antes de cerrar la comunicación con el STOP. Por tanto, el bit ACK debe estar a nivel alto cuando se trate del último byte de la transmisión.

6.2.2.1. CÓDIGO DE PROGRAMACIÓN

Para establecer la comunicación se han generado una serie de funciones que permiten el acceso y la recepción y transmisión de datos con los sensores que se conectan por medio de este bus. A estas funciones se le deben indicar valores para indicar la dirección de esclavo, si se trata de una recepción o de transmisión o si es el último byte de la comunicación.

Las funciones generadas son las siguientes:

- I2Cdelay(): Función que genera un retardo de 4.7 us. Esta función se utiliza para garantizar un tiempo mínimo necesario entre instrucción e instrucción en el bus. Esto garantiza una ejecución ordenada y controlada de las instrucciones y por tanto una correcta comunicación.

- I2CSendAddr(addr, rw): Esta función se encarga de inicializar la comunicación con el envío de START. Además de le debe proporcionar el "addr" que es la dirección de esclavo (SLAVE ADDRESS) para elegir el esclavo con el que se va a comunicar y el "rw" que es el bit que indica recepción o transmisión (READ/WRITE). El código asociado a esta función se muestra en la ilustración 9.

```
97  //Función que envía START + Byte de dirección del Slave (con bit LSB inicando R/W)
98  void I2CSendAddr(unsigned char addr, unsigned char rw) {
99      //CONFIGURAR PINs SDA, SCL COMO SALIDAS;  // Por si se nos olvidada en la conf. general.
100     LPC_GPIO0->FIODIR|=(1<<SDA)|(1<<SCL);
101
102     LPC_GPIO0->FIOSET|=(1<<SDA)|(1<<SCL);     // SDA y SCL a nivel alto para garantizar el
103                                               // nivel de reposo del bus + tiempo.
104     I2Cdelay();
105     LPC_GPIO0->FIOCLR=(1<<SDA);               //condicion de START: Bajar SDA y luego SCL
106     I2Cdelay();
107     LPC_GPIO0->FIOCLR=(1<<SCL);
108     I2Cdelay();
109     I2CSendByte((addr=addr<<1) + rw); //envia byte de direccion
110                                                                  //addr, direccion (7bits)
111                                                                  //rw=1, lectura
112                                                                  //rw=0, escritura
113   }
```

Ilustración 9 I2C SendAddr

- I2CSendByte(byte): Cuando se establece una transmisión (WRITE) esta función se encarga de enviar un byte al esclavo. Este "byte" es el argumento

de la función que hay que incluir dependiendo de lo que se quiera realizar. Estos datos son característicos de cada sensor y cada fabricante los especifica en su hoja de características. Por último, se configura el SDA como entrada y se genera un cicle de reloj para recibir el ACK de parte del sensor. Luego se vuelve a dejar el SDA como salida. El código asociado a esta función se muestra en la ilustración 10.

```
77  void I2CSendByte(unsigned char byte) {
78        unsigned char i;
79        for(i=0;i<8;i++){
80
81            if (byte &0x80)  LPC_GPIO0->FIOSET=(1<<SDA); // envia cada bit, comenzando por el MSB
82            else  LPC_GPIO0->FIOCLR=(1<<SDA);
83            byte = byte <<1;        // siguiente bit
84            pulso_SCL();
85           }
86
87       //Leer ACK que envía el Slave (el Master ha de enviar un pulso de reloj)
88       // CONFIGURAR PIN SDA COMO ENTRADA;          //espera ACK(config. pin como entrada)
89        LPC_GPIO0->FIODIR&=~(1<<SDA);
90        pulso_SCL();                               //Genera un pulso de reloj (1 ciclo)
91
92       // CONFIGURA PIN SDA COMO SALIDA;
93        LPC_GPIO0->FIODIR|=(1<<SDA);           // Dejamos SDA de nuevo como salida
94       }
```

Ilustración 10 I2C SendByte

- I2CGetByte(ACK): Es la función configurada para la recepción de datos (READ). Configura el pin SDA como entrada para recibir datos del sensor. Después se crea un bucle “for” para ir recibiendo los bits, de mayor a menor peso, del dato completo transmitido. Por último se configura el SDA como salida para enviar el ACK, indicando si es o no el último byte de la recepción. A esta función se le debe pasar el ACK como argumento por programa. El código asociado a esta función se muestra en la ilustración 11.

```
// Función para leer un Byte del Slave. El Master envía al final de la lectura
// el bit ACK o NACK (si es último byte leído) que se pasa como argumento de la función.
unsigned char I2CGetByte(unsigned char ACK) {
    // ACK = 0, para cualquier byte que no sea el ultimo.
    // ACK = 1 (NACK), despues de leer el ultimo byte
    unsigned char i, byte;
    //CONFIGURAR PIN SDA COMO ENTRADA; //configura pin SDA como entrada
    LPC_GPIO0->FIODIR&=~(1<<SDA);
    for(i=0;i<8;i++){                  //lee un bit comenzando por el MSB

             LPC_GPIO0->FIOSET=(1<<SCL);//mientras SCL=1
             I2Cdelay();
             byte=byte<<1;
             if(LPC_GPIO0->FIOPIN&(1<<SDA)) byte++;         //Si leemos "1" sumamos para introducir el "1"
             LPC_GPIO0->FIOCLR=(1<<SCL);                    //Si leemos "0" solo desplazamos (se introduce un "0")
             I2Cdelay();
            }

//CONFIGURAR PIN SDA COMO SALIDA;    // Master envía un ACK por cada byte leído.
LPC_GPIO0->FIODIR|=(1<<SDA);

    if(ACK)LPC_GPIO0->FIOSET=(1<<SDA);   // ACK o (NACK) es funcion del último byte leído
       else LPC_GPIO0->FIOCLR=(1<<SDA);

    pulso_SCL();                         // Pulso de reloj para su envío
    return (byte);
   }
```

Ilustración 11 I2C GetByte

- I2CSendStop(): Es la función creada para mandar el STOP y finalizar el periodo de comunicación entre maestro y esclavo. Para ello fuerza el SCL a nivel alto y posteriormente el SDA. Estos pasos se separan con la función I2Cdelay() para garantizar el correcto funcionamiento de la comunicación. Con este proceso se finaliza la comunicación. El código asociado a esta función se muestra en la ilustración 12.

```
void I2CSendStop(void) {
    LPC_GPIO0->FIOCLR=(1<<SDA);
    I2Cdelay();
    LPC_GPIO0->FIOSET=(1<<SCL);          // Subir SCL, y después SDA!! para dejar el bus en reposo
    I2Cdelay();
    LPC_GPIO0->FIOSET=(1<<SDA);
    I2Cdelay();
    }
```

Ilustración 12 I2C SendStop

6.2.3. UART

Este recurso es utilizado para establecer una comunicación serie entre el microcontrolador y el PC. Esto permite enviar o recibir información entre ambos dispositivos.

En este proyecto se utiliza sólo de manera unidireccional para enviar el conjunto de datos recogidos de los sensores por el microcontrolador hacia el PC.

6.2.3.1. CÓDIGO DE PROGRAMACIÓN

Para configurar los envíos se han generado varias funciones que permiten configurar la UART0.

La primera de ellas es el manejador de la UART0, UART0_IRQHandler(). Esta función se divide en dos posibles casos que nos indica el registro LPC_UART0->IIR de la Mini-DK2 en sus bits 3-1. Los dos posibles casos que nos podemos encontrar son que la Mini-DK2 reciba datos o que los transmita.

En caso de recepción el registro IRR en los bits 3-1 toma el valor 4. En este caso el puntero ptr_rx debe tomar el valor del registro LPC_UART0->RBR que es el que contiene el dato recibido.

En caso de transmisión el registro IRR en los bits 3-1 toma el valor 2. En este caso el LPC_UART0->THR toma el valor del puntero ptr_tx que previamente ha tenido que cargar el dato que se quiere transmitir.

El código de programación asociado a lo explicado anteriormente se muestra en la ilustración 13.

```
17  void UART0_IRQHandler(void) {
18
19      switch(LPC_UART0->IIR&0x0E) {
20
21    case 0x04:                      //RBR, Receiver Buffer Ready
22    *ptr_rx=LPC_UART0->RBR;         //Lee el dato recibido y lo almacena
23        if (*ptr_rx++ ==13)         //Caracter return --> Cadena completa
24                  {
25                *ptr_rx=0;          //Añadimos el caracter null para tratar los datos recibidos como una cadena
26                rx_completa = 1;    //rx completa
27                ptr_rx=buffer;      //Puntero al inicio del buffer para nueva recepción
28                  }
29      break;
30
31
32     case 0x02:                                 //THRE, Transmit Holding Register empty
33      if (*ptr_tx!=0) LPC_UART0->THR=*ptr_tx++; //Carga un nuevo dato para ser transmitido
34      else tx_completa=1;
35      break;
36
37      }
38  }
```

Ilustración 13 Manejador de la UART0

También se ha generado la función tx_cadena_UART0(char*cadena). Esta función, a la que se le debe pasar el argumento “cadena” que es el dato que se va a transmitir, carga en el puntero ptr_tx el valor de la cadena y posteriormente el registro LPC_UART0->THR accede al contenido del puntero para preparar el sistema para la transmisión.

El código asociado a esta función se muestra en la ilustración 14.

```
43  void tx_cadena_UART0(char *cadena)
44  {
45  ptr_tx=cadena;
46  tx_completa=0;
47  LPC_UART0->THR=*ptr_tx++; //IMPORTANTE: Introducir un carácter al comienzo para iniciar TX o
48  }                         //activar flag interrupción por registro transmisor vacio
```

Ilustración 14 Transmisión de datos UART0

6.3. DIAGRAMA DE BLOQUES DEL SISTEMA

En el sistema que se va a diseñar se van a conectar 4 sensores para la toma de datos. Los sensores de luz, temperatura ambiente y actividad se van a conectar a las líneas SDA y SCL del bus I2C mientras que el sensor de temperatura ambiente se conecta al puerto P0.23 de la Mini-DK2, que se configura como el canal 0 del ADC. Los datos de los sensores procesados por el microcontrolador se envían al PC por el puerto COM3 a través de la UART0 y se reciben en MATLAB® para representarlos en gráficas en tiempo real. El diagrama de bloques del sistema se representa en la ilustración 15.

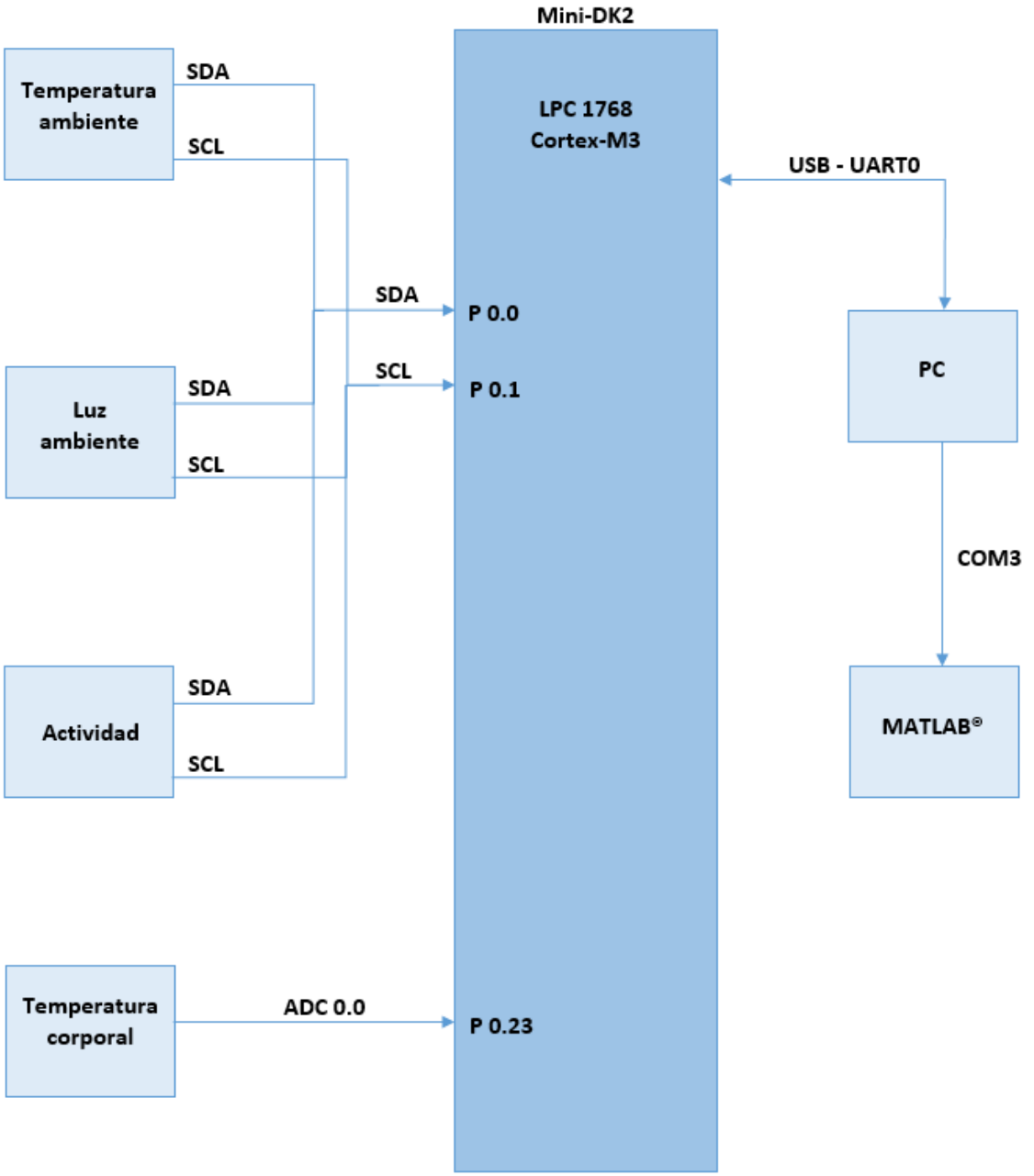

Ilustración 15 Diagrama de bloques de las conexiones del sistema

6.4. OBTENCIÓN DE DATOS

6.4.1. TEMPERATURA CORPORAL

Para la obtención de datos de la temperatura corporal se ha elegido un sensor de temperatura analógico PT100 de la marca “Sensor Solutions”, en concreto el NB-PTCO-162.

Principalmente la elección se ha basado en las pequeñas dimensiones que tiene, ya que para implantarlo en alguna zona de la piel no plantea dificultades y sin llegar a ser incómodo para la persona portadora. Además en una posible agrupación de todos los sensores puede resultar beneficioso el pequeño tamaño de los mismos.

Las dimensiones detalladas en del sensor aportadas por el fabricante son las mostradas en la ilustración 16.

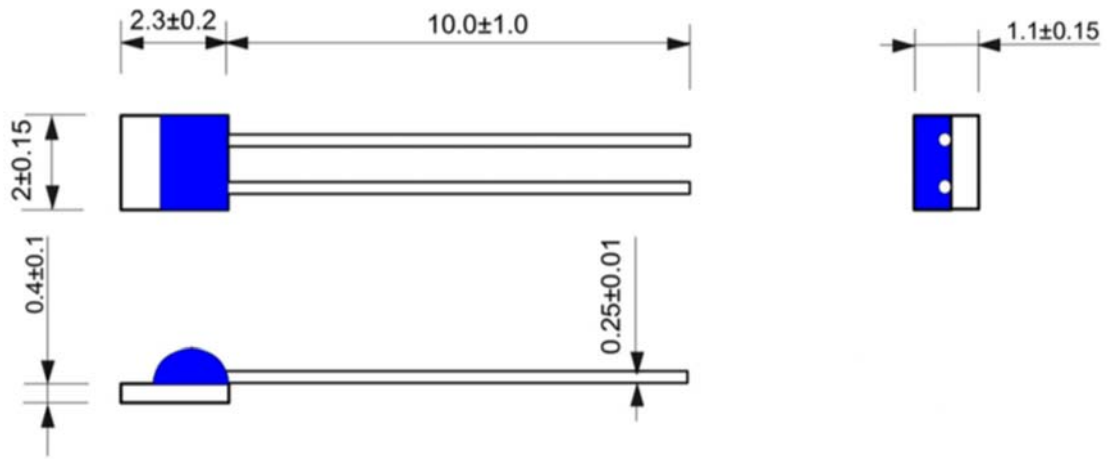

Ilustración 16 Dimensiones en mm del sensor de temperatura corporal [6]

El funcionamiento del sensor depende de una resistencia interna que varía en función de la temperatura a la que está expuesta la superficie del sensor. De este modo, cuando la temperatura a la que está el sensor aumenta, la resistencia interna también eleva su valor, por lo que podemos decir que la resistencia y la temperatura tienen una relación directa.

La gráfica que nos aporta el fabricante de esta relación se puede ver en la ilustración 17. Hay que tener en consideración que nuestro sensor tiene un rango de medición de -30 ^{o}C a 200 ^{o}C, aunque la gráfica del fabricante muestre valores fuera de este rango.

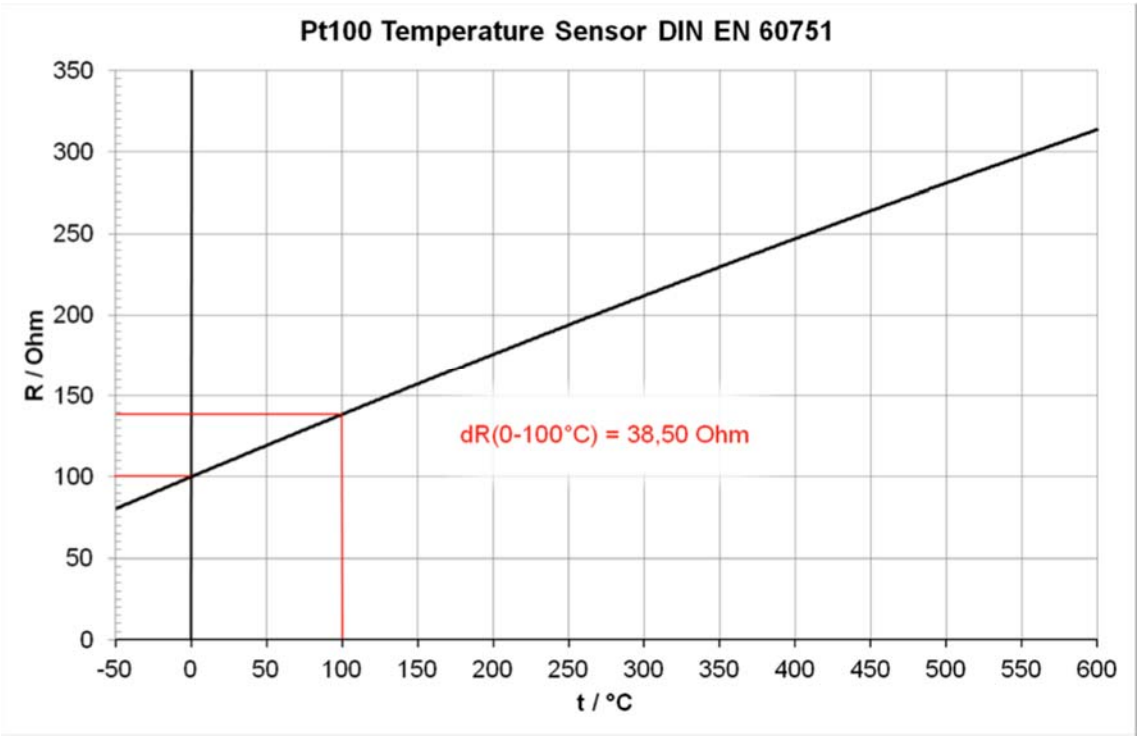

Ilustración 17 Relación resistencia/temperatura [6]

Para el conexionado del sensor se ha utilizado una resistencia auxiliar de 324 Ω que se ha conectado por un lado en serie con una patilla del sensor y por el otro lado a masa (0 voltios). A su vez, la patilla que queda libre del sensor se ha conectado a una tensión de 3.3 voltios.

La obtención de datos del sistema se hace a partir de un convertidor analógico-digital que se conecta en el punto en el que se enlazan la resistencia auxiliar y el sensor de temperatura. Este punto tiene una tensión variable, que depende de la resistencia interna del sensor, y se conecta por el puerto P0.23, que previamente ha sido configurado como el canal 0 del ADC del microcontrolador (AD0.0).

Con esta tensión de entrada al canal 0 del ADC y los cálculos pertinentes a partir de las especificaciones del fabricante además del funcionamiento básico de un circuito eléctrico, se calcula el valor de la temperatura en grados centígrados que aporta el sensor al sistema.

Las conexiones del circuito asociado a este sensor se muestran a continuación en la ilustración 18.

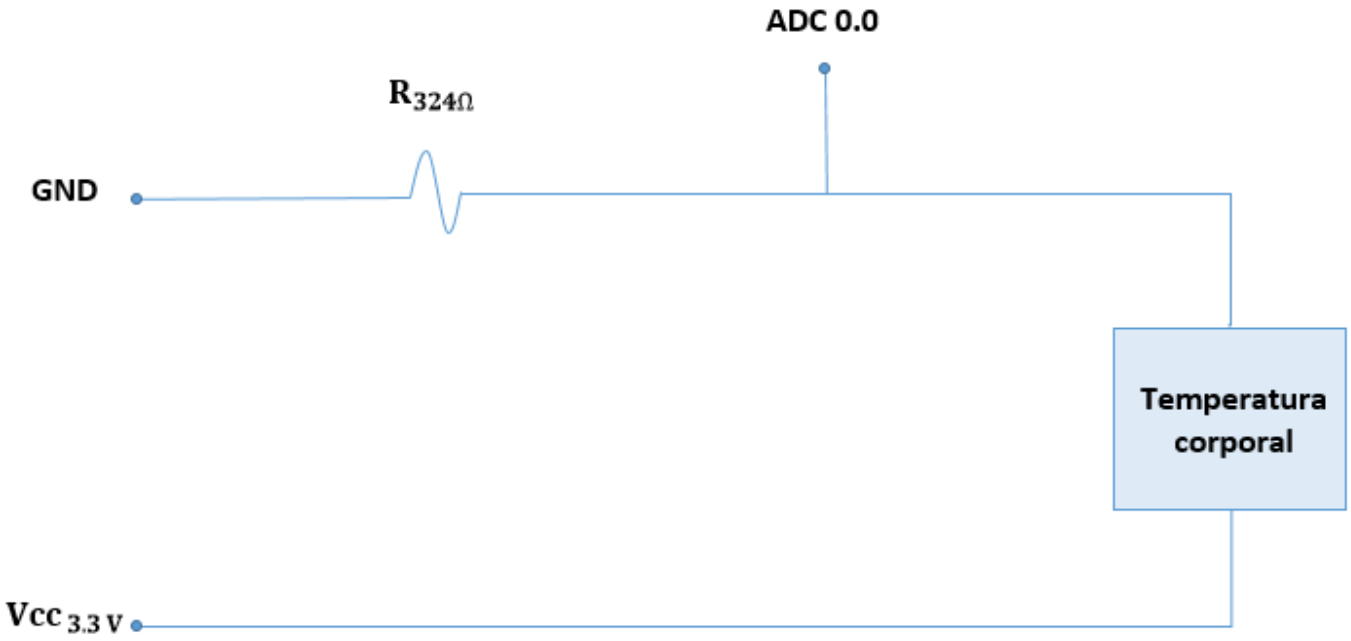

Ilustración 18 Conexión del sensor de temperatura ambiente

A continuación se muestran las ecuaciones y el proceso necesario para la obtención de la temperatura corporal en grados centígrados:

- Corriente de malla del ADC.

$$I_{ADC} = \frac{V_{ADC}}{R_{324\Omega}}$$

- Cálculo de la resistencia variable del sensor.

$$R_{PT100} = \frac{V_{R_{PT100}}}{I_{ADC}} = \frac{V_{3.3V} - V_{ADC}}{I_{ADC}}$$

$$R_{PT100} = \frac{(V_{3.3V} - V_{ADC}) * R_{324\Omega}}{V_{ADC}}$$

- Para el cálculo de la temperatura, el fabricante nos proporciona una ecuación que relaciona la temperatura a la que está expuesto el sensor, la resistencia del sensor y tres parámetros fijos [6].

$$R_{(T)} = R_{(0)} * (1 + a * T + b * T^2)$$

Con: $a = 3.9083 * 10^{-3}$

$b = -5.775 * 10^{-7}$

$R_{(0)} = 100\ \Omega$

- Se obtiene la siguiente ecuación de segundo grado.

$$b * T^2 + a * T + \left(1 - \frac{R_{PT100}}{R_{(0)}}\right)$$

- Con la fórmula de resolución de las ecuaciones de segundo grado y teniendo en cuenta el signo positivo de la raíz, se obtiene el valor de la temperatura corporal que nos da el sensor. La temperatura obtenida se representa con un decimal por decisión propia, ya que el sensor tiene una resolución superior.

$$T = \frac{-a + \sqrt{a^2 - \left(4 * a * \left(1 - \frac{R_{PT100}}{R_{(0)}}\right)\right)}}{2 * b}$$

6.4.1.1. CÓDIGO DE PROGRAMACIÓN

En primer lugar se crea la función config_ADC() donde se configuran los parámetros deseados del ADC. Se activa el modo burst que convierte el canal 0 con una frecuencia de conversión de 1.5 KHz. La parte del código asociada a esta función se puede ver en la ilustración 19.

```
211 void config_ADC(void){
212     LPC_SC->PCONP|= (1<<12);                        // Power ON
213     LPC_PINCON->PINMODE1|= (2<<14);                 // Deshabilita pullup/pulldown
214     LPC_SC->PCLKSEL0|= (0x00<<8);                   // CCLK/4 (Fpclk después del reset) (100 Mhz/4 = 25Mhz)
215     LPC_ADC->ADCR|=(1<<0)                           // canal 0
216                                 | (0x0FF<<8)        // CLKDIV velocidad minima
217                                 | (0x01<<21)        // PDN=1
218                                 | (1<<16);          // burst=1
219 }
```

Ilustración 19 Configuración del ADC

Una vez configurado el ADC, se crea una función para el cálculo del dato de temperatura, donde se incluyen las fórmulas del apartado anterior. Esta función recibe el nombre de leer_temp_corporal() y el código se muestra en la ilustración 20.

```
void leer_temp_corporal(){

  //Prueba con fórmula del fabricante
  temp_corporal_volt=((LPC_ADC->ADGDR>>4)&0xFFF)*tension_alimentacion/4095;                    //Configuración del rango del ADC
  resistencia_sensor=((((tension_alimentacion-temp_corporal_volt)*324)/temp_corporal_volt)-2); //Calculo de la resistencia variable
  primera_parte_raiz=pow(a,2);
  segunda_parte_raiz=(4*b*(1-(resistencia_sensor/100)));
  total_raiz=primera_parte_raiz-segunda_parte_raiz;                  //Calculo de la ecuacion de segundo grado
  raiz=sqrt(total_raiz);
  numerador=-a+raiz;
  denominador=2*b;
  temp_corporal_grados1=numerador/denominador;                       //Calculo de la temperatura corporal
```

Ilustración 20 Temperatura corporal

Para mejorar la estabilidad del sensor a la hora de mostrar los datos se ha generado un filtro en el código de programación para eliminar picos e inestabilidades que en ocasiones produce el sensor y su conexionado y afectan a la respuesta y los datos aportados. El código asociado a este filtro se muestra en la ilustración 21.

```
  if(temp_corporal_grados1>50||temp_corporal_grados1<0){
    temp_corporal_grados1=33;                                        //Regresión por dato fuera de rango
    salto_temp_corporal=temp_corporal_grados1-temp_corporal_grados0;  //Condición activación de filtro

    if(salto_temp_corporal>10 || salto_temp_corporal<-10){           //Filtro para gráficas
      temp_corporal_grados1=temp_corporal_grados0;
    }
    temp_corporal_grados0=temp_corporal_grados1;                     //Guardar temperatura instante anterior
  }
}
```

Ilustración 21 Filtro de estabilidad sensor de temperatura corporal

6.4.2. LUZ AMBIENTE

Para el seguimiento y el estudio de este parámetro se ha escogido el sensor BH1750FVI. Este sensor suministra el dato de luminosidad ambiente a la que está expuesto en un formato digital de 16 bits. La comunicación con el microcontrolador se establece por la interfaz I2C. A continuación se muestra en la ilustración 22 el esquema de conexión de este sensor [7].

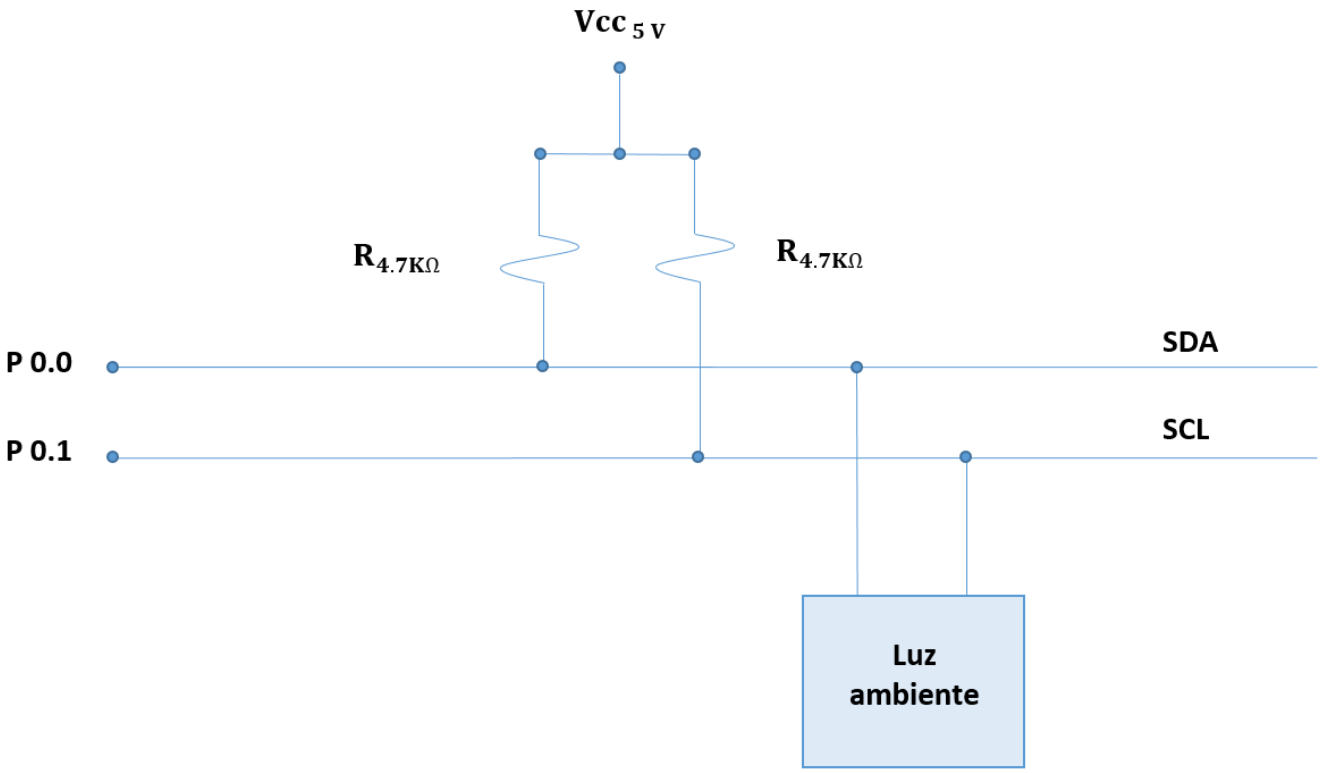

Ilustración 22 Conexión del sensor de luz ambiente

Además de los dos pines SDA y SCL para la conexión con el bus I2C, el sensor dispone de tres pines más para conectar la alimentación y configurar la dirección que utiliza el I2C para establecer la conexión y compartir la información. Se conecta el pin Vcc al punto del circuito con tensión de 5V ($Vcc_{5\,V}$), y el ADD y GND al punto del circuito con tensión de 0V (GND). Al conectar el pin ADD a 0V la dirección del sensor en hexadecimal se establece en el valor 0x23. En la ilustración 23 se muestra una foto del sensor real donde se pueden ver los pines y las conexiones.

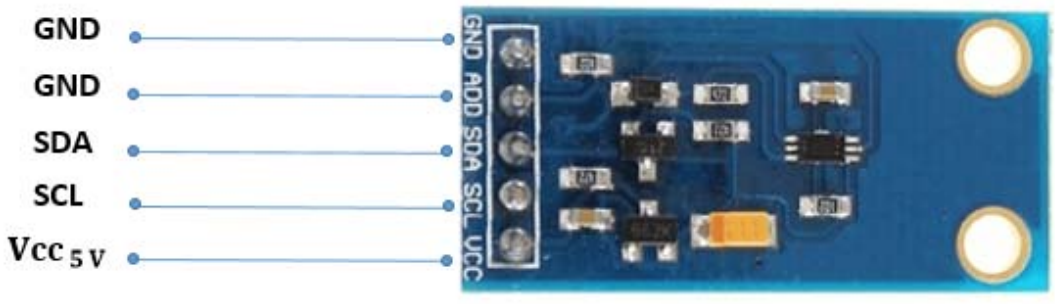

Ilustración 23 Sensor de luz ambiente

El BH1750FVI dispone de un amplio rango de medición. El dato proporcionado en luxes (unidad del sistema internacional destinada a la iluminancia), puede alcanzar valores

desde 0.11 a 65535. Este máximo lo establece un ADC de 16 bits que incorpora el sensor internamente. El diagrama de bloques interno del sensor lo proporciona el fabricante en la hoja de datos y se muestra en la ilustración 24.

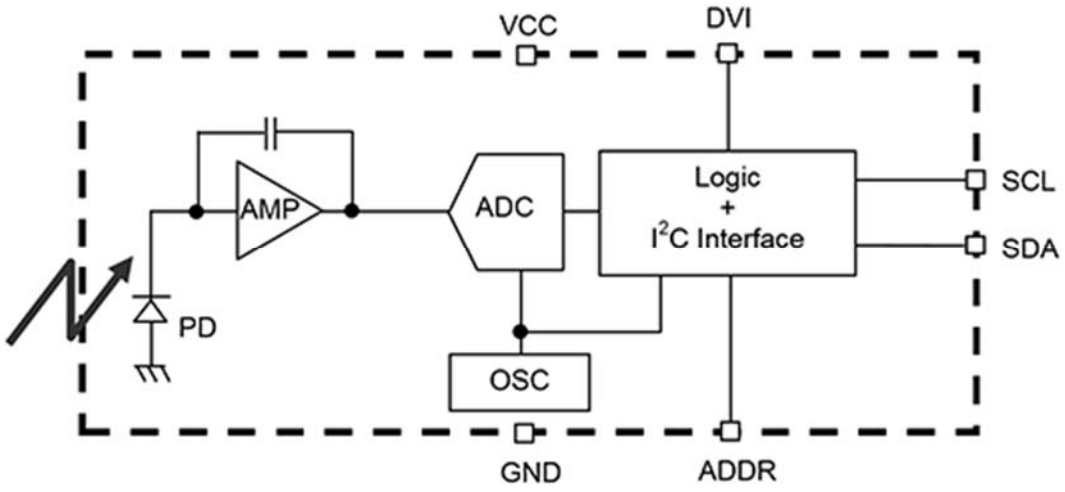

Ilustración 24 Diagrama de bloques del sensor de luz ambiente [7]

6.4.2.1. CÓDIGO DE PROGRAMACIÓN

Para la programación de este sensor se crean dos funciones, la de inicialización del sensor y la de obtención del dato de medición.

La función que inicializa el sensor se ha nombrado config_luz() y es la encargada de encender el sensor. Para ello, se envía por I2C el código de activación que aporta el fabricante ("Power On"). El código asociado a esta función se muestra en la ilustración 25.

```
199 void config_luz() {
200     I2CSendAddr(0x23,0);        //inicio comunicación de escritura
201     I2CSendByte(0x01);          //byte para encender el sensor
202     I2CSendStop();              //paro la comunicación
203     I2Cdelay();
204 }
```

Ilustración 25 Inicialización del sensor de luz ambiente

Para la obtención del dato de iluminación se ha creado la función leer_luz_ambiente(). En ella se configura el modo de conversión continuo con una resolución de 1 lux y un periodo de muestreo de 120ms (Continuously H-Resolution Mode). Después se recibe el dato del sensor en dos bytes y se almacena en la variable "luz_ambiente". Por último, se obtiene el dato real de iluminación dividiendo "luz_ambiente" entre 1.2 y se

almacena en la variable “iluminación”. El código asociado a esta función se muestra en la ilustración 26.

```
void leer_luz_ambiente(){
    I2CSendAddr(0x23,0);                        //START + Byte. Inicia la comunicacion de escritura del dispositivo
    I2CSendByte(0x10);                          //Acceso a la lectura "Continuosly H-Resolution Mode"
    I2CSendAddr(0x23,1);                        //Re-start para la lectura del dato
    luz_ambiente=I2CGetByte(0)<<8;              //guardo el byte de mayor peso
    luz_ambiente|=I2CGetByte(1);                //guardo el byte de menos peso
    I2CSendStop();                              //paro comunicación

    iluminacion=((float)luz_ambiente/precision_medida_luz);      //Obtencion dato real de iluminacion
}
```

Ilustración 26 Obtención del dato de iluminación

6.4.3. TEMPERATURA AMBIENTE

Para la configuración del termómetro del sistema, encargado de proporcionar el dato de temperatura ambiente a la que está expuesto el sistema, se ha elegido el sensor de temperatura DS1621. El sensor tiene una precisión de 0.5^{o}C y un rango de medición de -55^{o}C a +125^{o}C, rango más que suficiente para lo que se desea medir. El dato viene dado en dos bytes de los cuales tienen validez para la interpretación del dato los 9 bits de mayor peso. El MSB indica la parte entera del valor de la temperatura ambiente, y el bit de mayor peso del LSB es el encargado de indicar la precisión de 0.5^{o}C, ya que en caso de que este tenga el valor “1”, la parte decimal del dato de la temperatura ambiente será 0.5^{o}C [8].

La conexión del sensor para la transmisión de datos se hace por I2C y el esquema de conexión se muestra en la ilustración 27.

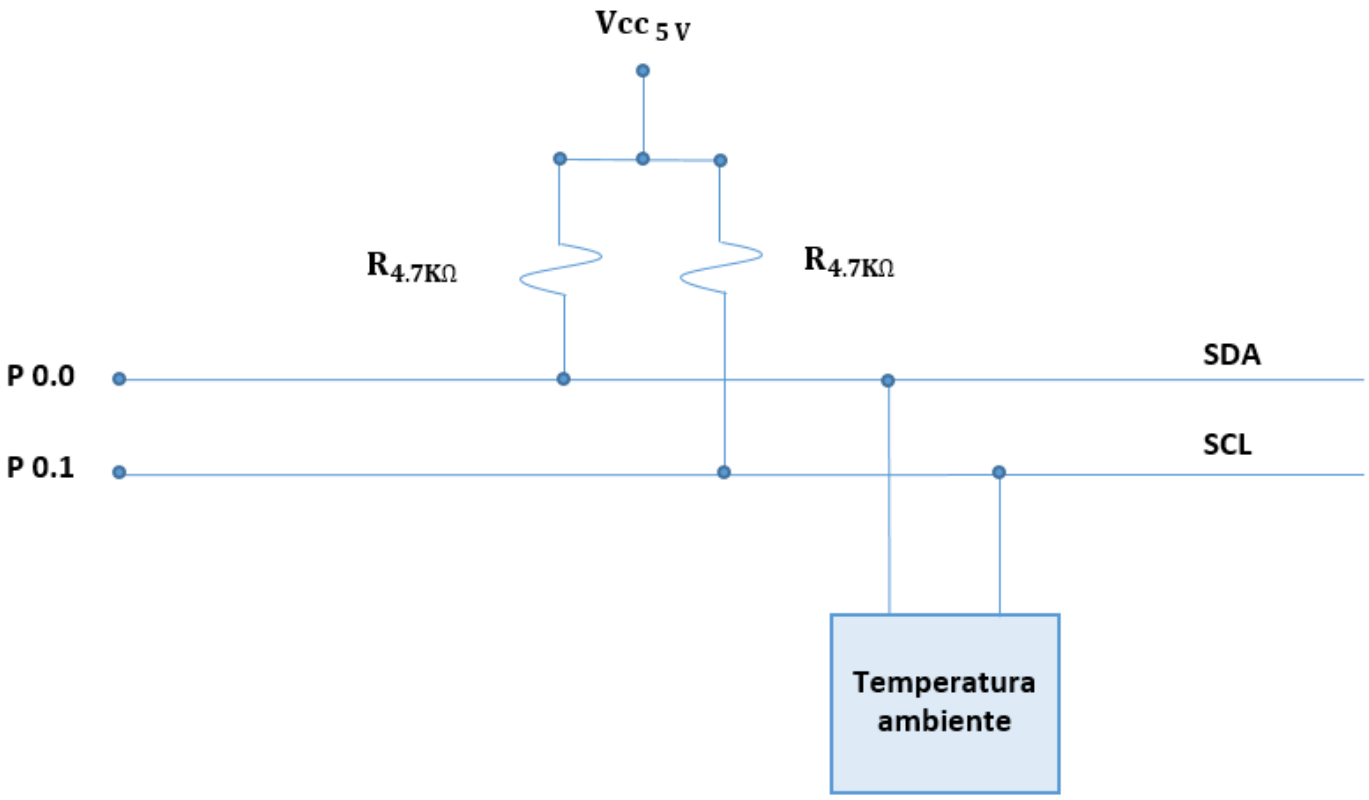

Ilustración 27 Conexión del sensor de temperatura ambiente

El DS1621 dispone de 8 pines externos para la conexión. Dos de ellos son el SDA y SCL, necesarios para la conexión I2C como bien sabemos de apartados anteriores. También dispone de dos pines de alimentación, un pin "V_{DD}" conectado a 5 voltios y otro "GND" conectado a 0 voltios. Además de estos 4, los pines "A_0, A_1 y A_2" son los encargados de configurar la dirección del esclavo en el bus I2C. Se han puenteado y se han conectado los tres a una tensión de 0 voltios, lo que asigna al esclavo la dirección 0x48. Por último, el pin que nos falta describir es el "T_{OUT}" que es la señal de salida del termostato. En nuestro caso este pin no se utiliza y se ha dejado al aire [8]. El esquema de conexiones se adjunta en la ilustración 28.

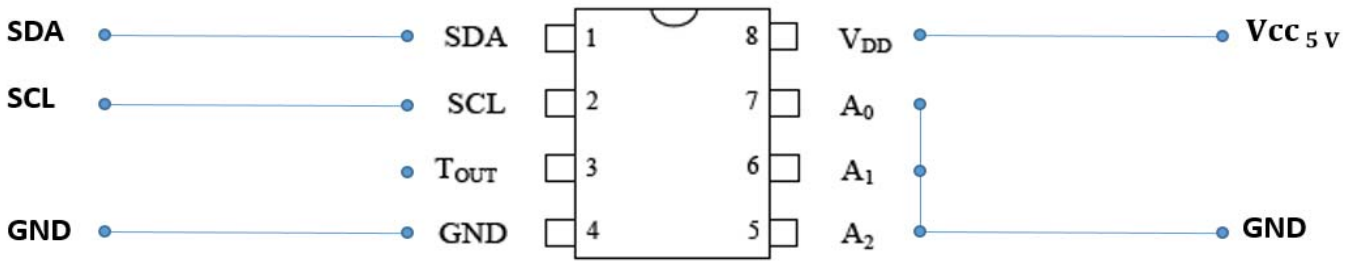

Ilustración 28 Sensor de temperatura ambiente

6.4.3.1. CÓDIGO DE PROGRAMACIÓN

Para la configuración de este sensor se ha creado la función config_temp_ambiente(). En ella se establecen dos intercambios de datos que son dos transmisiones. En la primera se envía al sensor el código 0xAC que sirve para activar el acceso de lectura o escritura para poder configurar registros del sensor. En la segunda transmisión se envía el código 0xEE para iniciar la conversión continua de la temperatura ambiente. Por indicaciones del fabricante en su hoja de datos, el tiempo máximo de esta conversión es de 750 ms. El código de programación asociado a esta función se muestra en la ilustración 29.

```
157 void config_temp_ambiente() {
158     I2CSendAddr(0x48,0);          //Inicio comunicación de escritura
159     I2CSendByte(0xAC);            //Configuracion del acceso R/W
160     I2CSendStop();                //Paro la comunicación
161     I2Cdelay();
162     I2CSendAddr(0x48,0);          //Inicio comunicación de escritura
163     I2CSendByte(0xEE);            //Inicio de la conversion continua
164     I2CSendStop();                //Paro la comunicación
165 }
```

Ilustración 29 Inicialización del sensor de temperatura ambiente

Para la obtención del dato de temperatura ambiente se ha generado la función leer_temp_ambiente(). En esta función después de establecer la conexión con el sensor para la lectura de datos con el código 0xAA, recibe el dato en dos pasos, primero recibiendo el MSB y después el LSB. Una vez guardados ambos datos, se almacena el dato MSB como float en la variable temp_ambiente y después se comprueba el valor del bit da mayor del LSB. Si este valor es “1”, se le suma a la variable temp_ambiente 0.5. En caso de que sea “0” no se hace ningún cambio. El código de programación asociado a esta función se muestra en la ilustración 30.

```
224 void leer_temp_ambiente() {
225     I2CSendAddr(0x48,0);                    //Inicia la comunicacion de escritura del dispositivo
226     I2CSendByte(0xAA);                      //Acceso a la lectura de la temperatura
227     I2CSendAddr(0x48,1);                    //Re-start para la lectura del dato
228     temp_ambiente_MSB=I2CGetByte(0);        //guardo el byte de mayor peso
229     temp_ambiente_LSB=I2CGetByte(1);        //guardo el byte de menos peso
230     I2CSendStop();                          //paro comunicación
231     temp_ambiente=(float) temp_ambiente_MSB;//paso a formato float el dato para añadir precisión de 0.5 °C
232     if((temp_ambiente_LSB>>7)&&1==1)        //si el byte de menor peso tiene un 1 en su bit de mayor peso
233         temp_ambiente=temp_ambiente+0.5;    //le sumo 0.5 °C*
234
235   }
```

Ilustración 30 Obtención del dato de temperatura ambiente

6.4.4. ACTIVIDAD

Para el seguimiento del parámetro de actividad física se ha elegido el acelerómetro MPU-6050. Este sensor se puede dividir en dos partes a la hora de aportar datos relativos a la posición:

- La primera es la parte de acelerómetro. Esta parte proporciona datos de aceleración, en 16 bits, en los tres ejes del espacio (X, Y, Z).
- La segunda es la parte de giroscopio. Esta parte proporciona datos de velocidad, en 16 bits, en los tres ejes del espacio (X, Y, Z).

Se ha configurado el sensor para que suministre los datos de aceleración actuando sobre los registros internos de ahorro de consumo de energía 107 y 108 del sensor. [9].

La aceleración que tiene en cuenta para el aporte de datos es la aceleración de la gravedad. Dependiendo de la orientación del sensor, esta aceleración gravitatoria se distribuye total o parcialmente en los tres ejes del espacio. La disposición de los ejes en el MPU-6050 viene establecida como se indica en la ilustración 31.

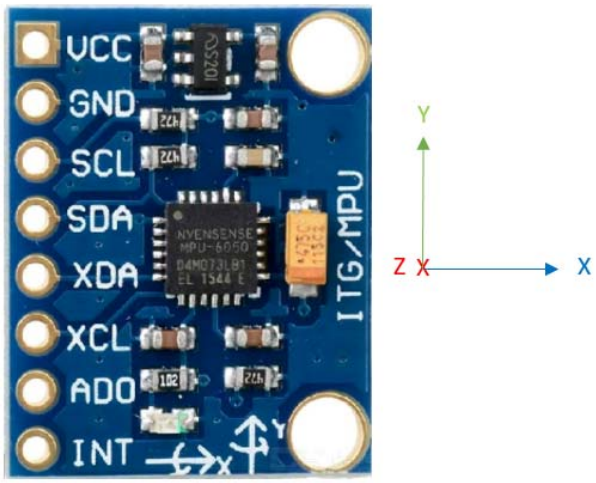

Ilustración 31 Disposición de los ejes en el sensor de actividad

La conexión del sensor para la comunicación con el microcontrolador se establece vía I2C y el enlace con el bus se representa en la ilustración 32.

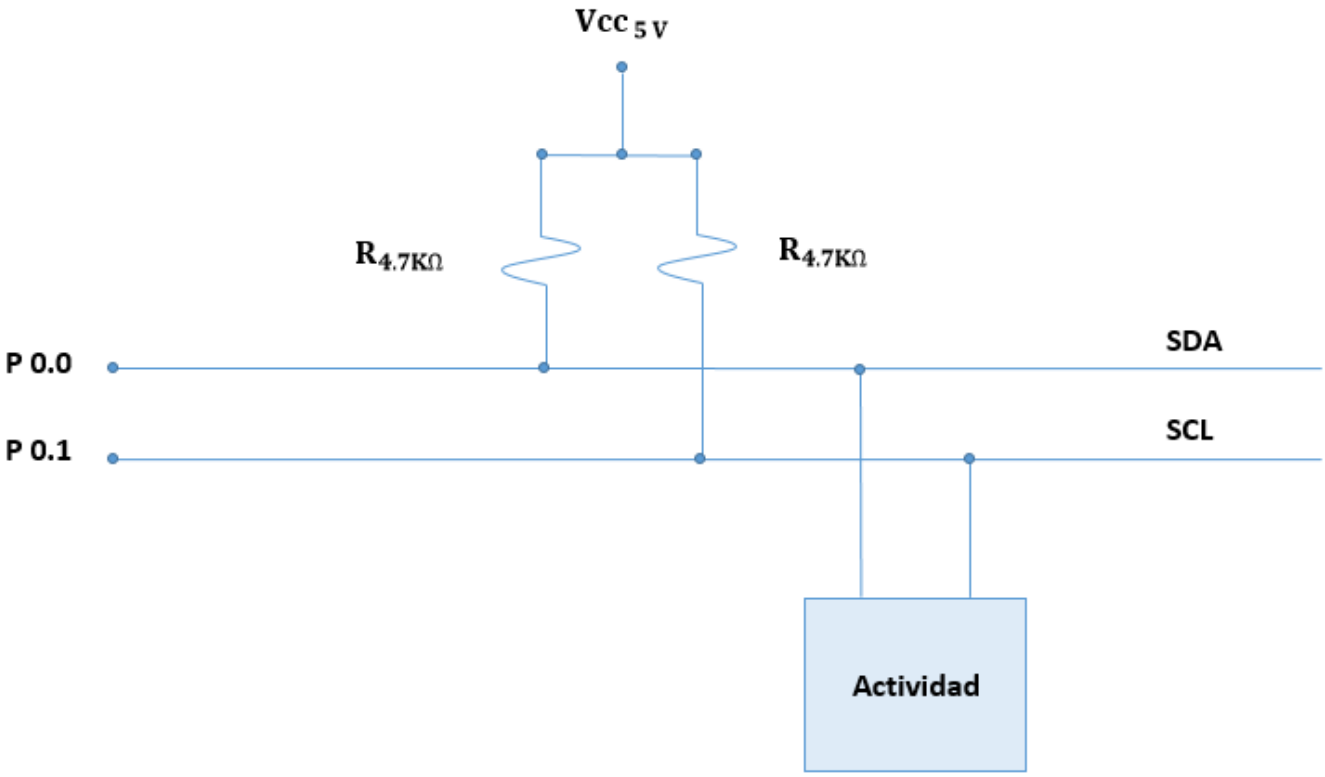

Ilustración 32 Conexión del sensor de actividad

EL MPU-6050 dispone 8 pines de para la conexión, de los cuales utilizamos 4. Estos son el VCC que se conecta a una tensión de 5 voltios, GND que se conecta a una tensión de 0 voltios, SDA y SCL que son los pines de conexión al bus del I2C. La dirección de esclavo que se utiliza para la comunicación con este sensor es la 0x68, que es la que viene predeterminada por el fabricante. Estas conexiones se muestran en la ilustración 33.

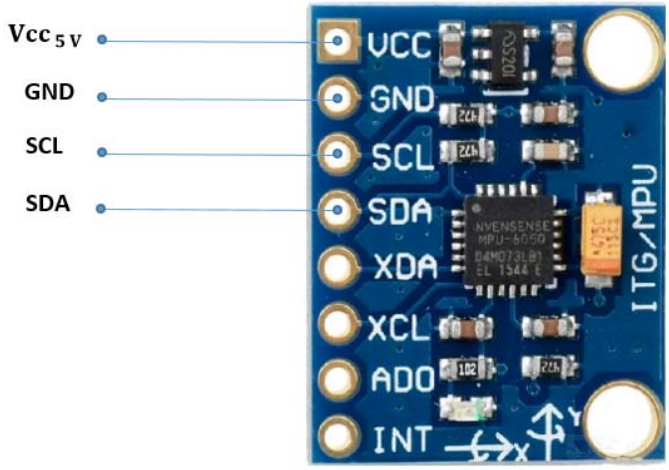

Ilustración 33 Conexión del sensor de actividad

6.4.4.1. CÓDIGO DE PROGRAMACIÓN

Para el control de este sensor y la adquisición de datos relativos a la actividad física del portador del sistema se han generado dos funciones.

La primera, nombrada como config_acelerometro(), es la que se encarga de la inicialización del MPU-6050. En ella se accede al registro 37 para establecer el periodo de muestreo del sensor en 250ms. Además de este registro, se configura el modo de ahorro de energía estableciendo los valores adecuados para los registros 107 y 108 para que el sensor solo convierta los valores correspondientes al acelerómetro. El código de programación asociado a esta función se muestra en la ilustración 34.

```
163 void config_acelerometro() {
164     I2CSendAddr(0x68,0);        //inicio comunicación de escritura para seleccionar la frecuencia de muestreo
165     I2CSendByte(0x25);          //byte de mayor peso dirección del registro
166     I2CSendByte(0xF9);          //byte de menor peso se establece frecuencia de muestreo en 4Hz (T=250ms)
167     I2CSendStop();              //paro la comunicación
168     I2Cdelay();
169
170     I2CSendAddr(0x68,0);        //Inicio comunicación de escritura
171     I2CSendByte(0x6B);          //Acceso al registro 107
172     I2CSendByte(0x20);          //deshabilito la temperatura
173     I2CSendStop();              //paro la comunicación
174     I2Cdelay();
175
176     I2CSendAddr(0x68,0);        //Inicio comunicación de escritura
177     I2CSendByte(0x6C);          //Acceso al registro 108
178     I2CSendByte(0xC0);          //representación de datos
179     I2CSendStop();              //paro la comunicación
180     I2Cdelay();
181 }
```

Ilustración 34 Inicialización acelerómetro

La segunda función generada para el manejo del sensor se ha nombrado leer_acelerometro(). En esta función se acceden a los registros donde el sensor deposita los datos de la conversión. Para la obtención de los datos en 16 bits, se debe acceder a dos registros diferentes para cada eje, ya que el sensor almacena los datos en 8 bits, por lo que se tiene que recoger el LSB y el MSB de cada eje por separado. Una vez se obtienen por separado, se juntan en una sola variable. Las variables donde se recogen los datos reciben el nombre “AccX”,” AccY” y “AccZ”.

Una vez se tienen los valores almacenados en variables independientes se realiza una operación para expresarlos en unidades de m/s^2. Se ha decidido no realizar más operaciones con estos valores y expresarlos como nos los da el acelerómetro para no distorsionar estos resultados. Algunos científicos opinan que dispositivos que tienen implantados acelerómetros pre-procesan los datos aportados por el sensor en sus códigos de programación y no muestran los datos reales, sino interpretaciones que los fabricantes realizan por programa [3]. El código asociado a la lectura de los datos se muestra en la ilustración 35.

```
void leer_acelerometro(){
  //Leer valores de acelerometro
    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x3B);                    //Acceso a la lectura del MSB del eje X
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccX=I2CGetByte(1)<<8;                //guardo el byte de mayor peso eje X
    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x3C);                    //Acceso a la lectura del LSB del eje X
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccX|=I2CGetByte(1);                  //guardo el byte de menor peso eje X

    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x3D);                    //Acceso a la lectura del MSB del eje Y
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccY=I2CGetByte(1)<<8;                //guardo el byte de mayor peso eje Y
    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x3E);                    //Acceso a la lectura del LSB del eje Y
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccY|=I2CGetByte(1);                  //guardo el byte de menor peso eje Y

    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x3F);                    //Acceso a la lectura del MSB del eje Z
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccZ=I2CGetByte(1)<<8;                //guardo el byte de mayor peso eje Z
    I2CSendAddr(0x68,0);                  //START + Byte. Inicia la comunicacion de escritura
    I2CSendByte(0x40);                    //Acceso a la lectura del LSB del eje Z
    I2CSendAddr(0x68,1);                  //Re-start para la lectura del dato
    AccZ|=I2CGetByte(1);                  //guardo el byte de menor peso eje Z

    AccX_f=(int)AccX*(9.8/15200);                   //Posición del eje X del acelerómetro
    AccY_f=(int)AccY*(9.8/15200);;                  //Posición del eje Y del acelerómetro
    AccZ_f=(int)AccZ*(9.8/15200);;                  //Posición del eje Z del acelerómetro
}
```

Ilustración 35 Obtención de datos del acelerómetro

6.5. ENVÍO DE DATOS

6.5.1. CONEXIÓN MICROCONTROLADOR - PC

Para el enlace entre el microcontrolador que maneja los sensores y el ordenador para la transmisión de los datos obtenidos se utiliza UART0. Ambos se conectan mediante un cable USB. La transmisión de los datos obtenidos por el sistema se realiza a través del COM3, de manera que accediendo a este puerto desde cualquier programa/aplicación que lo permita se pueden ver y realizar operaciones con ellos.

Los datos se envían en diferentes formatos, dependiendo de lo que cada sensor requiera, y separados por tabulaciones. Después del último dato se incluye el carácter "Enter" para introducir un salto de línea. Por este motivo, la transmisión se hace por líneas consecutivas, donde cada una de las líneas se corresponde con un envío nuevo. Esto hace que los datos se reciban de una manera estructurada y ordenada en el tiempo.

La correspondencia entre las posiciones del array de la transmisión y los datos trasmitidos es la siguiente:

- Primera posición: Temperatura ambiente con 2 dígitos para la parte entera y 1 para la parte decimal.
- Segunda posición: Temperatura ambiente con 2 dígitos para la parte entera y 1 para la parte decimal.
- Tercera posición: Iluminación con 5 dígitos para la parte entera y sin parte decimal.
- Cuarta posición: Aceleración en el eje X con 2 dígitos para la parte entera y 1 para la parte decimal.
- Quinta posición: Aceleración en el eje Y con 2 dígitos para la parte entera y 1 para la parte decimal.
- Sexta posición: Aceleración en el eje Z con 2 dígitos para la parte entera y 1 para la parte decimal.

Las dos aplicaciones que se han utilizado para el seguimiento de la transmisión de los datos son:

- Termite: Es un terminal al que se envían los datos continuamente con la frecuencia de muestreo seleccionable en el programa que se vuelca al microcontrolador. Los datos estructurados como se ha explicado anteriormente van apareciendo indefinidamente en la pantalla del terminal.
- MATLAB®: En este programa se han generado una serie de script que nos permiten representar los parámetros en tiempo real. En las gráficas se puede ver

la evolución y los cambios de las respuestas de cada sensor dependiendo de los factores que influyen a cada uno de ellos.

En la ilustración 36 se muestra la pantalla “Command Window” de MATLAB® donde se puede ver cómo llega la trama de datos para la representación.

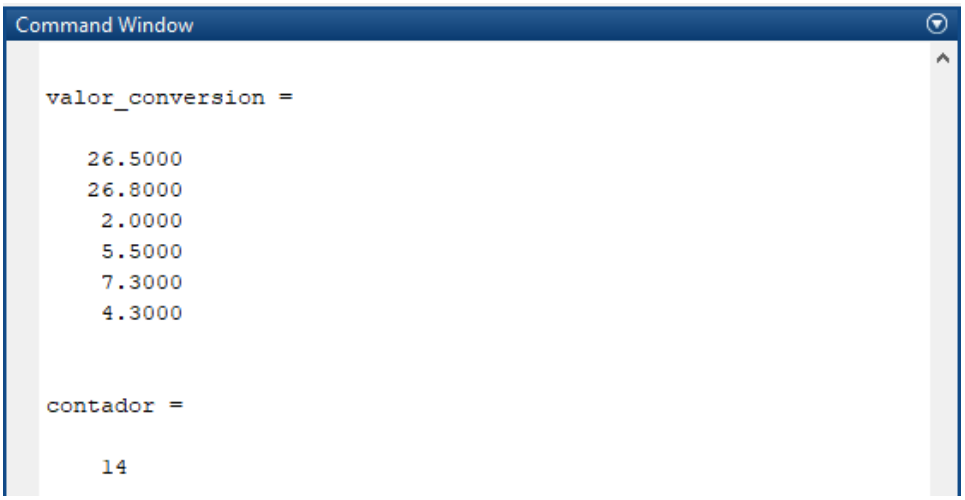

Ilustración 36 Trama de datos en MATLAB®

6.5.1.1. CÓDIGO DE PROGRAMACIÓN

En primer lugar, para la configuración y el manejo de la UART se adjunta una librería al proyecto llamada uart.c. En este fichero se crean funciones para la inicialización, transmisión y envío de cadenas de texto, etc.

Una vez inicializada la UART, se crea en el programa principal una función llamada uart_mostrar_datos() que es la encargada de mandar los datos al COM3. En ella lo primero que se hace es inicializar el puntero ptr_rx apuntando al buffer que almacena los datos para la transmisión.

Después se inicializa la velocidad de transmisión de la UART con la función uart0_init(baudios) incluida en la librería uart.c. La velocidad se establece a 9600 baudios.

Por último se van enviando los datos aportados por los sensores con la estructura creada para la transmisión de datos. El código asociado a esta función se puede ver en la ilustración 37.

```
302 void uart_mostrar_datos(){
303
304     ptr_rx=buffer;     // inicializa el puntero de recepción al comienzo del buffer
305     uart0_init(9600);  // configura la UART0 a 9600 baudios, 8 bits, 1 bit stop
306
307     sprintf(buffer, "%2.1f \t",temp_ambiente);
308     tx_cadena_UART0(buffer); //transmito esa cadena recien guardada
309     while(tx_completa==0);
310     tx_completa=0;
311
312     sprintf(buffer," %2.1f \t",temp_corporal_grados1);
313     tx_cadena_UART0(buffer); //transmito esa cadena recien guardada
314     while(tx_completa==0);
315     tx_completa=0;
316
317     sprintf(buffer, " %5.0f \t",iluminacion);
318     tx_cadena_UART0(buffer); //transmito esa cadena recien guardada
319     while(tx_completa==0);
320     tx_completa=0;
321
322     sprintf(buffer," %2.1f \t %2.1f \t %2.1f \t",AccX_f,AccY_f,AccZ_f);
323     tx_cadena_UART0(buffer); //transmito esa cadena recien guardada
324     while(tx_completa==0);
325     tx_completa=0;
326
327 }
```

Ilustración 37 Transmisión de datos UART0

Una vez configurada la función encargada de transmitir los datos se debe establecer la frecuencia de transmisión de datos con la que van a enviarse los datos al COM3. Para ello se configura el Systick como timer para actuar como reloj del sistema para el aporte de datos al PC.

El sistema está configurado para funcionar a frecuencia interna de 100 MHz. En la ilustración 38 se puede ver los posibles relojes del microcontrolador y el que se ha seleccionado para la aplicación.

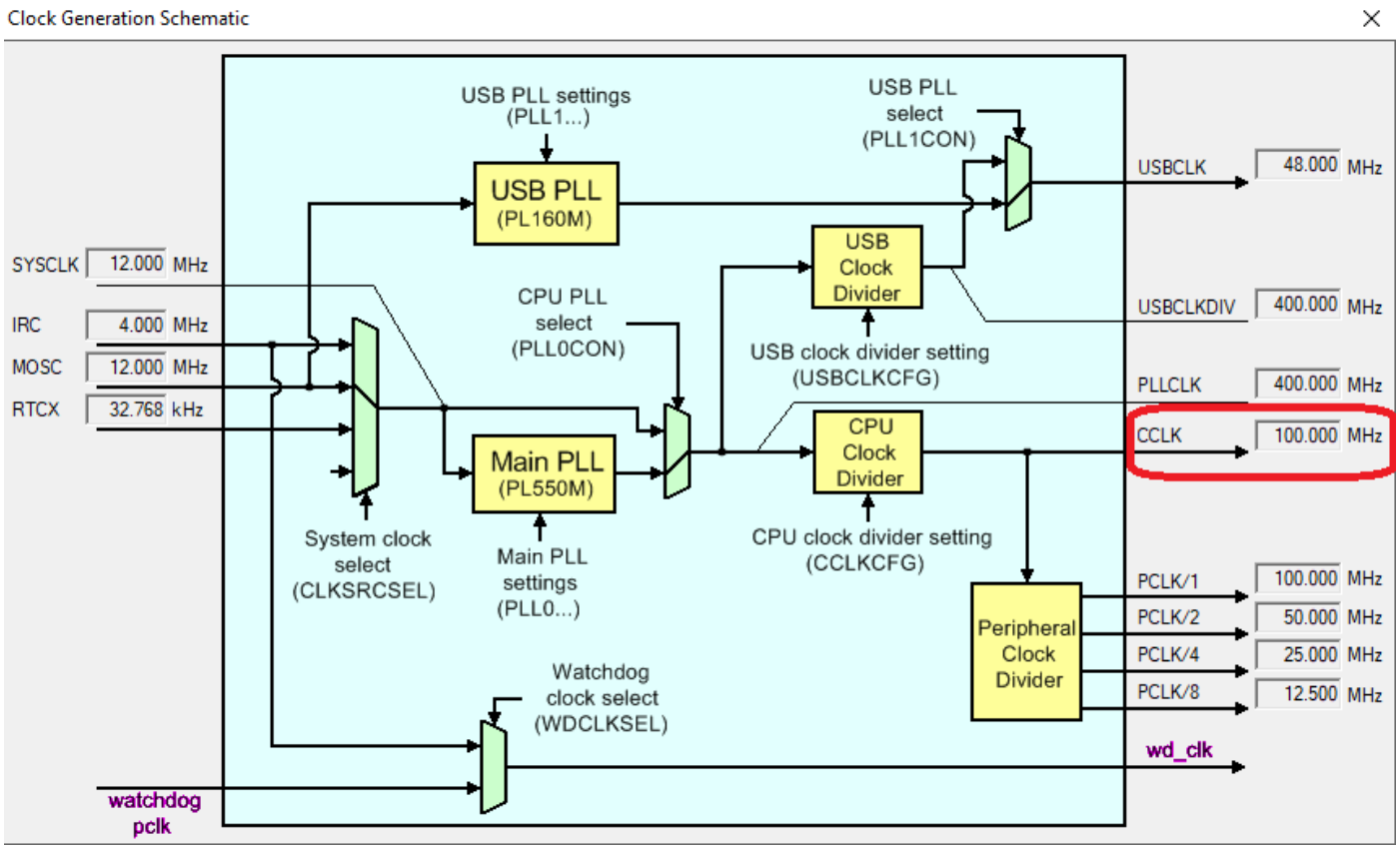

Ilustración 38 Relojes del microcontrolador

Por esto, en el programa principal, a la función SysTick_Config() se le ha pasado como argumento el valor 100000. Esto hace que el tiempo de interrupción del SysTick se establezca en 1 ms. Por tanto, al utilizar este timer para mandar los datos por el COM3, el tiempo mínimo para mandar los datos también se establece en 1 ms.

Una vez configurado el tiempo de interrupción del SysTick, se crea el código asociado a su manejador denominado SysTic_Handler(). En esta función se encarga de la medición de todos los sensores en la temporalidad que se desee y la activación de un flag para el envío de datos por la UART.

Para ello se inicia la función con una variable utilizada como contador que se incrementa cada vez que el SysTick interrumpe y entra en esta zona del código. Después de esto, se realiza una comparación entre el valor del contador y el valor con el que se quiere realizar la toma de datos. Este valor debe incluirse en el "if" de la comparación en milisegundos.

Cuando el contador alcanza el valor deseado entra en el "if" y tras resetear el contador comienza a ejecutar las funciones asociadas a la medición de los sensores una tras otra. Cuando termina de ejecutar las funciones de medición activa el flag "mostrar" que nos

indica que los datos están preparados para ser transmitidos por la UART. Esta parte del código se puede ver la ilustración 39.

```
451 void SysTick_Handler(void) {        //Se ejecuta periódicamente a Ftick (Hz)
452
453   cont++;
454
455   if(cont==1000){                  //Tiempo de muestreo en ms
456     cont=0;                        //Reset del contador
457     leer_temp_ambiente();
458     leer_luz_ambiente();
459     leer_temp_corporal();
460     leer_acelerometro_giroscopio();
461     mostrar=1;                     //Flag de mostrar datos
462   }
463 }
```

Ilustración 39 Manejador del SysTick

Cuando los datos están preparados para el envío y el flag “mostrar” está activo, sólo falta llamar a la función uart_mostrar_datos(). Esto se realiza en el programa principal, dentro de un bucle “while” infinito hasta que el envío de datos sea ininterrumpido. El bloque de programación asociado al programa principal se adjunta en la ilustración 40.

```
465 int main(void){
466
467   LPC_GPIO0->FIOPIN&=(~0xF);          //Limpiar bits SDA y SCL
468   LPC_PINCON->PINSEL1|= (1<<14);      // (AD0.0)
469
470   config_temp_ambiente();
471   config_acelerometro();
472   config_ADC();
473   SysTick_Config(100000);             //Configuración de interrupción 1ms
474
475
476   while(1){
477     if(mostrar){
478       mostrar=0;                        //Limpiar flag
479     uart_mostrar_datos();               //Muestra los datos
480     }
481   }
482 }
```

Ilustración 40 Programa principal

6.6. VISUALIZACIÓN DE DATOS EN TIEMPO REAL

6.6.1. RECEPCIÓN Y PROCESAMIENTO DE DATOS EN MATLAB®

Como se ha mencionado en el apartado 7.3.1. toda la secuencia de datos enviados desde el microcontrolador al PC por la UART debe ser recibida por algún programa capaz procesarlos para ser interpretados.

En este caso se usa el programa MATLAB® para que los datos se dibujen en tiempo real en gráficas independientes dependiendo de qué parámetro sea.

Para ello se ha generado un script en MATLAB®, llamado “ciclo_circadiano”, al que le tenemos que decir cuántas muestras queremos tomar para dibujar la gráfica. Este número de muestras será posteriormente la longitud del eje X de las gráficas.

Se tiene que tener en cuenta a la hora de introducir el número de muestras que este dato está directamente relacionado con el tiempo de funcionamiento del sistema durante el que va a estar pintando las gráficas. El tiempo total depende de dos parámetros:

- Número de muestras introducidas en MATLAB®.
- Tiempo de adquisición y envío de datos nuevos en SysTickHandler.

La ecuación que relaciona estos tres parámetros es la siguiente:

$$Tiempo\_Total = \frac{Número\_Muestras * Tiempo\_Adquisición}{1000} \ [s]$$

6.6.1.1. CÓDIGO DE PROGRAMACIÓN

En primer lugar se debe configurar el puerto de conexión COM3 para que se realice el enlazado correctamente y la transferencia de datos del microcontrolador al programa sea correcto. Después se debe abrir el puerto con la función fopen() para acceder a los datos. Estas primeras líneas del código se pueden ver en la ilustración 41.

```
 6    % Configuración puerto COM3
 7 -  delete (instrfind({'port'}, {'COM3'}));
 8 -  Puerto=serial('COM3');
 9 -  Puerto.BaudRate=9600;
10
11 -  fopen(Puerto);
```

Ilustración 41 Configuración y acceso al COM3

Cuando se tiene acceso a los datos que llegan del COM3 se deben recibir y meterlos en una variable para poder dividirlos. Recordemos que los datos vienen en una línea separados por tabulaciones, pero en esta primera variable se van a almacenar todos juntos. Para ello se utiliza la función fscanf() y se introduce este array de datos en la variable “valor_conversion”.

Posteriormente se debe dividir el array “valor_conversion” en datos independientes para posteriormete tratarlos de manera individualizada. Para ello se accede posición a posición del array y se guardan los datos en variables independientes.

El código asociado a esta recepción y posterior división de los datos se muestra en la Ilustración 42.

```
13      % Recepción muestras cronolojicamente
14 -    contador=1;
15 -    while contador<=muestras;
16 -        x=1:muestras;
17 -        valor_conversion=fscanf(Puerto,'%f')
18
19          %Guardar datos en variables independientes
20 -        temperatura_ambiente(contador)=valor_conversion(1);
21 -        temperatura_corporal(contador)=valor_conversion(2);
22 -        iluminacion(contador)=valor_conversion(3);
23 -        acel_x(contador)=valor_conversion(4);
24 -        acel_y(contador)=valor_conversion(5);
25 -        acel_z(contador)=valor_conversion(6);
26
```

Ilustración 42 Recepción y división de los datos en variables independientes

Una vez se tienen los datos separados en variables, se pasa al proceso de graficado de cada uno de los parámetros. Para ello se divide la pantalla principal en 4 espacios con la función subplot() para obtener las 4 gráficas de los datos que mide nuestro sistema.

Todo este proceso está incluido en un bucle que se ejecuta según el número de muestras, ya que todas estas instrucciones se deben ejecutar para cada una de las muestras que reciba el COM3 del microcontrolador.

El código asociado a la parte encargada de la creación y representación de las gráficas se muestra en la lustración 43.

```
% Obtención de las gráficas en tiempo real
    %Temperatura ambiente
    subplot(2,2,1),plot(temperatura_ambiente,'r'),title('Temperatura Ambiente')
    grid on
    xlabel('Muestras')
    ylabel('Grados celsius (°C)')
    ylim([0 50])
    xlim([0 muestras])

    %Temperatua corporal
    subplot(2,2,2),plot(temperatura_corporal,'r'),title('Temperatura Corporal')
    ylim([0 50])
    grid on
    xlabel('Muestras')
    ylabel('Grados celsius (°C)')
    xlim([0 muestras])

    %Luz ambiente
    subplot(2,2,3),plot(iluminacion,'r'),title('Iluminacion')
    grid on
    xlabel('Muestras')
    ylabel('Iluminación (Luxes)')
    ylim([0 600])
    xlim([0 muestras])

    %Posición
    subplot(2,2,4),plot(acel_x,'b'),title('Actividad')
    hold on
    plot(acel_y,'g')
    hold on
    plot(acel_z,'r')
    grid on
    xlabel('Muestras')
    ylabel('Aceleración (m/s2)')
    ylim([0 50])
    xlim([0 muestras])

    drawnow
    contador=contador+1

end
```

Ilustración 43 Obtención y dibujado de las gráficas

7. PRUEBAS Y RESULTADOS

Una vez configurado, desarrollado e implementado el sistema se pasa a realizar ensayos de situaciones reales a los que se puede enfrentar. Para ello se prueba la respuesta de cada uno de los sensores y se analiza que la respuesta sea adecuada de acorde a la situación que se desea medir.

7.1. TEMPERATURA CORPORAL

Para ver el funcionamiento del sensor de temperatura corporal se ha realizado una comparación con la medición simultánea del sensor de temperatura ambiente. Con ambos sensores midiendo a la vez durante 300 muestras con un periodo de muestreo de 800 ms, se establecen 3 ciclos de funcionamiento en el sensor de temperatura corporal. El resultado de la medición se puede ver en el fichero de MATLAB ®"Ensayo_temperatura_corporal.fig".

En el primer ciclo, ambos sensores están al aire midiendo la temperatura ambiente. En la gráfica se ve que están marcando temperaturas similares, el sensor de temperatura ambiente 27.5 ^{o}C y el de temperatura corporal oscila en torno a 27 ^{o}C. Se debe tener en cuenta que el sensor de temperatura corporal tiene muchas oscilaciones en su medición causadas por la toma de datos de la tensión un tanto inestable a la entrada del ADC.

En el segundo ciclo, desde la muestra 100 a la 200, se instala el sensor de temperatura corporal en el cuerpo de la persona ayudante para el ensayo. Se puede apreciar que la temperatura experimenta un ascenso bastante rápido hasta valores oscilantes alrededor de 36 ^{o}C.

Por último, a partir de la muestra 200 donde se retira el sensor del cuerpo del ayudante para dejarlo al aire. Se aprecia que tras unas muestras en las que el sensor tarda en responder, la temperatura desciende y se estabiliza a partir de la muestra 250 en valores similares a los del primer ciclo.

Las gráficas del ensayo referentes a la temperatura ambiente y corporal se muestran en las ilustraciones 44 y 45.

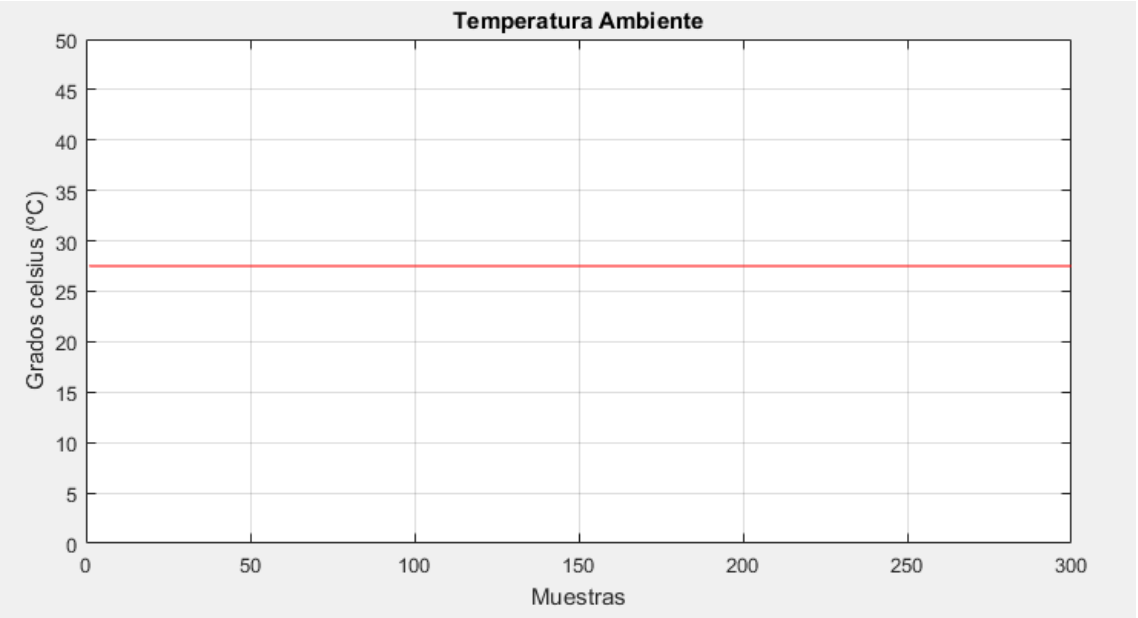

Ilustración 44 Evolución de la temperatura ambiente

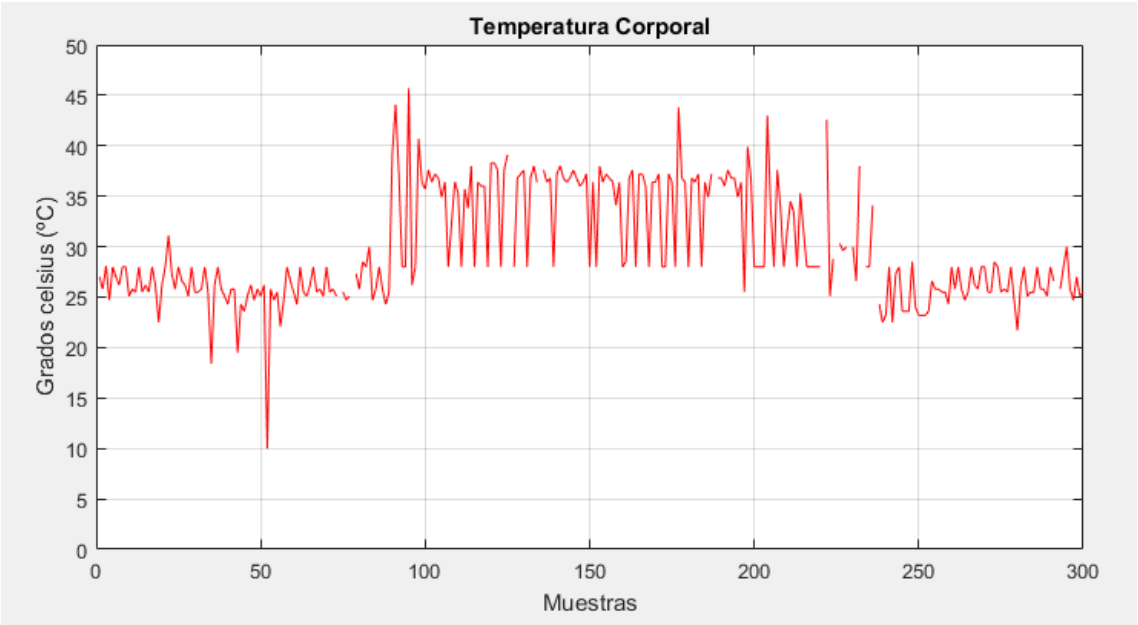

Ilustración 45 Evolución de la temperatura corporal

7.2. LUZ AMBIENTE

7.2.1. ENSAYO 1: PRUEBA CON VARIACIONES DE ALTURA EN PERSIANA

Para el ensayo de iluminación se han realizado unas variaciones en la elevación de una persiana y la apertura y cierre de una cortina. El periodo de muestreo del ensayo se establece en 800 ms.

En este ensayo se parte de la oscuridad, donde el sensor marca 0 luxes. Transcurridas unas 20 muestras, se eleva la persiana hasta la mitad de la ventana donde la gráfica alcanza el valor de 36 en el eje Y.

Una vez se estabiliza, se sube la persiana hasta lo máximo que nos permite la ventana y se abre la puerta de la ventana. Esto provoca que haya una subida hasta los 116 luxes en torno a la muestra 50. En este periodo, se ven oscilaciones en la señal provocadas por el movimiento de la cortina, ya que, al abrir la ventana la corriente de aire provoca movimientos en la cortina que causan oscilaciones en el flujo luminoso.

Cuando transcurren unas 70 muestras, se abre la cortina dejando libre el espacio entre el sensor y la calle. En este periodo se alcanza el valor de 266 luxes.

Una vez se estabiliza la medición, se comienza a bajar la persiana progresivamente hasta frenarla en el tercio inferior de la ventana. Esto produce un descenso en la luminosidad de la sala que se puede ver en el periodo comprendido entre las muestras 80 y 110.

Para finalizar el ensayo, sobre la muestra 140 se baja la persiana hasta el final y la habitación vuelve a la situación de oscuridad.

El archivo de MATLAB® referente a este ensayo se ha nombrado “Ensayo_iluminación.fig” y la gráfica del proceso se puede ver en la ilustración 46.

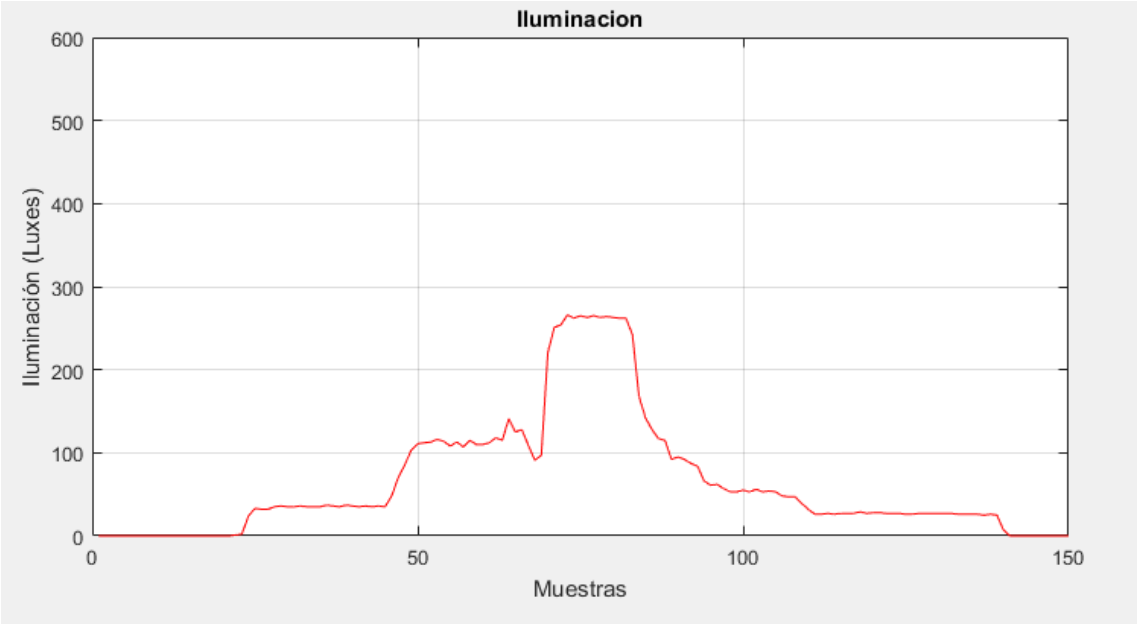

Ilustración 46 Evolución de la iluminación

7.2.2. ENSAYO 2: EVOLUCIÓN ILUMINACIÓN DURANTE 1 DÍA COMPLETO

En este ensayo se ha observado la evolución de la respuesta del sensor de luz ambiente durante un día completo. Durante el ensayo, el sistema ha estado enviando los datos aportados por los sensores con un tiempo de adquisición de 5 segundos, tiempo suficiente para captar las variaciones de luz en el transcurso del día.

El sensor se ha colocado en el cerco de una ventana sin ningún tipo de objeto que se interpusiera en la incidencia de la iluminación sobre el mismo.

El comienzo del ensayo se produjo el lunes 10 de septiembre de 2018 a las 16:30 horas, finalizando por tanto el 11 de septiembre de 2018 a las 16:30 horas.

La evolución del parámetro se puede ver en la ilustración 47.

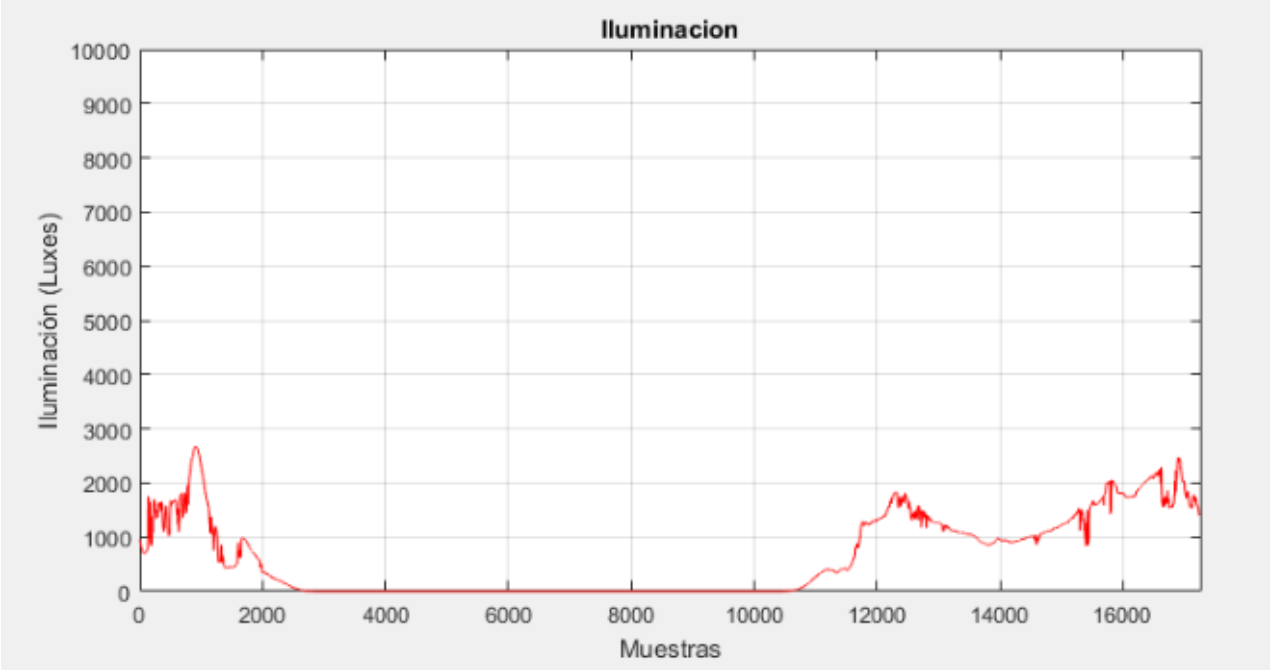

Ilustración 47 Evolución iluminación durante un día completo

7.3. TEMPERATURA AMBIENTE

Para ver el funcionamiento de este sensor y la evolución de este parámetro, se ha aprovechado el ensayo 2 de iluminación y se ha observado la evolución del dato de temperatura ambiente durante un día completo.

La gráfica de la respuesta del sensor se muestra en la ilustración 48.

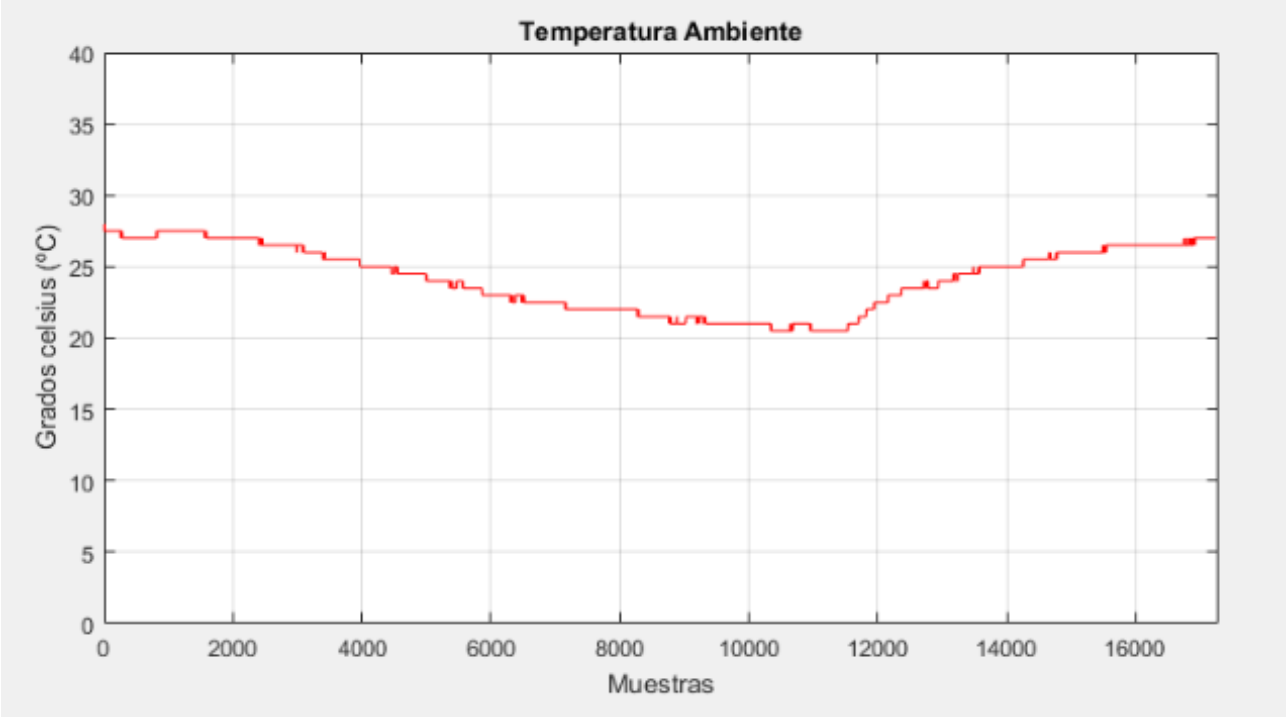

Ilustración 48 Evolución temperatura ambiente durante un día completo

7.4. ACTIVIDAD

Para ver el comportamiento del acelerómetro en un entorno de actividad física se ha sometido a un ensayo en el que, durante 300 muestras con un periodo de muestreo de 800 ms, el portador del dispositivo ha realizado dos ciclos simulando el estado de caminar, permanecer sentado y correr en estático.

Los periodos en el que se simula el estado de caminar son los comprendidos en los intervalos [0, 45] y [160, 210] muestras.

Los periodos en el que el paciente está sentado son los comprendidos en los intervalos [45, 90] y [210, 255] muestras.

Los periodos en el que el paciente está corriendo en estático son los comprendidos en los intervalos [90, 160] y [255, 300] muestras.

Recordar que el código de colores utilizado para la creación de la gráfica es la siguiente:

- Azul: Aceleración en el eje X.
- Verde: Aceleración en el eje Y.
- Rojo: Aceleración en el eje Z.

El archivo de MATLAB® referente a este ensayo se ha nombrado "Ensayo_actividad_fisica.fig" y la gráfica del proceso se puede ver en la ilustración 49.

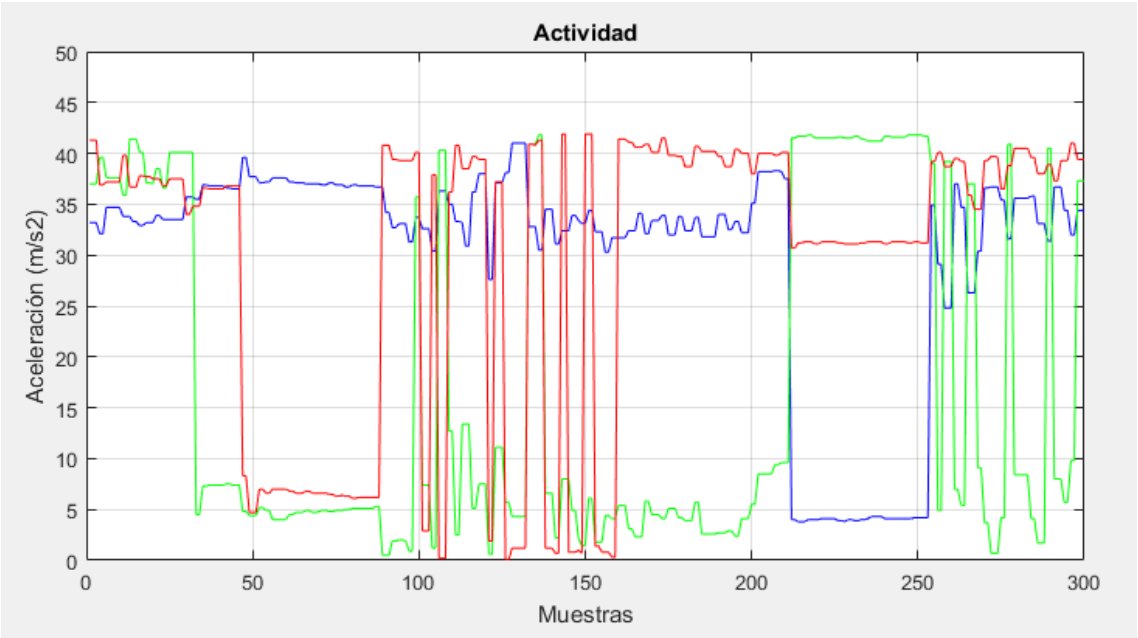

Ilustración 49 Ensayo de actividad física

8. CONCLUSIONES Y TRABAJOS FUTUROS

El sistema multisensorial diseñado junto a la aplicación creada para la visualización de los datos en tiempo real es un dispositivo de rápida respuesta a cambios producidos en su entorno, una virtud a recalcar si se desean tomar datos con unas frecuencias relativamente altas para los cambios producidos en la vida real humana.

Cierto es que el sensor de temperatura corporal genera mucho ruido que puede dificultar la visión de los valores reales del paciente, pero puesto que el sistema está pensado para estudios extendidos en el tiempo, la evolución de este parámetro se puede seguir y detectar cualquier comportamiento anómalo que se salga de los estándares de la evolución de la temperatura corporal periférica de los humanos.

En general, los datos aportados por los sensores son altamente fiables, ya que en los múltiples ensayos que se han hecho han respondido de una manera adecuada a los cambios generados en su entorno. Por este motivo creemos que el sistema puede ser de utilidad para la toma de datos y posteriores estudios del ciclo circadiano de personas con problemas de descanso, teniendo en cuenta que la interpretación de los datos obtenidos la deben realizar expertos en este campo de la medicina.

Como posibles trabajos futuros se propone la unificación de todos los sensores en un dispositivo portable. Para ello habría que dotar el sistema de una comunicación inalámbrica. Además se podrían crear nuevas aplicaciones para facilitar el estudio de los datos obtenidos.

9. PLIEGO DE CONDICIONES

Para la ejecución del proyecto se necesitan los siguientes materiales. Se desglosa el listado en las dos etapas generales en las que se divide dicho proyecto:

9.1. OBTENCIÓN DE DATOS

Para la realización de esta etapa se han utilizado los siguientes materiales:

- Ordenador PC con sistema operativo Windows 10.
- Programa Keil uVision versión 4.
- Microcontrolador Mini-DK2 LPC1768 Cortex-M3.
- Cables USB y JTAG de la marca HAOYU.
- Placa protoboard.
- Sensores.
    - Temperatura corporal. PTFC101T1A0
    - Luz ambiente. BH1750FVI
    - Temperatura ambiente. DS1621
    - Actividad. MPU-6050
- Cables jumper y resistencias.

9.2. MONITORIZACIÓN DE DATOS

Para la realización de esta etapa se han utilizado los siguientes materiales:

- Ordenador PC con sistema operativo Windows 10.
- Programa MATLAB® versión R2016b.

10. PRESUPUESTO

Los costes generados por la compra de los materiales necesarios para el desarrollo del proyecto se muestran en la tabla de la ilustración 50:

Descripción		Cantidad	Precio unida	Precio total
Hardware	PC	1	900,00 €	300,00 €
	LPC 1768 Mini-DK2	1	26,00 €	26,00 €
	Sensor de temperatura corporal PTFC101T1A0	1	4,46 €	4,46 €
	Sensor de luz ambiente BH1750FVI	1	5,99 €	5,99 €
	Sensor de temperatura ambiente DS1621	1	5,82 €	5,82 €
	Sensor de actividad MPU-6050	1	7,99 €	7,99 €
	Cable de conexión JTAG	1	13,00 €	13,00 €
Software	Windows 10	1	- €	- €
	Keil uVision	1	- €	- €
	MATLAB®	1	- €	- €
	Microsoft Office	1	- €	- €
TOTAL				**363,26 €**

Ilustración 50 Presupuesto de material

Tasas profesionales:

Los costes de acuerdo al tiempo profesional empleado en el diseño, desarrollo e implementación del proyecto se calculan tabla de la ilustración 51:

Descripción	Tiempo (horas)	Precio hora (€)	Precio total
Ingeniería	250	50	12.500,00 €
Escritura	80	25	2.000,00 €
TOTAL			**14.500,00 €**

Ilustración 51 Tasas profesionales

11.MANUAL DE USUARIO

A continuación se pasa a detallar la forma de poner en funcionamiento el sistema, así como la inicialización de la aplicación creada en MATLAB® para la visualización de las gráficas creadas con los datos recibidos.

11.1. EJECUCIÓN DE LA APLICACIÓN

Para la ejecución de la aplicación encargada de la representación de los datos se tienen que realizar una serie de pasos y tener en cuenta una serie de parámetros que influyen en el periodo de muestreo y el tiempo de operación de la representación de los datos.

En primer lugar se debe cargar el programa que maneja el sistema en el microcontrolador. En este código tenemos que tener especial atención en el valor con el que se leen los sensores y se envían los datos al PC. Este dato, que indica el tiempo de muestreo de datos en milisegundos, es el que aparece recalcado con un cuadrado rojo en la figura 52.

```
334 void SysTick_Handler(void) {          //Se ejecuta periódicamente a Ftick (Hz)
335
336   cont++;
337
338   if(cont==1000){                    //Tiempo de muestreo en ms
339     cont=0;                          //Reset del contador
340     leer_temp_ambiente();
341     leer_luz_ambiente();
342     leer_temp_corporal();
343     leer_acelerometro();
344     mostrar=1;                       //Flag de mostrar datos
345   }
346 }
```

Ilustración 52 Parámetro de configuración del tiempo de adquisición de datos

Una vez este cargado el programa en la Mini-DK2 se debe pulsar el botón de “Reset” de la placa, instante en el que la toma de datos se realiza de manera ininterrumpida con un tiempo de adquisición de datos que anteriormente establecido.

Cuando la toma de datos está en ejecución, se debe ejecutar el script de MATLAB® llamado “ciclo_circadiano.m”. Para su ejecución, se deberá escribir en la ventana de comandos la instrucción “ciclo_circadiano(60)”, siendo el valor numérico ubicado en el paréntesis el número de muestras que la aplicación va a representar en las gráficas. Este valor es introducido por el usuario y puede ser cualquiera.

En este proceso, es interesante tener en cuenta la ecuación mencionada en el apartado 6.6. para poder calcular el tiempo de ejecución de la aplicación de representación.

12. BIBLIOGRAFÍA

[1] Madrid, J. A. y Rol, M. A. (2015). Ritmos, Relojes y Relojeros. Una introducción a la cronobiología. *Revista Eubacteria, 33.* 1-7.

[2] Saavedra, J.S.; Zuñiga, L.; Navia, C.A. y Vásquez, J.A. (2013). Ritmo circadiano: el reloj maestro. Alteraciones que comprometen el estado de sueño y vigilia en el área de la salud. *Morfolia, 3*, (5). 16-30.

[3] Martínez-Nicolás, A. y Blázquez-Manzanera, A. (2015). La hora de nuestro cuerpo. Monitorización Ambulatoria Circadiana. *Revista Eubacteria, 33.* 22-28.

[4] Argüelles, R. y Bonmatí, M.A. (2015). Melatonina, la hormona de la noche. *Revista Eubacteria, 33.* 16-21.

[5] Blazquez, A.; Martinez-Nicolas, A.; Salazar, F.J.; Rol, M.A. y Madrid, J.A. (2012) Wrist Skin Temperature, Motor Activity, and Body Position as Determinants of the Circadian Pattern of Blood Pressure. *Chronobiology International, 29.* 747-756.

[6] Sensor Solutions (2015). PT TEMPERATURE SENSOR – PTF FAMILY. Recuperado de https://docs-emea.rs-online.com/webdocs/1554/0900766b8155405c.pdf en fecha 10/09/2018.

[7] ROHM Semicondutor (2014). Digital 16bits Serial Output Type Ambient Light Sensor IC. Recuperado de https://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf en fecha 10/09/2018.

[8] Dallas Semiconductor (2015). DS1621 Digital Thermometer and Thermostat. Recuperado de https://docs.isy.liu.se/pub/VanHeden/DataSheets/ds1621.pdf en fecha 10/09/2018.

[9] IvenSense Inc. (2013). MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2. Recuperado de https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf en fecha 10/09/2018.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá