Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Diseño de un instrumento basado en Arduino para la medida de irradiancia solar

Autor:
Bayón Alonso, Eduardo

Tutor:
Moríñigo Sotelo, Daniel
Ingeniería Eléctrica

Valladolid, Febrero de 2018.

Agradecimientos

En primer lugar, me gustaría indicar que este trabajo y todo lo que conlleva detrás, ha sido posible gracias al apoyo continuo e incansable de mi familia, en especial, mis padres que creyeron en mí, siempre. Me han hecho darme cuenta que, todo, con trabajo y constancia, se consigue.

También agradecer a mis amigos, tanto los encontrados en la carrera como los de fuera de la universidad, los cuales me han ayudado y apoyado mostrándome su amistad y confianza.

Por último, agradecer a mi tutor Daniel y al técnico de laboratorio David Moro su ayuda en la realización de este trabajo, solucionando cualquier duda o problema que pudiera tener.

Con vuestra ayuda todo ha sido más fácil, por todo ello, gracias.

Resumen

El objetivo de este trabajo es el diseño y construcción de un instrumento autónomo y de bajo coste que permita capturar y guardar la información que proporciona una célula fotovoltaica calibrada y un sensor de temperatura. Esos datos, son guardados en una tarjeta MicroSD, y mostrados por una pantalla LCD. El control y manejo de la información se realiza por medio de un microcontrolador Intel Edison, y el control físico del dispositivo se realiza mediante unos botones y un interruptor. Todo ello, se trata de un proyecto destinado al grupo ADIRE de la Universidad de Valladolid y permite obtener esta información, que en instalaciones domésticas o de poca potencia, no suele ser posible al no estar dotadas de una estación meteorológica como pueda ser en las grandes instalaciones fotovoltaicas.

Palabras clave

Fotovoltaica, Irradiancia, Temperatura, Microcontrolador, Electrónica.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1: Introducción y objetivos.

1.1 Antecedentes

A finales del siglo XIX el inventor Charles Fritts desarrolló la primera célula solar con una eficiencia mínima entorno a un 1 por ciento, aunque el efecto fotovoltaico fuera descubierto o nombrado por primera vez sobre 1840 por el físico Becquerel [1]. Diferentes estudios llevados a cabo por los físicos e ingenieros más importantes como H. Hertz, M. Planck, A. Einstein o R. A. Millikan, proporcionaron la base al efecto fotoeléctrico para fundamentar la conversión de energía solar a energía eléctrica [2].

Una vez ya más desarrollados y avanzados los estudios, se patentó la primera célula solar moderna a mediados del siglo XX. Su primer «boom» llegó de la mano de los Laboratorios Bell mediante el dopaje de silicio, lo que hizo que la eficiencia de la conversión llegara aproximadamente al 6 por ciento [3]. Poco más tarde, la compañía Hoffman Electronics llevó la producción de las células solares a gran escala, con una eficiencia del 14 por ciento y con una gran reducción de los costes de fabricación debido a ese tipo de producción [3][4].

Como primeras aplicaciones de las células fotovoltaicas destacan las destinadas a la carrera espacial sobre 1960 y los años posteriores. Las células solares se encargaban y se encargan de alimentar eléctricamente los diferentes vehículos y satélites que orbitan alrededor de la tierra [3]. Cabe a destacar la Estación Espacial Internacional que se puede observar en la figura 1.

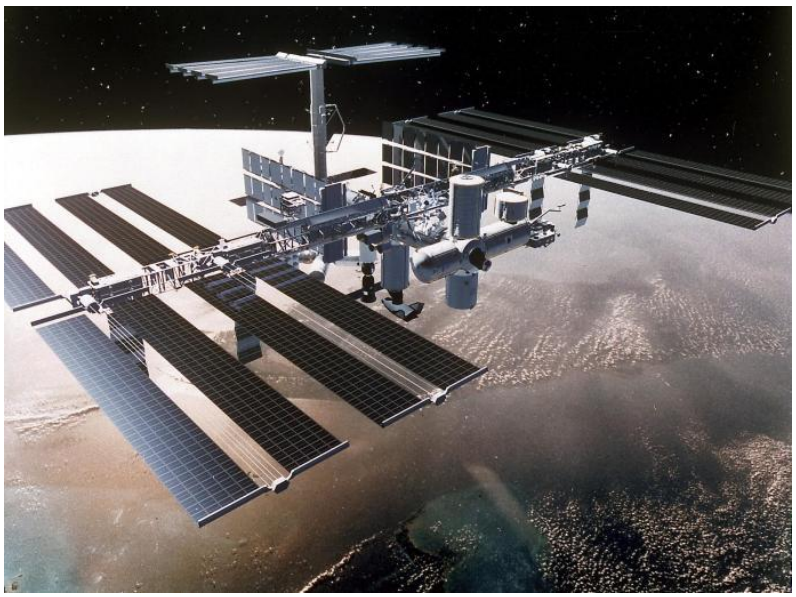

Figura 1. Estación Espacial Internacional. NASA.

En las siguientes décadas esta tecnología fotovoltaica se ha desarrollado de forma lenta y gradual hasta principios del siglo XXI debido al predominio del carbón y petróleo como fuentes de energía mundial.

El segundo gran «boom» que comenzó a principios del presente siglo, se ha visto propiciado por la necesidad de obtener energía de una forma más limpia, es decir, menos contaminante a la manera en que se llevaba haciendo durante todo el siglo anterior con la quema de combustibles fósiles. Esta quema de combustibles hace que el medio ambiente y la población salgan realmente perjudicados ya que se produce un importante aporte de gases de efecto invernadero que aparte de ser tóxico para los seres vivos, aumenta la temperatura del planeta con las consecuencias que ello provoca como, por ejemplo, el deshielo de los casquetes polares, un aumento de la desertificación, así como de la magnitud de fenómenos naturales como pueden ser los huracanes.

Por ello, en la actualidad, cada día son más las instalaciones de grandes campos fotovoltaicos productores de energía eléctrica (figura 2), los cuales se sitúan fuera de las ciudades (debido a las extensiones que ocupan) con el fin de suministrar esa energía obtenida a las redes de distribución o transporte. Es una gran forma de obtener energía limpia contribuyendo al medio ambiente.

Figura 2. Campo solar.

No obstante, no solamente se realizan instalaciones solares fotovoltaicas fuera de las ciudades y de gran envergadura. Cada vez son más los hogares que al construirse, se dotan de esta tecnología. Su uso está en constante y creciente expansión debido a la necesidad de ahorrar costes en la factura de la luz, mejorar la calidad y habitabilidad de los edificios a la vez que

se intenta reducir el uso de combustibles fósiles (que utilizan las calderas contaminantes) y así, proteger el medio ambiente.

También existen multitud de aplicaciones aisladas para la energía solar como pueden ser el bombeo de agua, la electrificación de núcleos de población aislados, el abastecimiento de la energía necesaria para la señalización (balizas, boyas…), iluminación (faros, farolas…) y sistemas de comunicación (estaciones repetidoras), la recarga de aparatos electrónicos aislados, etc. [5] Se pueden observar algunas de ellas en la figura 3.

Figura 3. Diversas aplicaciones fotovoltaicas.

Todos estos aspectos, ya sean de mayor o menor envergadura, son muy importantes ya que, al fin y al cabo, contribuyen al bienestar de todos mediante unas simples medidas.

1.2 Justificación del TFG

Este Trabajo Fin de Grado (TFG) se va a realizar en el Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales, y más concretamente, en colaboración con el Grupo de Investigación Reconocido (GIR) Análisis y Diagnóstico de Instalaciones y Redes Eléctricas (ADIRE).

Este grupo tiene una línea de investigación sobre el análisis de la calidad de la energía eléctrica generada en sistemas fotovoltaicos. Actualmente está realizando trabajos en una planta solar de 50 MW en la provincia de Cuenca y en las instalaciones solares del Edificio Lucía de la Universidad de Valladolid. Los estudios demuestran que la calidad de la energía eléctrica generada se ve afectada por las condiciones meteorológicas (irradiancia y temperatura) en las

que trabajan las placas fotovoltaicas. En grandes instalaciones se suele disponer de estaciones meteorológicas que miden y capturan esta información. Sin embargo, en instalaciones domésticas o de poca potencia, esta información no suele estar disponible. Para continuar con los trabajos del grupo ADIRE en este tipo de instalaciones es necesario disponer de un equipo propio que permita capturarla. Esta es la razón inicial que justifica la propuesta y realización de este TFG, donde se quiere obtener un instrumento autónomo y de bajo coste que permita capturar y guardar la información que proporciona una célula calibrada y un sensor de temperatura. También se requiere que esos datos sean guardados en un elemento de memoria, como pueda ser una tarjeta MicroSD, y mostrados por un LCD o similar.

1.3 Objetivos y competencias

En este TFG, además de los objetivos técnicos, que se describen más adelante, también se pretende desarrollar unos objetivos formativos, que son los siguientes:

- Aprender a organizarse y estructurar el trabajo de una forma eficiente.
- Adquirir una base de conocimientos relacionados con las energías renovables.
- Adquirir conocimientos sobre instalaciones fotovoltaicas y los diferentes elementos que la componen.
- Estudiar sus diversas aplicaciones y entender su funcionamiento.
- Conocer las diferencias entre los distintos tipos de placas computadoras y microcontroladores como puedan ser Raspberry Pi, Arduino… y su entorno.

Los objetivos específicos técnicos del TFG se describen a continuación:

- Utilizar los conocimientos obtenidos con el fin de adoptar la mejor solución a la hora de ejecutar el proyecto.
- Mediante la programación en Arduino, crear un programa capaz de almacenar los datos recibidos de la placa solar calibrada y del sensor de temperatura en una tarjeta de memoria MicroSD.
- Indicar en un LCD o similar esos datos que va obteniendo el dispositivo.

Este TFG también pretende contribuir al desarrollo de las siguientes competencias del Grado en Electrónica Industrial y Automática [6]:

- Competencias generales:

  - **CG1**. Capacidad de análisis y síntesis.
  - **CG2**. Capacidad de organización y planificación del tiempo.
  - **CG3**. Capacidad de expresión oral.
  - **CG4**. Capacidad de expresión escrita.
  - **CG5**. Capacidad para aprender y trabajar de forma autónoma.
  - **CG6**. Capacidad de resolución de problemas.
  - **CG7**. Capacidad de razonamiento crítico/análisis lógico.
  - **CG8**. Capacidad para aplicar los conocimientos a la práctica.
  - **CG10**. Capacidad para diseñar y desarrollar proyectos.
  - **CG11**. Capacidad para la creatividad y la innovación.
  - **CG12**. Capacidad para la motivación por el logro y la mejora continua.
  - **CG13**. Capacidad para actuar éticamente y con compromiso social.
  - **CG14**. Capacidad de evaluar.
  - **CG15**. Capacidad para el manejo de especificaciones técnicas y para elaboración de informes técnicos.

- Competencias específicas:

  - **CE3.** Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
  - **CE10.** Conocimiento y utilización de los principios de teoría de circuitos y máquinas eléctricas.
  - **CE11.** Conocimientos de los fundamentos de la electrónica.
  - **CE12.** Conocimientos sobre los fundamentos de automatismos y métodos de control.
  - **CE16.** Conocimientos básicos y aplicación de tecnologías medioambientales y sostenibilidad
  - **CE18.** Conocimientos y capacidades para organizar y gestionar proyectos. Conocer la estructura organizativa y las funciones de una oficina de proyectos.
  - **CE21.** Conocimiento de los fundamentos y aplicaciones de la electrónica digital y microprocesadores.
  - **CE29.** Capacidad para diseñar sistemas de control y automatización industrial.

1.4 Planificación del proyecto.

Para el desarrollo del proyecto se han realizado las siguientes actividades planeadas en el horizonte temporal.

1. Primera reunión con el Tutor del TFG para fijar las bases del proyecto.
2. Reuniones de seguimiento. Con el fin de indicar al profesor el desarrollo del TFG y consultar las dudas sobre éste.
3. Estudio del problema y búsqueda de información. Se analiza el problema a resolver y se busca la información que permite llegar a una solución a ese problema.
4. Selección de los dispositivos. Con la información buscada, se analiza, y según los requerimientos y limitaciones que se tengan y surjan, se escogen los elementos que conforman el dispositivo.
5. Programación en Arduino. Se recuerdan y se aprenden conceptos en la programación de Arduino y se implementan en el desarrollo del TFG.
6. Construcción del dispositivo. Se construye el dispositivo físico una vez se han adquirido los materiales y se han hecho las pruebas suficientes anteriormente.
7. Prueba del dispositivo. Una vez construido el dispositivo, se comprueba que funciona correctamente.
8. Análisis de los resultados. Se analizan los resultados correspondientes a las pruebas comparándolos con la realidad.
9. Conclusiones. Se realiza un análisis de lo aprendido en el proyecto, incluyendo posibles mejoras.
10. Escritura de la memoria. Se redacta el informe donde se detallan los pasos seguidos en la realización del proyecto.

En la página siguiente, se puede observar el diagrama de Gantt, así como una estimación en horas (tabla 1) de la realización del proyecto.

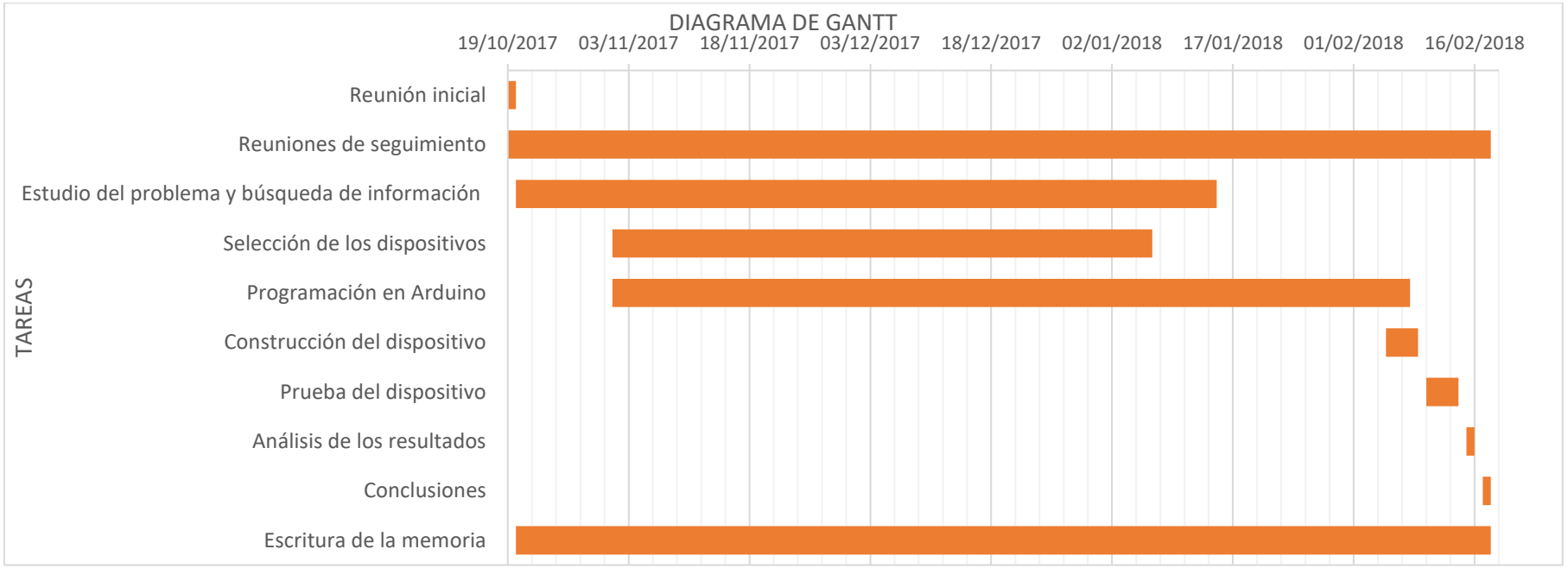

Tabla 1. Duración estimada del proyecto.

TAREA	FECHA DE INICIO	FECHA FINAL	HORAS ESTIMADAS
Reuniones	19/10/2017	18/02/2018	8
Estudio del problema y búsqueda de información	20/10/2017	15/01/2018	80
Selección de los dispositivos	01/11/2017	07/01/2018	20
Programación en Arduino	01/11/2017	08/02/2018	65
Construcción del dispositivo	05/02/2018	09/02/2018	15
Prueba del dispositivo	10/02/2018	14/02/2018	15
Análisis de los resultados	15/02/2018	16/02/2018	4
Conclusiones	17/02/2018	18/02/2018	3
Escritura de la memoria	20/10/2017	18/02/2018	110
		TOTAL	320

1.5 Estructura de la memoria

Este documento se estructura en forma de capítulos subdivididos en diferentes apartados que se expresan de la siguiente manera:

- Capítulo 1: Justificación y objetivos. En estos apartados se explica brevemente el fundamento del proyecto, así como los objetivos que se pretenden alcanzar.
- Capítulo 2: Estado del arte. En este capítulo se hace un resumen de las diferentes disciplinas de las que se va a necesitar para realizar el TFG, así como la información ya existente de lo que se va a tratar en este proyecto.
- Capítulo 3: Elección y descripción de los dispositivos utilizados. En este capítulo se explica el porqué de los dispositivos utilizados, ventajas e inconvenientes frente a otras posibilidades y una descripción detallada de estos.
- Capítulo 4: Programación Arduino. A lo largo de este capítulo se explica el funcionamiento del programa realizado mediante el IDE de Arduino que será implementado en el microcontrolador de la placa computadora. Cada apartado explicará las diferentes funciones.
- Capítulo 5: Construcción del dispositivo y pruebas. Aquí se relatará el procedimiento de construcción y montaje del aparato medidor y el prototipado del mismo. A su vez, se relatarán las pruebas realizadas y los resultados de estas.
- Capítulo 6: Costes del proyecto. Este capítulo expone el coste del proyecto dependiendo de los tipos de costes directos o indirectos.
- Capítulo 7: Problemas encontrados. Se explican todos los problemas encontrados a la hora tanto de buscar información como programar como de entender el manejo y utilización de los dispositivos.
- Capítulo 8: Conclusiones y posibles mejoras futuras. En este apartado se relatan las conclusiones y resultados obtenidos de la consecución del TFG, además de las posibles mejoras futuras que puedan realizarse.
- Bibliografía. Se detallan todas las fuentes de las que se ha obtenido la información utilizada para el correcto desarrollo trabajo.
- Anexos. Se incluyen en este capítulo todos los anexos relacionados con el programa de Arduino, el manual de usuario y las hojas técnicas de los elementos más esenciales del proyecto.

Capítulo 2: Marco de trabajo y estado del arte.

2.1 Marco de trabajo

Para contextualizar este TFG es necesario entrar a hablar sobre las principales disciplinas y campos de la ingeniería en los cuales se basa. Estos son la electrónica y la energía solar fotovoltaica.

2.1.1 Electrónica

Este campo se basa en la ingeniería eléctrica ya que utiliza numerosos principios matemáticos y físicos con base en el fenómeno eléctrico, aunque se especializa en circuitos de bajo voltaje principalmente.

La electrónica mediante su implementación por medio de circuitos electrónicos (serie de componentes como resistencias, condensadores, inductancias, diodos, reguladores… conectados eléctricamente entre sí) se encarga principalmente de la conversión y distribución de la energía eléctrica, y del control, procesado, y distribución de la información.

Se podría decir entonces, que los campos de acción de la electrónica están relacionados con la electrónica de potencia, por la facultad para adaptar y transformar la energía eléctrica para un uso posterior en dispositivos electrónicos y eléctricos, y con la computación y electrónica digital, ya que utiliza de microprocesadores y microcontroladores que tratan la información. Gracias a este tratamiento de la información también se permite el control de procesos industriales, pudiendo supervisarlos, automatizarlos… Tiene a su vez gran relación con las telecomunicaciones debido a todo el procesamiento y transmisión de datos.

El diagrama de la figura 4 resume la composición y la forma de actuar de un sistema electrónico.

Figura 4. Diagrama de un sistema electrónico.

2.1.2 Energía solar fotovoltaica

En la introducción se ha realizado un resumen de la historia de la energía fotovoltaica. Ahora se procede a explicar a grandes rasgos de qué trata ésta.

La energía solar fotovoltaica es una fuente de energía capaz de producir electricidad a partir de la radiación solar incidente en una célula fotovoltaica (figura 5), normalmente de silicio, mediante el efecto fotoeléctrico.

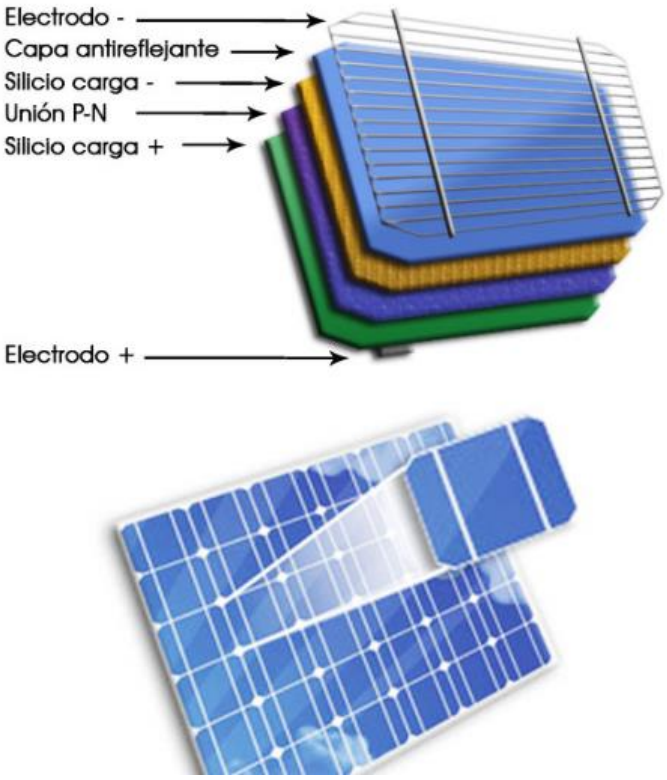

Figura 5. Célula fotovoltaica.

Este efecto se produce cuando las partículas de la luz (fotones) impactan sobre los semiconductores de la célula transmitiendo su energía a los electrones de valencia del semiconductor y haciendo que rompan el enlace de sus átomos. De esta forma, por cada enlace que se rompe se genera un electrón libre. La falta de ese electrón que circula libremente por el sólido se denomina hueco, y también puede desplazarse por el sólido libremente comportándose como una partícula de carga positiva.

El movimiento de estos pares electrón-hueco en direcciones opuestas hacia los electrodos (negativo y positivo respectivamente) genera una corriente en el semiconductor que puede circular por un circuito externo convirtiendo así, la energía procedente de los fotones en energía eléctrica (figura 6) [7].

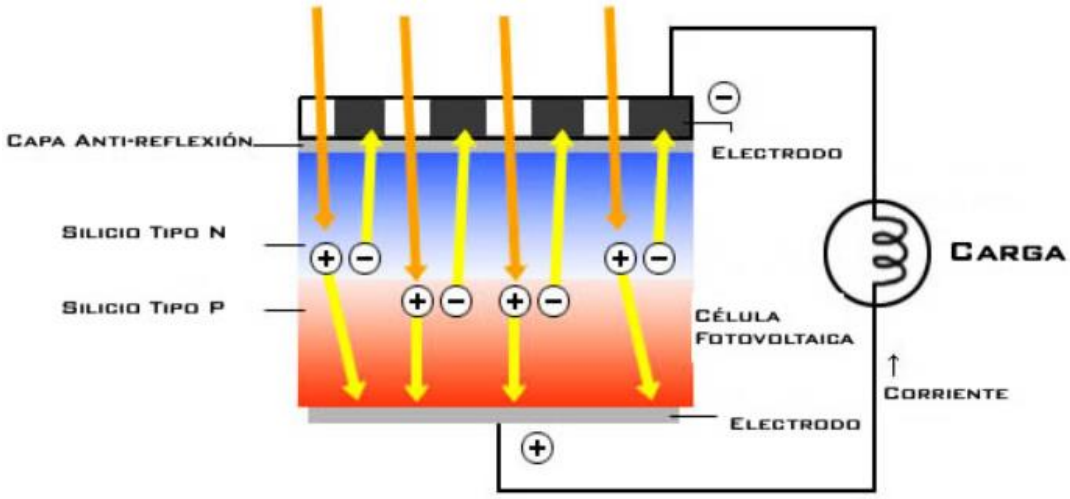

Figura 6. Explicación efecto fotoeléctrico.

Esta energía se considera renovable ya que se obtiene de fuentes naturales como es, en este caso, el sol. Además es una energía limpia ya que, únicamente contamina cuando se producen las placas solares y los demás elementos que la acompañan, y no cuando están generando electricidad.

2.2 Estado del arte

A la hora de hablar del estado del arte, en este TFG se debe de hacer hincapié en cuatro elementos que serán los fundamentales. Estos son, el microcontrolador, el sensor de temperatura, el sensor de irradiancia solar y la fuente de alimentación.

2.2.1 Microcontrolador y placas computadoras

En la actualidad, cada vez son más los dispositivos que utilizan microcontroladores (figura 7), su evolución ha sido exponencial. Desde su aparición, nuestra forma de vida ha cambiado sin apenas darnos cuenta, facilitándonos tareas que antes eran mucho más complejas y costosas.

Figura 7. Microcontrolador.

Un microcontrolador (μC) es una estructura de dimensiones reducidas formada por un material semiconductor, normalmente silicio, que contiene un circuito integrado programable, el cual, ejecuta las tareas que se le programan y que tiene grabadas.

Existen dos tipos de arquitectura de computadores: la Von Neumann y la Harvard. Los microcontroladores al ser pequeñas unidades de cómputo, poseen también estas arquitecturas, siendo la Von Neumann la utilizada inicialmente y dejando paso en los últimos años a la Harvard. Se detallan brevemente a continuación [8]:

- Arquitectura Von Neumann. En esta arquitectura la CPU está conectada a una sola memoria, la cual contiene todas las instrucciones de programa y los datos (figura 8).

Figura 8. Arquitectura Von Neumann.

Para acceder a esta memoria se cuenta con un único sistema de buses (de direcciones, datos e instrucciones y control). El ancho del bus de datos e instrucciones fija el tamaño de estos, y en el caso de tener que acceder a un dato o instrucción con un tamaño mayor al que proporciona el bus, tendrá que realizar más de un acceso a memoria, retardando el proceso. Esto sumado a que no se puede realizar la búsqueda de una nueva instrucción sin haber finalizado la transferencia de datos de la instrucción anterior, hace que se esté utilizando más la arquitectura Harvard.

- Arquitectura Harvard. A diferencia de la arquitectura Von Neumann, la Harvard posee dos memorias independientes, la de instrucciones del programa y la de datos, conectadas por buses diferentes (figura 9).

Figura 9. Arquitectura Harvard.

Al ser memorias independientes y buses independientes, la longitud de los datos o instrucciones no afecta, ya que el ancho de estos no tiene por qué ser el mismo, y a su vez, al tener sistemas de buses diferenciados, la CPU puede acceder simultáneamente a los datos para realizar la instrucción que se está ejecutando a la vez que lee la siguiente instrucción a ejecutar, lo que hace que se obtenga una mayor velocidad de operación.

Con respecto al tipo de instrucciones que manejan los controladores, se dividen en dos tipos (figura 10) [9]:

- RISC (Reduced Instruction Set Computer). Son los más utilizados actualmente en microcontroladores. Aquí el microcontrolador únicamente reconoce y ejecuta operaciones básicas, tiene un juego de instrucciones reducido y de tamaño fijo. Las operaciones complicadas resultan de combinar las básicas. Una de las ventajas es, por ejemplo, que facilitan el paralelismo en la ejecución y reducen los accesos a memoria, haciendo éste más rápido.

- CISC (Complex Instruction Set Computer). Los microcontroladores de instrucciones complejas están diseñados para reconocer un gran número de instrucciones, que resultan más potentes, aunque debido a ello también se dificulta el paralelismo entre instrucciones. En la actualidad muchos de los sistemas CISC utilizan un conversor de instrucciones complejas a simples tipo RISC (microinstrucciones) con el fin de facilitar ese paralelismo.

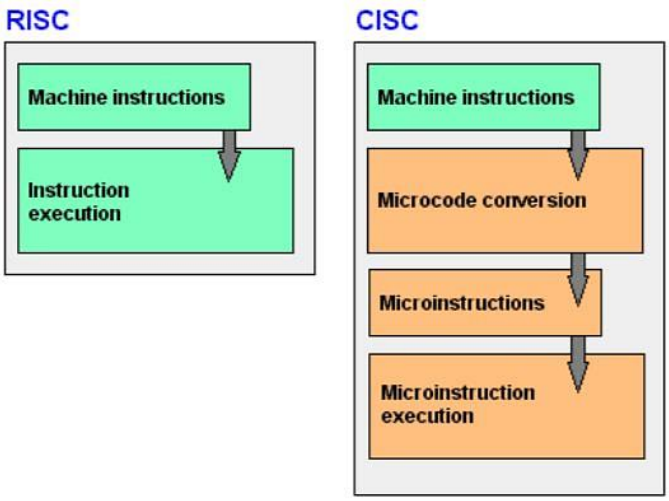

Figura 10. Diagrama RISC-CISC.

Físicamente un microcontrolador está formado principalmente por tres bloques o unidades funcionales (figura 11) que se detallan a continuación [10]:

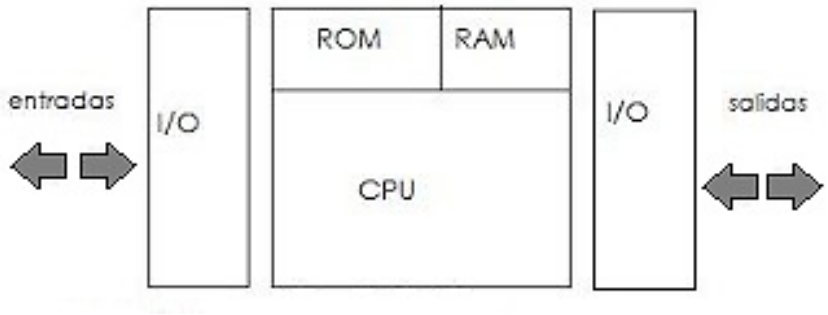

Figura 11. Diagrama interno del microcontrolador.

- <u>Unidad central de procesamiento (CPU) (figura 12).</u> Es lo que habitualmente llamamos microprocesador o procesador. Se considera el elemento principal del microcontrolador, ya que se encarga de controlar, procesar y ejecutar todas las instrucciones almacenadas en la memoria mediante la realización de operaciones aritméticas y lógicas.

Figura 12. CPU.

- <u>Memoria.</u> Son los componentes que sirven para almacenar la información que es utilizada durante la ejecución del programa. Esta información estará formada, tanto por el código del programa, como por los datos y variables que utilicemos en la ejecución del mismo. El tipo de memoria a emplear dependerá de la información que se vaya a guardar. Por ello, para el programa de instrucciones será necesaria una memoria no volátil (que no borre lo almacenado una vez se desconecte de alimentación) de tipo ROM (*Read Only Memory*) o de «sobretodo escritura».

  Ahora bien, dentro de las memorias ROM, las que se pueden encontrar en microcontroladores son [11]:

- Máscara ROM. Su contenido se graba durante la fabricación del chip. El diseño de la máscara tiene un precio muy alto, lo que hace que solo sea recomendable cuando se necesitan varios de miles de estos microcontroladores.
- Memoria PROM u OTP. En este caso la memoria solo es programable una vez por el usuario mediante algún tipo de programador especial. Recomendable para situaciones en las que no se necesitan nuevas actualizaciones y para tiradas pequeñas.
- Memoria EPROM. Es un tipo de memoria que pueden grabarse y borrarse muchas veces. Una vez programada, si se desea borrar, se ha de someter a rayos ultravioleta una ventana de cristal ubicada en el chip durante un determinado tiempo. Se encuentra en desuso debido a la aparición de las dos memorias siguientes.
- Memoria EEPROM. En esta memoria, se puede escribir y borrar muchas veces. La diferencia reside en que el borrado se realiza eléctricamente desde el propio grabador desde el que se programa. Su borrado es rápido, no como las EPROM.
- Memoria Flash. Es una memoria no volátil derivada de la EEPROM que utiliza pequeños impulsos eléctricos y que puede funcionar como ROM o RAM. De pequeño tamaño y menor consumo. En ella se puede escribir y borrar, tolera más ciclos, es más rápida y de menor consumo que la EEPROM.

Respecto a los datos y las variables, será utilizada una memoria volátil de almacenamiento temporal lectura/ escritura, tipo RAM (*Random Access Memory*), sin necesidad de tener una gran capacidad ya que solo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa.

Las más empleadas correspondientes a la memoria RAM son [12]:

- DRAM. Es un tipo de memoria dinámica, la cual necesita continuamente ser actualizada (teniendo que añadir circuitería). Tiene mayor capacidad de almacenaje de información que la SRAM en el mismo tamaño. Son más baratas y de menor tamaño al necesitar de menos transistores por bits de datos.
- SRAM. Es un tipo de memoria estática que no necesita ser "refrescada" constantemente, y no tienen una gran capacidad. Sus ventajas son el consumo energético y la velocidad.

Hay algunos microcontroladores que utilizan una EEPROM o Flash para que en caso de cortarse el suministro no se ocasione pérdida de información, pero esto, es menos común.

- **Líneas de entradas y salidas.** Estos elementos tienen diversos usos. Sirven para alimentar el microcontrolador, permiten comunicarse con elementos externos y escribir o leer en o de ellos desde el microcontrolador. Muchas de estas E/S, para tener más funcionalidades se encuentran multiplexados con funciones diferentes, que por ejemplo soportan comunicación serie, protocolos SPI, I2C, USB, modulación por PWM…

  Estos elementos dependiendo del tipo de microcontrolador, pueden ir acompañados de más bloques o componentes como pueden ser contadores, temporizadores, watchdogs, convertidores analógico/digitales, comparadores, etc.

  Los ámbitos en los que se pueden emplear estos dispositivos son muy variados, aunque caben a destacar los siguientes:

- **Automoción.** Utilizados tanto para controlar el motor, como el climatizador, los sistemas de seguridad (ABS, ESP entre otros)…
- **Electrodomésticos.** Se pueden encontrar en cualquier lavadora, televisor, frigorífico, batidora…
- **Informática.** Controlando periféricos como pueden ser un ratón, un teclado, una impresora, etc.
- **Telefonía y comunicación.** Donde el ejemplo más claro es el teléfono móvil.
- **Domótica.** En sistemas de vigilancia y antirrobo, climatizadores…
- **Instrumentación.** Encontrándolos en todo tipo de equipos de medida.

Respecto a la industria, se ha de destacar la labor fundamental de estos, consistente en la regulación, ya sea de velocidad de las máquinas, de regulación de niveles de aceite, alturas, temperaturas, etc; en automatismos con el control de máquinas, y herramientas, cierre y apertura de pinzas, puertas; en robótica con el control de los motores y sensores varios… [13] [14]

Entre las empresas más importantes fabricantes de microcontroladores (figura 13) se encuentran Intel, Atmel, Freescale (subdivisión de Motorola), National Semiconductor, Microchip Technology, NXP Semiconductors (subdivisión de Philips), Renesas Technology (una joint venture entre Hitachi y Mitsubishi Electric), STMicroelectronics, Texas Instruments y Zilog [15].

Figura 13. Principales fabricantes de microcontroladores.

A partir de estos microcontroladores, han aparecido en los últimos años unos dispositivos denominados placas de desarrollo o placas computadoras. Estas placas de circuito impreso son gobernadas por un microcontrolador o un microprocesador, y dependiendo de lo especializada que sea ésta, pueden poseer diversos elementos como, puertos analógicos y digitales de entrada y salida que permiten conectar componentes y otras placas (*shields*) que amplían su funcionamiento, entradas y salidas de vídeo y audio, conexiones Ethernet, USB y microUSB, slots para SD o microSD, convertidores analógico-digitales, etc.

Existen algunas placas gobernadas por microcontroladores más avanzados, complejos y potentes que pueden incluir un Sistema Operativo (SO). Este tipo de microcontrolador más desarrollado posee mayor memoria RAM, y es denominado *System on Chip* (SoC). Algunos de estos incluyen módulos de comunicación inalámbrica (WiFi, Bluetooth u otros), Unidades de Procesamiento Gráfico (GPU), chips de gestión de interfaces USB, SD, circuitos de gestión de energía..., todos ellos dentro del mismo integrado [16].

A pesar de que existen gran variedad de fabricantes de placas computadoras, principalmente destacan dos que dominan la mayor cuota del mercado. Estos son:

- Arduino (figura 14) [17]. Esta compañía comenzó en Italia hace 13 años como un proyecto para estudiantes que abaratara los costes de los microcontroladores que ellos usaban. El proyecto basado en hardware y software libre fue tomando forma, mejorando e incorporando apoyos hasta llegar a lo que es hoy en día, una compañía electrónica de código abierto con un gran crecimiento.

Figura 14. Logo Arduino.

Al ser de carácter abierto (*open-source*), cualquier usuario que lo utilice puede contribuir a esta plataforma mediante la aportación de ideas, códigos de programación o proyectos. Esto le ha permitido que su evolución se haya producido rápidamente y que la comunidad haya crecido tanto debido a la facilidad de implementación y aprendizaje que presenta.

Las características más reseñables para la utilización de las placas y equipos creados por Arduino son:

- Posesión de un entorno de desarrollo integrado propio (IDE de Arduino), es decir, un conjunto de herramientas de programación, además de un lenguaje de programación simple pero eficaz para su utilización por todo tipo de usuarios.
- Son multiplataforma, es decir, el software Arduino es capaz de ejecutarse en diferentes SO como son Mac, Linux y Windows.
- Tiene un software de código abierto en el que programadores experimentados pueden, mediante bibliotecas C++, dar una mayor utilidad a los proyectos.
- Se trata de hardware libre, con lo cual, los planos de todas las placas y elementos Arduino son publicados, y pueden ser utilizados por diseñadores para crear sus propias versiones de placa y expansiones.
- El bajo costo que presentan frente a otras plataformas en relación con las características que aportan, lo hacen realmente atractivo.

La primera placa computadora de Arduino fue la Arduino Uno de la que se hablará más adelante, pero esta compañía ha desarrollado muchas más. Estas difieren en gran cantidad de aspectos, como pueden ser el tamaño, la cantidad de entradas y salidas, la capacidad de los microcontroladores, las formas de comunicación... También han desarrollados diferentes expansiones y kits que mejoran las funcionalidades básicas.

En la actualidad, Arduino ofrece los productos que se observan en la figura 15.

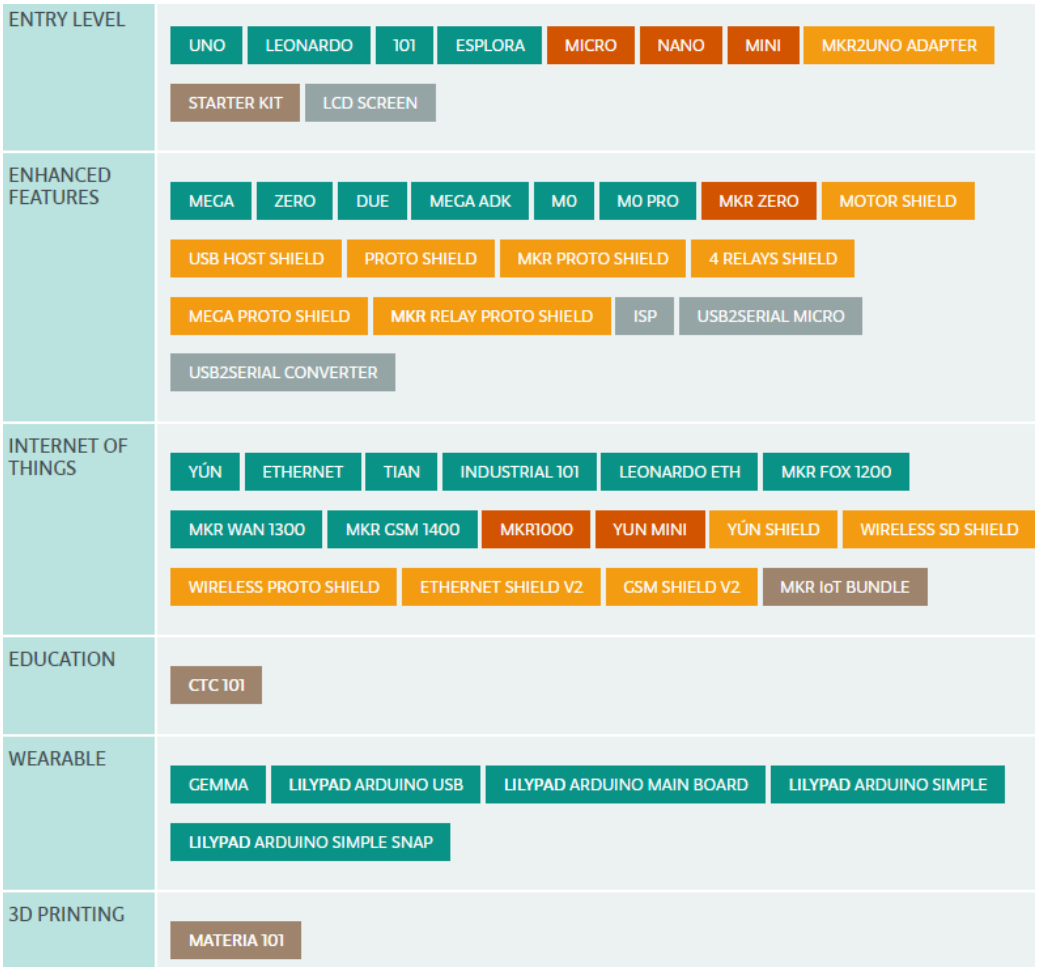

Figura 15. Productos Arduino.

Todas estas placas computadoras de Arduino, a diferencia de algunas de sus competidoras, llevan integradas unos microcontroladores AVR o ARM que no permiten la instalación de un sistema operativo en su interior.

- **Fundación Raspberry Pi (figura 16).** Se trata de una organización británica fundada en 2009 con la intención de estimular la enseñanza de la informática en las escuelas, aunque el proyecto fue ideado sobre el año 2006. El primer gran lanzamiento de una placa computadora de la fundación fue en 2012 con el modelo Raspberry Pi 1 Modelo A, aunque en ese mismo año llegaron sus versiones mejoradas Modelo B y Modelo B+ [18].
  Sus productos no se consideran de hardware libre, por lo que mantienen el control de estos, pero permiten su libre uso tanto a nivel educativo

como particular. En cambio, el software sí que es de código abierto, ya que utiliza una versión adaptada de sistema operativo basado en GNU/Linux "Debian", llamado "Raspbian". Al igual que las placas Arduino, estas también son multiplataforma. Soportan diferentes lenguajes de programación, aunque el más extendido para estas placas es el Phyton.

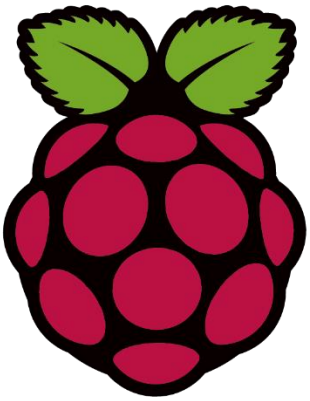

Figura 16. Logo de Rasberry Pi.

A diferencia de Arduino, las placas Raspberry Pi están gobernadas por un SoC Broadcom que comprende en su interior una unidad de procesamiento central ARM, una unidad de procesamiento gráfico, una unidad de procesamiento de señales y una memoria SDRAM mucho mayor que la que poseen los Arduino. Respecto al almacenamiento, las Rasberrys Pi no tienen una memoria ROM o similar incorporada, sino que por medio de una ranura para SD o MicroSD es como se dota a la placa del sistema operativo y de la capacidad de almacenamiento. Al poseer una GPU, para poder interaccionar y realizar funciones relacionadas con el procesado de vídeo, las placas poseen salidas de vídeo y audio [19].

Las Raspberry Pi que actualmente oferta en el mercado la fundación aparecen en la figura 17, aunque algunas comparten diseño, las características de la placa difieren en aspectos como pueden ser, el SoC (capacidad de la memoria RAM, la CPU, el tipo de juego de instrucciones 32 bits o 64), almacenamiento en SD o MicroSD, la presencia o no de conectividad WiFi y/o Bluetooth, el consumo energético…

Todas estas placas, al igual que Arduino presentan un coste muy reducido y son empleadas en numerosos proyectos en todos los ámbitos de la electrónica.

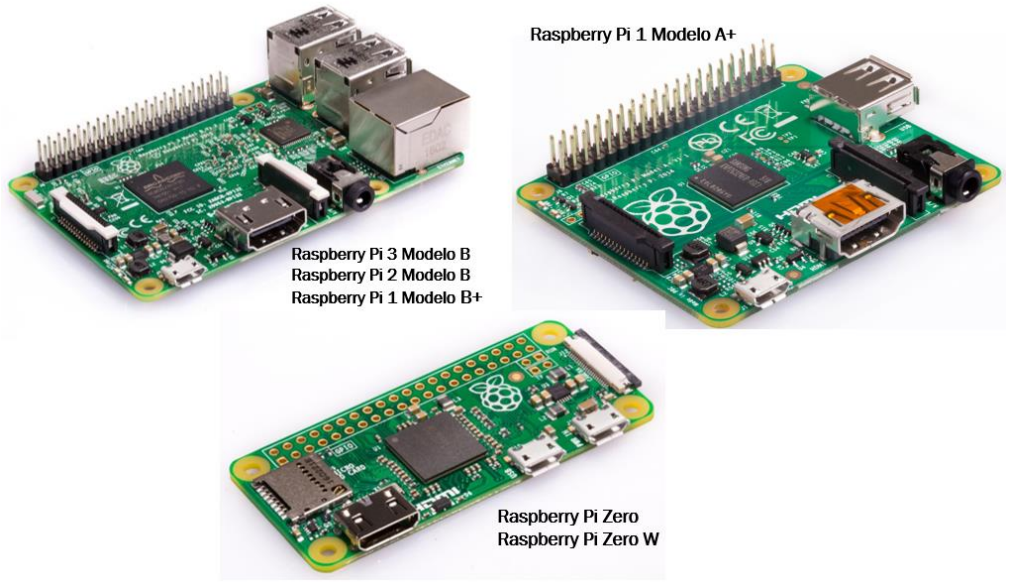

Figura 17. Modelos de Raspberry Pi.

Entre las placas de desarrollo más destacadas se encuentran los siguientes:

- Arduino Uno (figura 18). De la compañía Arduino, se trata de una placa de iniciación en el mundo de la electrónica para la realización de proyectos multidisciplinares de no muy alto nivel. Es la placa más utilizada y de la que más documentación existe de toda la familia Arduino. Presenta pines para la comunicación serie, soporta comunicaciones I2C y SPI, control por PWM y uso de interrupciones, permite alimentarlo con una fuente externa a la tensión que se desee siempre y cuando no exceda los límites, presenta convertidores A/D que favorecen la toma de datos…

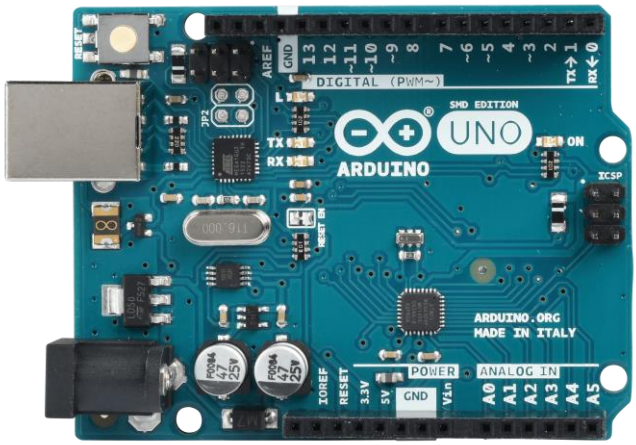

Figura 18. Arduino Uno.

Sus características más destacables se detallan en la tabla 2.

Tabla 2. Características de la placa Arduino Uno.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm

Como inconvenientes pueden destacar el limitado número de pines, la ausencia de ranura para SD o MicroSD que podría solventarse con un adaptador y su escasa memoria.

- Arduino Mega 2560 (figura 19). También del fabricante Arduino, ésta placa ya no es considerada de nivel de iniciación, sino que representa un nivel intermedio. Es una versión mejorada del Arduino Mega inicial que poseía una menor memoria flash.

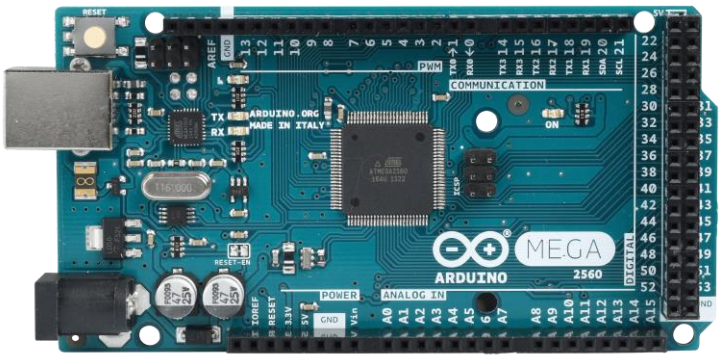

Figura 19. Arduino Mega 2560.

Sus características principales se observan en la tabla 3.

Tabla 3. Características de la placa Arduino Mega 2560.

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm

Como puede observarse, a diferencia del Arduino Uno, tiene un mejor procesador y presenta mayor memoria, tanto RAM como ROM, el tamaño de esta placa es mayor y, posee una mayor cantidad de pines (tanto analógico como digitales) que permite que se puedan conectar un mayor número de elementos. Además, el hecho de tener más pines permite la multiplexación dotándolos de más funcionalidad respecto a comunicación o control (UART, SPI, I2C, PWM, interrupciones…). En el resto de los aspectos y componentes es prácticamente igual a un Arduino Uno.

Por todo ello, es recomendada para su implementación en impresoras 3D y proyectos robóticos, y presenta un precio más elevado en el mercado.

- Intel Edison kit Arduino (figura 20) [20]. Esta placa sigue la misma filosofía que las placas Arduino, con la salvedad de que en vez de un microcontrolador, incorpora un SoC Intel de 32 bits. Se trata de una placa de prestaciones físicas similares a las del Arduino Uno debido al

número de pines, pero con una capacidad de procesado mucho mayor y destacando la presencia de un módulo de comunicaciones que incluye WiFi y Bluetooth. Se pensó principalmente para los emprendedores y profesionales interesados en el llamado "Internet de las cosas IoT".

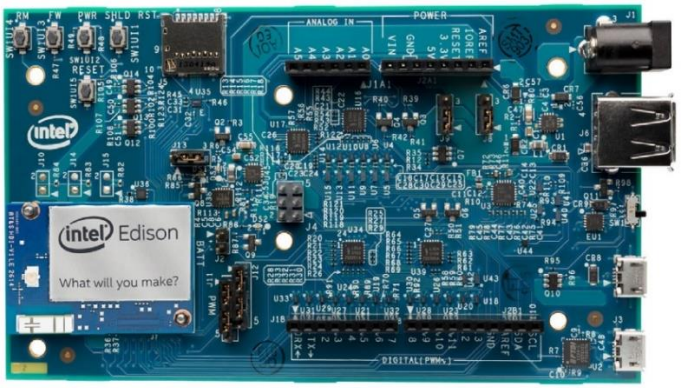

Figura 20. Intel Edison kit Arduino

Esta placa puede actuar como dispositivo o como host, albergando un sistema operativo. Al ser una placa oficial desarrollada conjunto a Arduino, la forma de programar para ella se podrá realizar de la misma manera, aunque habrá que descargar la librería correspondiente. En la tabla 4 se muestran sus principales propiedades.

Tabla 4. Características Intel Edison kit Arduino.

SoC	Dual-core, dual-threaded Intel® Atom™ CPU and a 32-bit Intel® Quark™ microcontroller
Operating Voltage	3.3V/5V
Input Voltage	7-15V
Digital I/O Pins	20 (of which 4 provide PWM output at a time)
Analog Input Pins	6
DC Current per I/O Pin	20 mA
Flash Memory	4 GB eMMC
RAM	1 GB LPDDR3 POP
Clock Speed	500 MHz (Intel® Atom™ CPU) 100 MHz (Quark™ microcontroller)
LED_BUILTIN	13
Features	WiFi, Bluetooth LE, MicroSD
Length	127 mm
Width	72 mm

- **RaspBerry Pi 3 Modelo B (figura 21) [19] [21].** De la fundación Raspberry Pi, se trata de una placa computadora de un nivel mucho mayor a la Arduino Uno o Mega. Se puede programar en diferentes lenguajes, posee un módulo de gestión de video y audio, permite funcionar como un pequeño ordenador debido a su SoC…

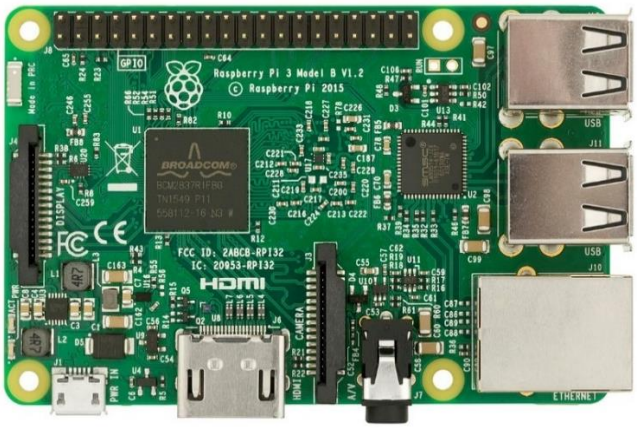

Figura 21. Raspberry Pi 3 Modelo B.

El principal inconveniente de este producto radica en la inexistencia de puertos analógicos ni convertidores analógico/digitales. En la tabla 5 se pueden observar las características más destacadas de este modelo.

Tabla 5. Características principales de la Raspberry Pi 3 Modelo B

SoC	Broadcom BCM2837 (Processor + GPU + Memory + Wifi and BLE module)
Processor	64-bit quad-core ARM Cortex-A53
GPU	Dual core Videocore IV® Multimedia co-processor
Operating Voltage	3.3V/5V
Input Voltage	5V
Pins	40 (26 GPIO pins, rest power and grounds pins
Connectors	4x USB ports, MicroUSB, CSI-2 (Camera), DSI (Display), RCA HDMI
Input Current	2.5 A
Flash Memory	Depends on MicroSD
RAM	1 GB LPDDR2
Connectivity	WiFi, Bluetooth LE (Low Energy), HDMI, Ethernet
Dimensions	85 x 56 x 17 mm

2.2.2 Sensor de irradiancia solar

Sabiendo que la irradiancia solar es una magnitud que describe la potencia incidente por unidad de superficie (W/m^2) generada por la radiación electromagnética del sol, la necesidad de obtener su valor ha hecho que, a lo largo de los años se hayan desarrollado diversos útiles que puedan cuantificarlo. En el caso de este TFG, estos elementos nos deberán proporcionar tensiones pequeñas para que el microcontrolador pueda manejarlas. Entre los instrumentos de medida principales destacan los siguientes:

- Piranómetro térmico (figura 22) [22]. Se trata de un elemento capaz obtener la medida de radiación solar global (directa y difusa o reflejada) de un gran intervalo de longitud de onda y con un campo de amplitud de 180°. En este, la radiación incide sobre dos cúpulas semiesféricas de vidrio y llega a una pila termoeléctrica (figura 23) formada por una serie de termopares (dos metales distintos), que captan el calor generando una diferencia de potencial proporcional a la diferencia de temperatura de los extremos de los termopares. Esa pila se encuentra pintada de negro con el objetivo de absorber la máxima radiación posible. Existen variantes del piranómetro térmico en los que la termopila esta bañada en diferentes pinturas, diferenciando zonas calientas o frías que hacen que la respuesta del piranómetro varíe en un pequeño porcentaje.

  Con el fin de medir la radiación difusa únicamente, se puede tapar el sensor de radiación directa con una pantalla parasol. Además, también se puede filtrar el rango de frecuencias por medio de un filtro en el domo de vidrio. En este caso, la cúpula filtra las radiaciones en el espectro de radiación solar, de 300 a 3000 nm.

  Su ubicación ha de ser en zonas abiertas, para evitar elementos que puedan obstruir la radiación en el dispositivo. En algunos casos, estos se encuentran montados en seguidores solares, con el fin de obtener la mejor medida de los datos situando el piranómetro perpendicular a los rayos solares.

Figura 22. Piranómetro.

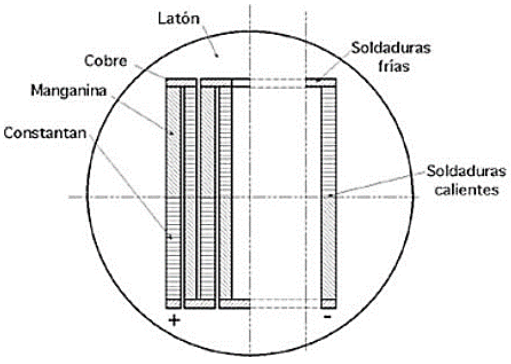

Figura 23. Pila termoeléctrica.

Este piranómetro es el más utilizado comercialmente, caracterizado por una alta precisión y una sensibilidad "plana" (le afectan de menor forma las condiciones externas como nubes o contaminación).

- Piranómetro de fotodiodo (figura 24) [23]. En éstos el principio de funcionamiento no se basa en el calor, sino que tiene fundamento el efecto fotoeléctrico. Se sustituye la pila termoeléctrica por un diodo fotosensible que produce una corriente proporcional a la irradiancia, y mediante un circuito de salida genera una tensión proporcional a la corriente. Esa tensión, al igual que en el apartado anterior, permite conocer la radiación.

Figura 24. Piranómetro fotodetector.

El espectro solar que admite es mucho menor que en los piranómetros de pila, aunque también esa admisión está relacionada con el material del que está fabricado el fotodiodo (siendo el más común el de silicio) y su respuesta, depende en mayor proporción de las condiciones externas del medio.

- Célula de referencia calibrada (figura 25). Se basa en la utilización como sensor de una célula fotovoltaica. Al igual que los anteriores también generan una corriente y dependen de las condiciones externas, aunque presentan una rápida respuesta, ligereza y menor coste, lo cual hace que hayan ido ganando terreno a los piranómetros convencionales.

Figura 25. Célula solar calibrada.

En las células calibradas más modernas, se incorporan componentes de compensación de temperatura.

- Pirheliómetro (figura 26) [22]. Este instrumento se utiliza para medir la radiación directa de un haz de rayos solares. Tiene forma telescópica y una pequeña abertura por la que se cuelan los rayos y se dirigen sobre una termopila en la base del pirheliómetro que trabaja como la de un piranómetro. Éste se ha de montar sobre un seguidor solar y colocarse en todo momento perpendicular a la dirección de los rayos solares.

Figura 26. Pirheliómetro.

Existen otros elementos como los pirgeómetros (miden radiación solar infrarroja), o los albedómetros (formados por dos piranómetros en contraposición que permiten medir la radiación neta), pero estos son menos frecuentes.

Estos elementos se suelen encontrar de forma conjunta, y entre sus aplicaciones principales destacan las observaciones meteorológicas y climáticas científicas, la investigación de ensayos de materiales y la evaluación de la eficiencia en captadores solares y fotovoltaicos.

Con el fin de elegir la mejor opción se debe prestar atención a los parámetros que se detallan a continuación:

- Sensibilidad. Magnitud mínima que el sensor es capaz de medir para tener una salida detectable (mínima irradiancia para la que se obtiene un voltaje válido).
- Rango de trabajo. Valores mínimos y máximos a los que es capaz de trabajar el sensor (temperatura máxima y mínima a la que puede trabajar el sensor y rango espectral que admite).
- Precisión. Se considera el error entre el valor real y el obtenido.
- Resolución. Mínima variación de la entrada para la que se obtiene una variación en la salida del sensor.
- Tiempo de respuesta. Es el tiempo transcurrido entre que cambia la señal de entrada (irradiancia) y la señal que reporta el sensor (salida).
- Offset. Se trata de un factor de corrección que se debe tener en cuenta a la hora de hallar el valor final del dato que se desea obtener. En muchos casos es cero (lo ideal), pero en otros no.
- Dependencia de la temperatura. Variación de la salida en relación con la temperatura.

2.2.3 Sensor de temperatura

Las características del sensor de temperatura en caso de este TFG no son muy exigentes. Este se deberá conectar a un microcontrolador y nos proporcionará el valor correspondiente en milivoltios (mV). Entre los más utilizados se encuentran los siguientes [24]:

- Termopar (figura 27). Se trata de un sensor termoeléctrico de contacto, es decir, ha de estar en contacto con el material del que se quiere tomar la temperatura. Como ya se ha comentado en el piranómetro, el termopar es la unión de dos hilos metálicos distintos y aislados, y la temperatura de esa unión caliente es la que genera un voltaje muy pequeño entre los extremos de los dos metales (unión fría).

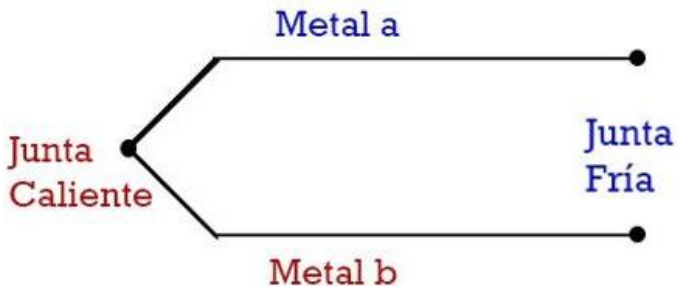

Figura 27. Esquema de un termopar.

Según el tipo de unión (soldada en extremos, en paralelo, trenzada, expuesta), y su encapsulamiento o ausencia de este, los termopares darán mejores o peores prestaciones dependiendo del medio y la función a la que vayan dedicados.

Estos termopares son económicos, pequeños, no requieren de alimentación y aceptan un gran rango de temperaturas (alrededor de unos 2000°C). El hecho de proporcionar una salida tan baja, hace que sean muy sensibles al ruido. Estos también requieren de calibración, lo que hace más complejo su uso.

Existen varios tipos designados por letras según la composición química de los metales, y sus envolventes según la norma aplicada en el territorio, son de colores diferentes.

- <u>Detector de temperatura resistivo (RTD).</u> En este caso, el principio de funcionamiento se basa en la variación de la resistencia del metal conductor que constituye el sensor conforme varía la temperatura. A mayor temperatura, mayor agitación de los electrones y por lo tanto mayor resistividad, y viceversa. Los elementos más utilizados para la construcción de estos son el platino (muy estable y de gran exactitud), el níquel y el cobre. Según su construcción, estos sensores pueden ser bobinados, de película fina, enroscados o de anillo hueco como se observa en la figura 28. A su vez, también pueden presentar dos, tres, o cuatro hilos dotándoles de mayor exactitud a mayor número de alambres. Entre las ventajas más destacadas se encuentran su alta estabilidad, linealidad, mayor sensibilidad que los termopares… Como inconvenientes se observan su menor velocidad de respuesta, su inestabilidad ante vibraciones y les afecta el autocalentamiento.

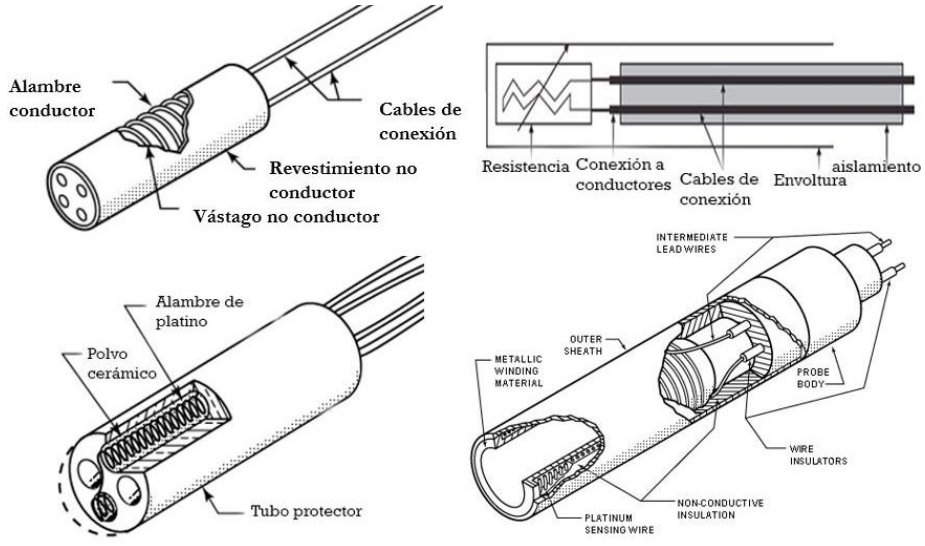

Figura 28. Tipos de RTD según construcción.

El aspecto exterior de las RTD depende de su uso. Por ejemplo, para inmersión en fluidos su aspecto es muy parecido o idéntico a las sondas de los termopares, sin embargo para un contacto superficial, suelen ser de película fina envueltos por elementos flexibles para poderse adaptar.

- **Termistor.** Se trata de otro sensor resistivo con el mismo principio que el anterior, la resistencia del material varía con la temperatura. Los termistores son más económicos, pequeños y sensibles que los RTD; destacan su no linealidad frente a la de los RTD y un rango de medición de temperatura menor. Las configuraciones en que se pueden encontrar estos son diversas, aunque destacan los de tipo disco, perla, de barra o axial y los SMD, que se pueden observar en la figura 29.

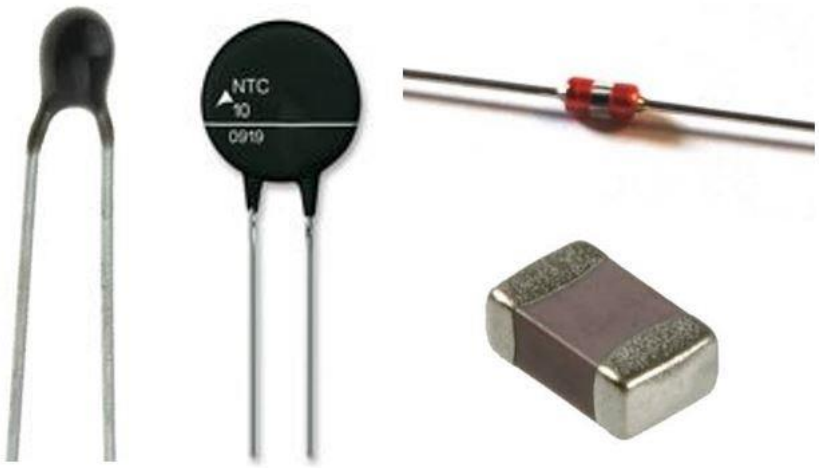

Figura 29.Tipos de termistor según construcción.

Existen de dos tipos según su respuesta a la variación de temperatura. Los NTC (Coeficiente de temperatura negativo) en los cuales a medida que aumenta la temperatura, disminuye la resistencia del conductor, y viceversa. Los PTC (Coeficiente de temperatura positivo) que funcionan de forma que, ante un aumento de la temperatura, su resistencia aumenta también, y ante una disminución, su resistencia se ve reducida.

- **Sensor de temperatura de silicio**. Son circuitos integrados basados en propiedades térmicas semiconductoras que utilizan la variación de voltaje de la unión base-emisor (Vbe) de los transistores bipolares. No requieren de circuitos acondicionadores externos (de linealización, amplificación, compensación) como pueden requerir los otros, ya que los incorporan en su interior [25]. En la figura 30 se puede observar un esquema simplificado de un sensor de este tipo.

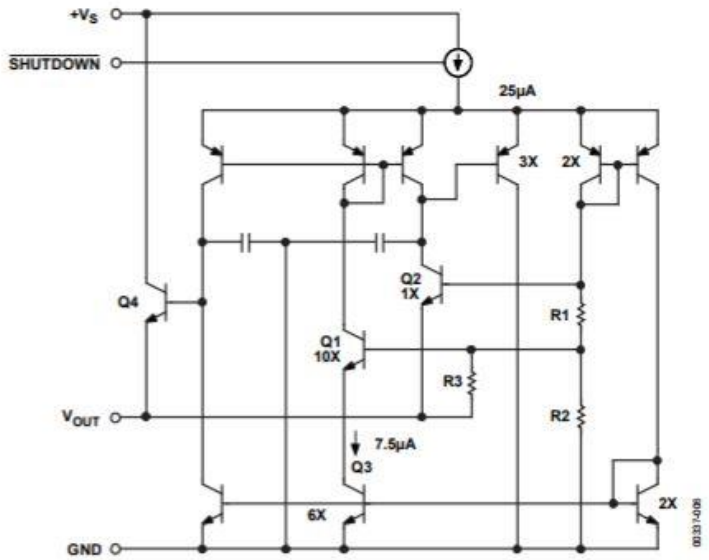

Figura 30. Esquema simplificado de un sensor de temperatura de silicio.

La salida en tensión o corriente varía linealmente con la temperatura. Existen algunos, incluso que proporcionan directamente una salida digital. Son exactos, con alta sensibilidad, de pequeño tamaño y un rango de temperatura de aproximadamente unos 200°C. Se han de alimentar a una tensión concreta para el correcto funcionamiento del dispositivo y su corriente es reducida lo que evita el autocalentamiento. Existen muy variados encapsulados para su montaje.

Los sensores de temperatura se utilizan en gran cantidad de aplicaciones tales como son la climatización, dispositivos médicos, manipulación de productos químicos y alimenticios, estaciones meteorológicas y plantas fotovoltaicas, control de procesos con el fin de proporcionar seguridad en el funcionamiento de máquinas o dispositivos…

Al igual que en el apartado anterior, con el fin de elegir el dispositivo más adecuado hemos de prestar atención a los siguientes parámetros:

- **Sensibilidad.** Magnitud mínima que el sensor es capaz de medir para tener una salida detectable (mínima temperatura para la que se obtiene un voltaje válido).
- **Rango de trabajo.** Valores mínimos y máximos con los que es capaz de trabajar el sensor (temperatura máxima y mínima a la que puede trabajar el sensor).
- **Precisión.** Se considera el error entre el valor real y el obtenido.
- **Resolución.** Mínima variación de la temperatura para la que se obtiene una variación de voltaje por el sensor.

- **Tiempo de respuesta.** Es el tiempo transcurrido entre que cambia la señal de entrada (temperatura) y la señal que reporta el sensor (salida).
- **Offset.** Se trata de un factor de corrección que se debe tener en cuenta a la hora de hallar el valor final del dato que se desea obtener. En muchos casos es cero (lo ideal), pero en otros no.

2.2.4 Fuente de alimentación

Las formas de alimentar eléctricamente el dispositivo que se va a crear como objetivo de este trabajo son variadas. Todo dependerá de las condiciones de uso y del tipo de microcontrolador o placa computadora que se utilice. Se puede realizar a través de corriente alterna (CA) convirtiéndola seguidamente en corriente continua (CC) por medio de un adaptador, o directamente por corriente continua:

- **Adaptador de corriente AC/DC (figura 31).** Este método es el más fiable. Se conecta a la red de corriente alterna y el adaptador se encarga de convertirla en corriente continua por medio de su circuito interno.

Figura 31. Adaptador AC/DC.

Los circuitos más comunes son [26]:

- Fuente lineal con aislamiento galvánico (figura 32): Compuesta por un transformador, un rectificador, un filtro y un regulador. Primeramente, el transformador reduce el voltaje y proporciona aislamiento entre la red y el circuito, después el rectificador convierte la señal negativa alterna en positiva por medio de un puente de diodos, seguidamente un filtro y un regulador estabilizan el voltaje al valor deseado.

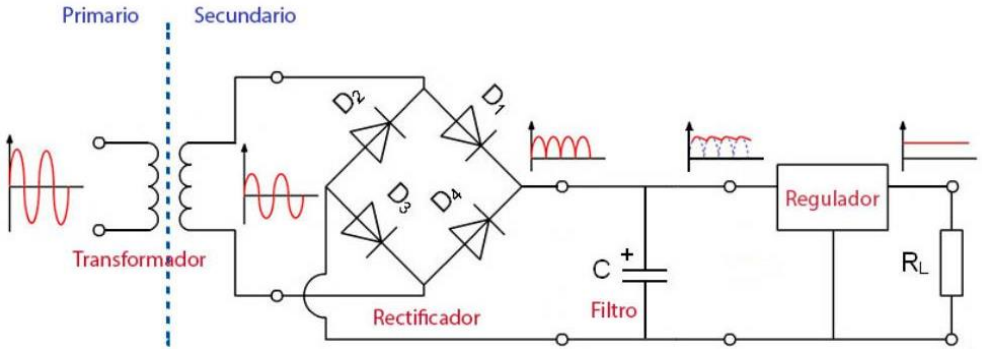

Figura 32. Fuente lineal.

- Fuente conmutada (figura 33): En este caso la energía se transforma mediante transistores en conmutación. Su esquema se detalla en la siguiente figura:

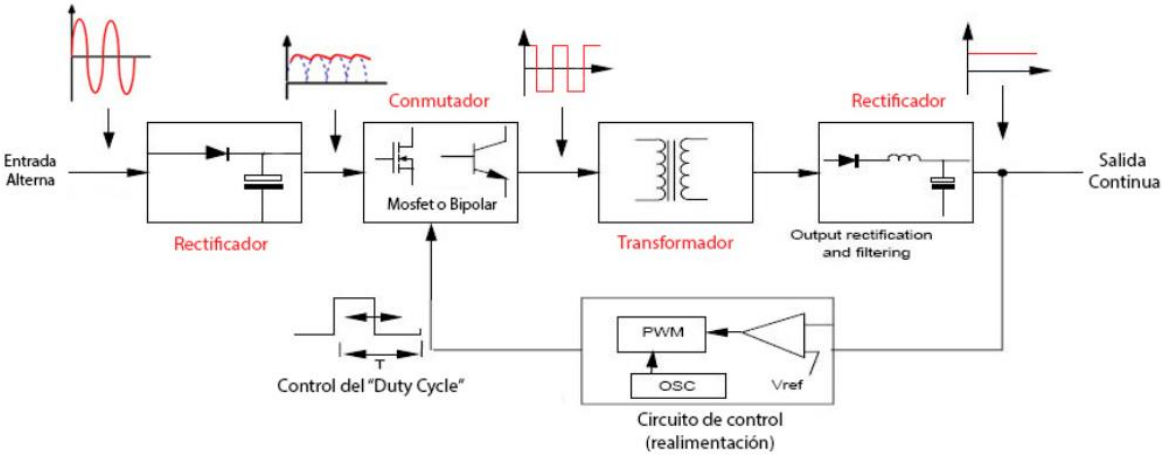

Figura 33. Fuente conmutada.

En primer lugar el rectificador transforma la señal alterna en continua. Seguidamente, el conmutador formado por transistores MOSFET o bipolares trabajando en corte-saturación abrirán y cerrarán el circuito a una gran frecuencia gracias a un circuito de control. El transformador realiza la misma función que en las fuentes lineales, pero va a trabajar con frecuencias mucho mayores y con señal cuadrada, lo que permite que el transformador sea más pequeño. Por último el segundo rectificador transformará la señal cuadrada en una señal estable y continua.

Las fuentes lineales otorgan simplicidad, bajo ruido a la salida, componentes menos costosos a bajas potencias y una respuesta dinámica rápida. Su principal inconveniente es el tamaño y la baja eficiencia.

Respecto a las fuentes conmutadas destacar que, al contrario que las lineales, tienen una alta eficiencia y un menor tamaño de transformador, reduciendo el coste y el peso. Su desventaja es la mayor complejidad de su diseño, así como que el ruido producido es mayor, lo que puede dificultar el control, por ello se deben utilizar más filtros [27].

Con la cantidad de adaptadores existentes, se tiene la posibilidad de alimentar el dispositivo casi como se desee respecto a tamaño, seguridad, voltaje y corriente de salida se refiere.

- Batería (figura 34). Estos dispositivos son los ideales para proveer de energía a la placa y demás elementos de forma autónoma, es decir, sin necesidad de estar conectados a la red para suministrarla. Proporcionan directamente corriente continua por lo que no tienen que transformarla. Su funcionamiento se basa una reacción química de oxidación-reducción. La batería está formada por dos electrodos metálicos sumergidos o incrustados en un electrolito (conductor de iones) que al reaccionar entre ellos, en el ánodo se produce la oxidación (liberación de electrones), y en el cátodo la reducción (un defecto de electrones). Al pasar los electrones libres del ánodo al cátodo se produce por el circuito externo una corriente eléctrica. Hay baterías que permiten la recarga, y esta ocurre cuando la reacción se produce en sentido inverso (oxidación en el cátodo y reducción en el ánodo), para lo cual se ha de aplicar una diferencia de potencial [28].

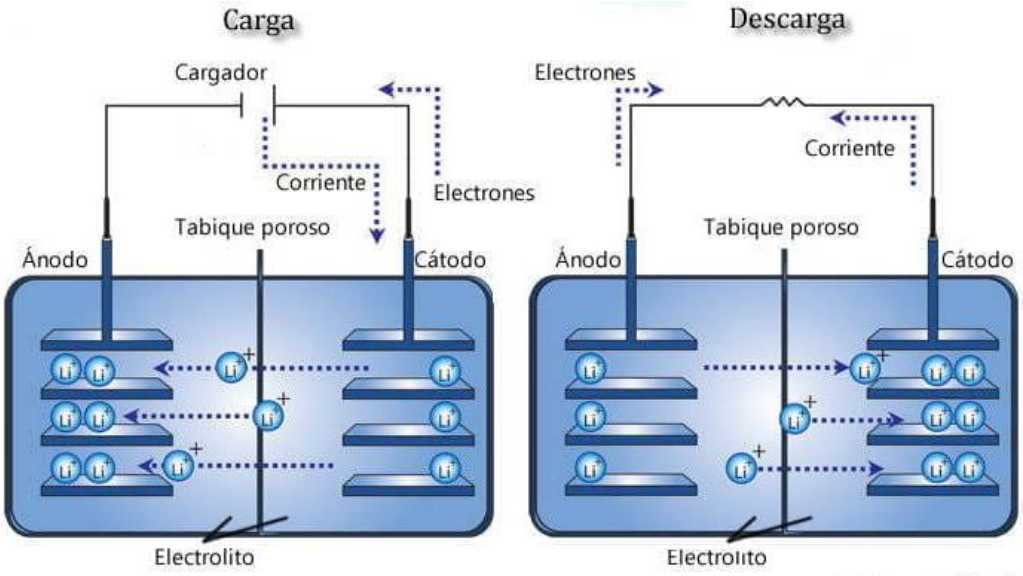

Figura 34. Carga y descarga de una batería.

En la figura 34 se puede observar el funcionamiento descrito anteriormente. Se corresponde con una batería de iones de litio, pero su funcionamiento es extensible a las demás.

Existen muchos tipos de baterías y composición de estas, las más comunes para las placas computadoras son las siguientes [29]:

- Baterías Li-ion (figura 35). En este caso el electrolito está formado por una sal de litio contenida en un líquido orgánico. Son recargables y tienen un muy bajo efecto memoria (formación de microcristales que reducen la capacidad de la batería debido al sobrecalentamiento a la hora de cargar, o la realización de cargas incompletas).

Figura 35. Batería Li-ion.

Debido al auge de la telefonía y la necesidad de recargar los móviles cuando no se tiene una red eléctrica cerca, en los últimos años, se han puesto muy de moda las baterías portátiles (figura 36) que proporcionan 5V regulados. Estas están son también baterías de iones de litio y tienen las características necesarias para alimentar una placa computadora. Pueden presentar la desventaja de que la intensidad máxima que proporcionan es reducida, pero para el uso en este TFG sería indiferente.

Figura 36. Powerbank.

- Baterías LiPo (figura 37). Se trata de baterías en las que la sal de litio está contenida en una especie de gel (un compuesto polimérico), pero trabajan de la misma forma que las baterías Li-

ion. Existen de diferentes voltajes según el número de celdas en paralelo. Las más comunes son las de 2 y 3 celdas (2S y 3S) respectivamente. A mayor número de celda mayor voltaje aportan. Estas, al igual que las Li-ion, son ligeras y presentan una densidad de energía alta y una alta eficiencia en comparación a otras baterías de diferente composición (como pueden ser las NiCd o las NiMH), aunque también son ligeramente más caras. Son recargables.

Figura 37. Batería LiPo.

De la misma forma que las Li-ion, se ha de tener cuidado con su manipulación debido a que, si se calienta en exceso, se sobrecarga o se daña, puede combustionar.

- Pilas convencionales (Figura 38). Estas son utilizadas para pequeños proyectos que no exijan grandes requerimientos. Existen de diversas composiciones y para conseguir un mayor voltaje se han de conectar en serie. No otorga grandes corrientes y algunas no son recargables.

Figura 38. Pilas y portapilas.

El principal inconveniente de las baterías es el agotamiento. Debido al paso del tiempo o utilización del dispositivo las baterías acaban descargándose, por lo que hay que estar atento en caso de que esto ocurra. Si las baterías son recargables el problema es menor, ya que como se ha dicho, se pueden recargar, pero si no lo son, el coste del dispositivo será mayor a lo largo del tiempo. Por ello, la utilización de adaptadores de corriente es más segura, pues proporcionan corriente en todo momento, con el inconveniente de que no siempre tienes cerca la red eléctrica.

La elección entre todas estas posibilidades dependerá de los parámetros que más se adecuen a las necesidades de nuestro dispositivo. Conviene enfocar la mirada en los siguientes aspectos:

- Voltaje de salida. La tensión en la salida ha de ser la adecuada para que la placa computadora trabaje correctamente y no se sobretensione.
- Intensidad de corriente de salida. Al igual que el voltaje, la corriente ha de ser la suficiente para suministrar energía a todos los componentes sin exceder el límite que le pueda perjudicar.
- Capacidad. Refiriéndose aquí a las baterías, la intensidad de corriente que puede suministrar a lo largo del tiempo hasta agotarse. Se suele expresar en mAh. Cuanto mayor sea, mayor tiempo de funcionamiento tendrá el dispositivo, pero a la vez el tamaño y precio de la batería será mayor.
- Seguridad. En este caso se tendrá que tener en cuenta que dispositivos pueden o no producir una descarga eléctrica con mayor facilidad, o cuales pueden contaminar en mayor cantidad. También se ha de tener en cuenta el riesgo de ignición.
- Conexión a la placa computadora. Se habrá de observar la conexión que se hará entre la fuente y la placa computadora. No todos los elementos se pueden conectar de la misma manera y con los mismos conectores.

Capítulo 3: Marco de trabajo y estado del arte.

En este apartado se procede a seleccionar los elementos nombrados en el capítulo anterior, los que más convengan para la construcción del dispositivo y en concordancia con lo solicitado por el promotor del proyecto. De todos ellos se dará una explicación de su elección y se realizará una descripción detallada.

3.1 Placa computadora Intel Edison kit Arduino

Ante la necesidad de obtener magnitudes físicas del medio ambiente (irradiancia y temperatura) y poder procesarlas, se ha visto la necesidad de utilizar una placa computadora que incorpore pines analógicos y convertidores analógicos-digitales. A su vez, ante la necesidad de almacenar los datos obtenidos en un elemento de memoria, se ha optado por la placa que incorporaba la ranura MicroSD con su correspondiente conexionado.

Al ser un proyecto que no necesita de grandes exigencias, se ha optado por descartar la opción de la Raspberry Pi 3, ya que ésta incorporaba diversos elementos que no se aprovecharían como puede ser el módulo de gestión de vídeo y audio, conectores varios como HDMI, RCA, varios USB innecesarios para el cometido del proyecto. También, el hecho de configurar el dispositivo es mucho más lento y complicado al tener que cargar el sistema operativo desde la tarjeta de memoria. Aunque el principal problema de la Raspberry Pi es la ausencia de pines y convertidores analógico-digitales, que hace que no se puedan utilizar la mayoría de los sensores directamente. Por esta razón, se tendría que utilizar otro dispositivo externo que proporcionara esa utilidad, aumentando el cableado y la dificultad de conexión de los elementos. También existen sensores que proporcionan una salida digital, pero estos son más difíciles de encontrar y de mayor precio.

Por otra parte, se ha descartado la utilización del Arduino Uno debido a que el microcontrolador presenta una escasa memoria (principalmente la SRAM para variables y datos). Una vez realizado el programa se intentó cargar en la placa obteniendo un error que indicaba que la memoria era insuficiente y que podía ser que el programa no realizara correctamente todas sus funciones, y así ha sido. Además, en esta placa a pesar de que sí incorpora los convertidores A/D, no posee el adaptador para SD, con lo que este tendría que ser externo complicando el montaje.

El Arduino Mega por su parte, presenta buenas condiciones para este proyecto, ya que tiene una gran cantidad de pines que permite sobradamente la conexión de todos los elementos necesarios para el proyecto. Además, el microcontrolador que incorpora satisface las necesidades de procesado y

memoria. El único inconveniente que tiene es la falta del adaptador SD para el almacenaje de los datos.

Las tres placas computadoras presentan unos precios reducidos que los hacen dispositivos realmente atractivos, pero debido a lo explicado anteriormente y también en consecuencia a que el Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales disponía de esta placa en desuso, se ha optado por utilizar la placa Intel Edison.

Esta placa, es comparada con la Arduino Uno debido al número de pines, pero como se ha comentado en el Capítulo 2, el microcontrolador es un SoC con una gran potencia y capacidad de memoria, con lo cual no tiene problemas a la hora de ejecutar el programa. Presenta un convertidor A/D para transformar las magnitudes físicas del medio a señales binarias. Este convertidor es un ADS7951 de Texas Instruments con 8 canales (2 no se utilizan) y una resolución de 12 bits aunque por defecto se encuentra limitado a 10 bits utilizándolo como un Arduino Uno. Por ello, la lectura que muestra presenta un valor entre 0 y 1024 (2^n bits) y puesto que los pines analógicos aceptan mínimo de 0V y un máximo de 5V, la diferencia entre cada valor medido será de:

$$5000\ mV/1024 = 4{,}83\ mV$$

Los pines pueden admitir un máximo de corriente de 40 mA pero lo normal es que trabaje a 20 mA. Esta placa puede ser alimentada por medio de una batería que se conecta al Jump 2 de la placa con hasta 4,3 V, por el pin *Vin* entre 7 y 15V, por un puerto microUSB hasta 5 V, o por el conector para adaptador de corriente entre 7 y 15 V. Cuando se tiene conectada la placa mediante el USB o el adaptador, si en el Jump 2 se encuentra conectada una batería, ésta se cargará.

La Intel Edison presenta además un módulo de WiFi y Bluetooh, que en este dispositivo no se va a utilizar pero que podría llegar a utilizarse en mejoras futuras, ya que puede llegar a ser muy útil con el cada vez más desarrollado Internet de las Cosas (IoT). [20]

3.2 Célula solar calibrada compensada Atersa

Como sensores de irradiancia se ha hablado del piranómetro térmico, del piranómetro fotovoltaico, de la célula fotovoltaica calibrada, y del pirheliómetro.

En primer lugar, se va a desechar la idea de utilizar un pirheliómetro. Esto se debe a que éste, únicamente mide la irradiancia solar directa y no la difusa proveniente de la reflexión. Además, es un elemento especializado, con

una gran sensibilidad y por ello de elevado precio, que ha de colocarse en un seguidor solar que no todas las instalaciones fotovoltaicas disponen.

Respecto a los piranómetros térmicos o solares, se tratan de elementos con una gran sensibilidad también, aunque aceptan un rango de radiación menor que las fotocélulas, y el rango de precios es amplio dependiendo de las características, siendo más caros los piranómetros térmicos, a la par que más precisos que los fotovoltaicos.

El hecho de que sean un poco más caros hace que en este proyecto se haya utilizado una célula solar calibrada compensada.

Se trata de la célula calibrada compensada del fabricante Atersa (figura 39) que presenta dos posibles configuraciones. La primera otorga dos señales de salida de 65 mV por cada kW/m^2 con un error de ±2.1% (relación tensión-radiación), mientras que la segunda presta una salida de 100 mV por cada kW/m^2 con un error de ±2.2%. Estas salidas únicamente dependen de la irradiancia solar, ya que vienen compensadas en temperatura internamente. A la placa se le añadirá un sensor de temperatura externo. Presenta unas dimensiones de 266 x 266 x 35 mm y un peso de 1.6 kg [30]. Su conexión con la placa se realizará por medio del puerto analógico A0.

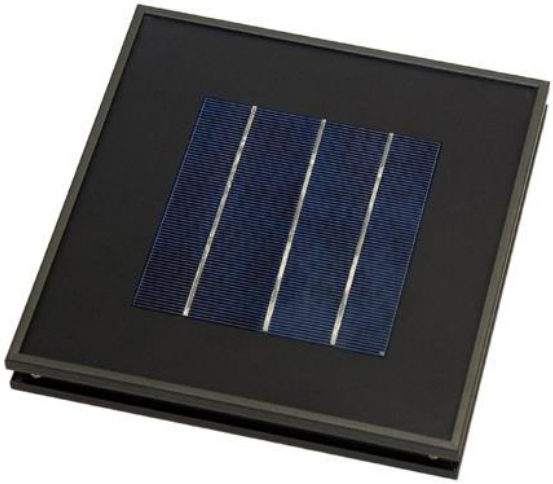

Figura 39. Célula calibrada Atersa.

3.3 Sensor de temperatura LM35

Hay varios sensores de temperatura disponible, como son los termopares, los RTD, los termistores y los sensores de temperatura de silicio. Para este trabajo se han escogido estos últimos.

Los termopares miden en un gran rango de temperaturas, y para el cometido del proyecto no es necesario un rango muy amplio. Además son muy sensibles al ruido al proporcionan una señal de salida muy baja, del orden de

µV, por lo que se necesitará de un sistema amplificador que permita a la placa detectar los cambios, ya que como se ha explicado estos se detectan cada 4,83 mV. Otro inconveniente es que requieren de calibración lo que hace más engorroso su manejo. Por todo ello se ha desechado esta opción.

En el caso de las RTD, determinar la temperatura también es una tarea compleja si no se dispone de un circuito de acondicionamiento como puede ser un puente *Wheatstone* y un amplificador, puesto que la variación de la salida es muy grande respecto a una pequeña variación en la entrada, es decir, la variación para un grado centígrado se produce con una variación de décimas de ohmios. Tiene una velocidad de respuesta lenta, y son afectados por el autocalentamiento. Todo ello ha hecho que tampoco se escoja este elemento como sensor de temperatura para el proyecto, a pesar de tener una gran estabilidad.

Los termistores son muy económicos y tienen mayor sensibilidad que los anteriores, así como un tiempo de respuesta bajo, pero el hecho de poseer poca masa provoca problemas de autocalentamiento. Además, al no ser lineales, se necesita acondicionar la señal mediante software aplicando modelos matemáticos que ralentizan la ejecución del programa significativamente.

El sensor de temperatura de silicio es el elemento idóneo para este proyecto. No requiere de acondicionamiento externo porque lo lleva incorporado. Son lo suficientemente exactos para este proyecto, con un rango de temperaturas amplio para lo que se va a utilizar, y al pasar por ellos una corriente pequeña, no se produce autocalentamiento. Además su montaje es muy cómodo. No necesitan de amplificación, aunque puede ser recomendable para mejorar la resolución.

Por ello se ha decidido usar el LM35 (figura 40) de la compañía Texas Instruments. Se encuentra calibrado directamente en grados Celsius, su salida lineal se corresponde con 10 mV por grado centígrado (°C) y presenta un rango de medición desde -55°C a 150°C.

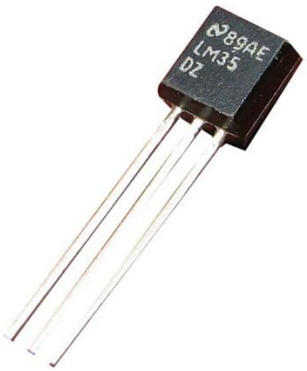

Figura 40. Sensor de temperatura LM35.

La corriente consumida es aproximadamente de 60 µA y tiene baja impedancia de salida. La tensión de operación oscila entre los 4 V y los 30 V, por lo que el conexionado se realizará a través de +5 V, tierra y la señal se conectará al pin analógico A1 y A2 [31].

3.4 Batería de pilas.

En el caso de la alimentación del dispositivo, ésta se va a poder realizar de diferentes maneras. Puesto que va a ser un elemento portátil que no va a tener normalmente una red eléctrica de baja tensión cerca cuando se esté utilizando, se va a rechazar la idea del suministro de energía principal por medio de un adaptador de corriente. Aun así, en caso de que se utilice, es prácticamente indiferente el tipo de fuente (lineal o conmutada) a usar, habría que fijarse principalmente en el voltaje y corriente que admite la placa y que aporta el adaptador (en este caso una tensión de entrada entre 7 y 15 voltios, y una corriente no mayor de 1 amperio), así como en el coste del adaptador.

Como se ha comentado, la placa Intel permite alimentarla por medio del Jack, y mediante el puerto MicroUSB. Se ha comprobado, que mediante el puerto microUSB, alimentándolo desde un ordenador la placa Intel funciona correctamente, pero se ha intentado utilizar una PowerBank, y hay momentos en los que funciona y otros en los que no.

Entonces, a la hora de alimentarlo, se ha decidido hacerlo por el Jack DC mediante la utilización de un portapilas con las correspondientes pilas. Para ello, a la hora de escoger la batería de pilas más adecuada para el trabajo, se va a buscar principalmente una batería recargable, que aporte una energía suficiente para alimentar al dispositivo durante el tiempo que se utilice, y que presente un bajo efecto memoria que no perjudique su capacidad. Además al utilizar la placa Intel Edison se recomienda el uso de una batería con electrolito de litio, que cumple con todas las necesidades anteriores. El problema es que estas pilas son más caras que las NiMH que tienen un mayor efecto memoria y una aportación de corriente más baja, aunque para el cometido del proyecto son suficientes. Por ello, se decide utilizar este tipo de pilas.

Se ha comprobado mediante la utilización de un polímetro que el consumo del dispositivo es de unos 115 mA (añadiendo un pequeño margen), por lo que para que el dispositivo esté en funcionamiento durante aproximadamente un día completo:

$$115\ mA \times 24\ horas = 2760\ mAh$$

La capacidad de la batería ha de ser de unos 2800mAh para asegurar el suministro deseado sin contratiempos. Por todo ello, la batería seleccionada

para alimentar el dispositivo se compone de 6 pilas EBL de alta capacidad. Sus características se presentan en la tabla 6 [32].

Tabla 6. Especificaciones EBL 2800.

Nombre	EBL High Capacity
Capacidad	2800 mAh
Composición	Ni-MH
Forma	Cilíndrica
Dimensiones	50 x 14 mm
Ciclos de carga	1200

3.5 Interruptor y botones

Para el encendido y apagado del sistema medidor, se incorporará en la parte trasera de la envolvente un interruptor (figura 41) que permitirá el paso de la corriente del adaptador o del portapilas desde el conector exterior al conector DC de la placa Intel.

Figura 41. Interruptor.

Por otro lado, el cuadro de mandos estará formado por 3 botones (figura 42) con embellecedores de color negro que permitirán la introducción de la fecha y hora en el dispositivo, y la selección de los modos de visualización por el display LCD (mostrar irradiancia, mostrar temperatura o encender o apagar el display para un menor consumo de energía). Estos elementos se colocarán en la parte superior del dispositivo justo por debajo del display y separados a una distancia adecuada para su cómodo manejo. Por todo ello, habrá que perforar la caja.

Figura 42. Pulsador.

Destacar que para la conexión de los pulsadores, se han utilizado resistencias internas *pull-up* de la placa (figura 43) permitiendo así ahorrar unas resistencias externas que se han de incorporar para que la corriente que circula no sea excesivamente alta y dañe los componentes.

De esta forma, cuando se pulsa el botón, la corriente fluye desde el pin a tierra por el pulsador, leyéndose un "0", y cuando no está pulsado la corriente va desde alimentación pasando por la resistencia que limita la corriente, hasta el pin donde se lee un "1". Se han utilizado los pines analógicos de la placa que estaban en desuso, ya que estos se pueden configurar también como pines digitales.

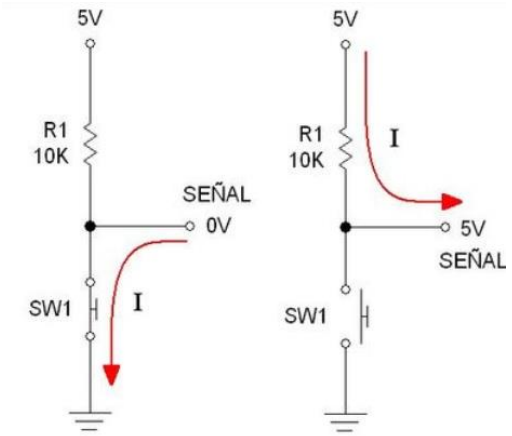

Figura 43. Configuración Pull-up.

Además, con la finalidad de eliminar cualquier posible rebote al pulsar o soltar el botón, se incorporará en paralelo con el pulsador un condensador que filtre la señal. De esta forma se evita que pueda producirse una lectura errónea por parte del dispositivo, la cual perjudicaría la experiencia de usuario.

3.6 Display LCD

Puesto que mediante el cuadro de mandos se realiza el control del dispositivo, es necesario poder visualizar los cambios que se produzcan y los mensajes que este envíe. Por ello se instalará un display LCD (figura 44) en la parte superior de la caja por medio de 4 tornillos y los correspondientes agujeros [33]. Éste posee 32 caracteres distribuidos en 2 filas y 16 columnas, cada uno de ellos formados por una matriz de 5x8 puntos o *dots*. Por medio de la programación se pueden crear caracteres que no están incluidos en su librería.

Figura 44. Display LCD.

El LCD está gobernado por un controlador el cual debe manipular varios pines a la vez, y su conexionado se resume en la tabla 6. Puede trabajar en 2 modos, uno de 8 bits que los transmite de golpe, y otro de 4 bits que los transmite en 2 mitades utilizando solo 4 pines.

Tabla 6. Descripción y conexionado LCD.

Pin del LCD	Descripción	Conexionado a la placa
VSS	GND	GND
VDD	Alimentación +5V	5V
VO	Ajuste de contraste de la pantalla	A una resistencia que va a tierra
RS	Control de los registros del LCD	A pin digital
R/W	Lectura o escritura	GND (escritura)
E	Habilitador de lectura o escritura	A pin digital
D0-D3	Bits del bus de datos	Al aire (se utiliza modo de 4 bits)
D4-D7	Bits del bus de datos	A pines digitales
A	Ánodo de los LEDs de retroiluminación	Pin digital a resistencia de 220Ω para limitar corriente a LEDs
K	Cátodo de los LEDs de retroiluminación	GND

3.7 Led RGB

En el proyecto se ha añadido una funcionalidad en la que un led RGB de cátodo común (imagen 45) emitirá diversos colores, en orden a un rango de valores asignado según la irradiancia recibida por la célula en ese momento.

Este led es la unión de 3 LEDs de los colores básicos rojo, verde y azul, los cuales comparten el cátodo. La tensión umbral típica para el led de color rojo es de 1,95 V, la del verde de 3,3 V y la del azul de 3,3 V [34].

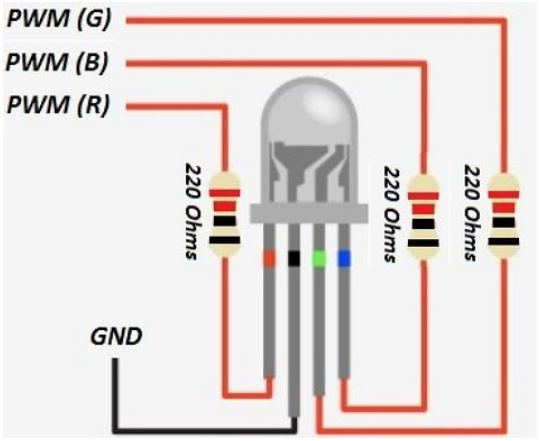

Figura 45. LEG RGB de cátodo común.

El conexionado se realizará a los pines digitales capaces de modificar la anchura de los pulsos que emiten (PWM), con el fin de poder dotar de diferentes tonalidades a la luz emitida por el RGB. Además se utilizarán unas resistencias de 220 Ω entre el pin digital de la placa y la patilla de color del led para limitar la corriente y no dañarlo, mientras que el cátodo se conecta a tierra.

3.8 Envolvente Schneider Electric

Ante la necesidad de proteger todos los elementos que conforman el dispositivo, y poderlos transportar de una forma cómoda y segura, se ha buscado una envolvente que los albergue.

Ésta se trata de una caja de plástico ABS de color gris del fabricante Schneider Electric como la de la figura 46, la cual aporta un grado de protección IP66 (resistente a polvo y chorros de agua), y unas dimensiones de 241 mm de longitud, 194 mm de anchura y 87 mm de altura. La caja consta de 4 tornillos de plástico que son los que ajustan y cierran la tapa. En esta caja se realizarán los agujeros pertinentes que permitan llevar los cables desde la placa a la célula solar y al sensor de temperatura, así como los correspondientes a los botones que permiten controlar el dispositivo, al display LCD, al led RGB y al interruptor de encendido y apagado del mismo [35].

Figura 46. Envolvente del dispositivo.

Capítulo 4: Programación Arduino.

En los anteriores capítulos se ha hablado y explicado el hardware del dispositivo, su funcionamiento y el porqué de su elección. Ahora, en este capítulo se procede a explicar el software creado, que será el que ejecute el microcontrolador de la placa computadora.

Se describirán todas las funciones utilizadas que permiten el correcto funcionamiento del dispositivo, y se expresarán mediante diagramas de flujo para una mejor comprensión de las mismas.

Al haber utilizado la tarjeta Intel Edison con el kit Arduino, como ya se ha comentado con anterioridad, se ha programado mediante el IDE de Arduino. Por ello, existen dos ramas de ejecución principales en el programa, el *setup* (la cual se ejecuta la primera y una única vez) y el *loop* (que se ejecuta constantemente una vez ha finalizado *setup*).

Cabe a destacar, que antes de la ejecución del *setup* al principio del programa, se incluyen todas las librerías necesarias que permiten que se realicen las funciones pertinentes respecto al control del display LCD y de la tarjeta SD. Además se definen todos y cada uno de los pines utilizados y se declaran las variables y constantes globales, las cuales se usarán a lo largo de todo el programa siendo empleadas por varias funciones.

Igualmente, además de este capítulo, al final del documento se incluye el código de programación (anexo 1) con una explicación detallada añadida en comentarios.

4.1 Setup

Esta función es llamada cuando comienza la ejecución del programa o *sketch*. Se trata de una función de configuración ya que en ella se declaran los modos en los que actuarán los pines (entrada o salida) y se inicializan las diferentes comunicaciones. La función *setup* (figura 47) no se ejecuta más veces a no ser que el dispositivo sea reiniciado, apagado o el programa se vuelva a cargar.

En el programa creado, inicialmente se declaran las entradas: el sensor de irradiancia, de temperatura y los tres pulsadores. También se declaran las salidas: los LEDs indicadores y la retroiluminación del display LCD. Seguidamente se inicia la comunicación serie, la comunicación con el LCD y se crean diversos caracteres para éste, definidos anteriormente.

Se realiza un bucle, el cual se ejecuta constantemente hasta que la comunicación de la tarjeta MicroSD con el dispositivo se haya producido

correctamente y se haya inicializado. En caso de no inicializarse correctamente, se manda un mensaje de error por el monitor serie y el LCD y se esperan dos segundos para volver a realizar el bucle. En caso contrario, manda un mensaje indicando el correcto inicio de la comunicación, añadiendo un retraso para que se observe de forma clara en el LCD y continúa con la ejecución de la función *establecer_fecha* (figura 48).

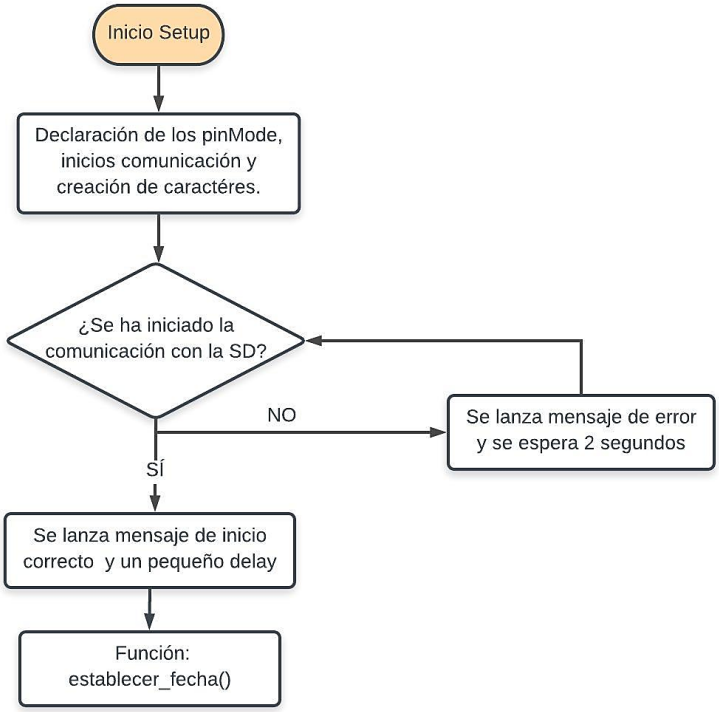

Figura 47. Diagrama Setup.

4.1.1 Función establecer_fecha

Con esta función finaliza la función principal *setup*, a pesar de que dentro de ésta se pueda hacer alguna llamada a otra. En ella, por medio de los pulsadores instalados en el dispositivo se indicará la fecha en la que se comienzan a tomar los datos.

Primeramente, se pide mediante el monitor serie y el LCD que se establezca la fecha según los parámetros {día, mes, año, hora, minuto, segundo}, mostrando el parámetro en el que se encuentra el sistema y el valor de este gracias a la llamada de la función *info_parametro,* que se explica más adelante.

Seguidamente, se entra en un bucle en el que de forma continuada se comprueba si se está pulsando uno de los tres pulsadores.

En caso de haberse realizado una pulsación corta (menor de 2 segundos) en el primer pulsador, el sistema pasa al siguiente parámetro, de

modo que si inicialmente se encontraba en el día, pasa al parámetro mes, y así hasta el segundo. Una vez se realiza otra pulsación corta en el segundo, vuelve al día. Ahora bien, en caso de que se haya realizado una pulsación larga (mayor de 2 segundos), se envía el mensaje «Reseteando» y todos los valores de los parámetros se resetean volviendo a su valor inicial. Después de cada pulsación, ya sea corta o larga se incluye un *delay* de 200 ms para diferenciar correctamente las pulsaciones.

En caso de ser pulsado el segundo botón (es indiferente con pulsación larga o corta, ya que solo tiene una funcionalidad), el valor del parámetro del dispositivo en ese momento es aumentado en una unidad hasta el máximo que se haya establecido. Una vez se sobrepasa el máximo vuelve al valor mínimo (el inicial). Los máximos serán {31, 12, 37, 23, 59, 59}, siendo 37 el año máximo debido al problema existente ahora mismo en los dispositivos de 32 bits con el denominado «Efecto 2038» del que se hablará en el Capítulo 7. En caso de ser un mes de 28, 29 o 30 días, el propio estándar de Linux tiene las funciones necesarias para indicar correctamente los pasos al día y mes correspondiente. Aquí, también se incluye un *delay* de 200 ms, y el hecho de dejar pulsado el botón hace que cada 200 ms aumente en uno el valor, haciendo que este aumente más rápido que al ser pulsado de uno en uno.

Por último, al accionar el tercer pulsador menos de 2 segundos, al contrario que la función del pulsador 2, el valor del parámetro disminuye en una unidad. En caso de sobrepasar el mínimo establecido {01, 01, 00, 00, 00, 00} en cualquier parámetro, se establece el máximo. Ahora bien, si se pulsa durante más de 2 segundos se imprime el mensaje «Guardando valores» dejando un tiempo para su buena lectura. Se cambia la variable MODO, lo que hace salir al sistema del bucle de comprobación de botones y de nuevo se envía otro mensaje en el que se ofrece un menú de selección para observar la irradiancia, temperatura o apagar o encender el LCD mediante la pulsación de los botones. El *delay* en este caso funciona de la misma forma que en el pulsador 1.

Una vez acabado el bucle de comprobación, se asigna a una variable local (*tiempo_intel*) la hora interna de la tarjeta Intel Edison, y a otra variable local (tiempo_introducido) el valor que devuelve la función *estruct_a_time_t,* la cual es una función que devuelve la fecha introducida por el usuario en formato condensado (en segundos). Esta función se explica más adelante.

Tomando estas dos variables se realizará la diferencia entre el tiempo introducido y el de la Intel con la que más tarde, se calculará la fecha correcta introducida por el usuario. Se ha realizado de este modo, ya que no se ha encontrado una manera más sencilla de obtenerla. También esto se explicará detalladamente más adelante.

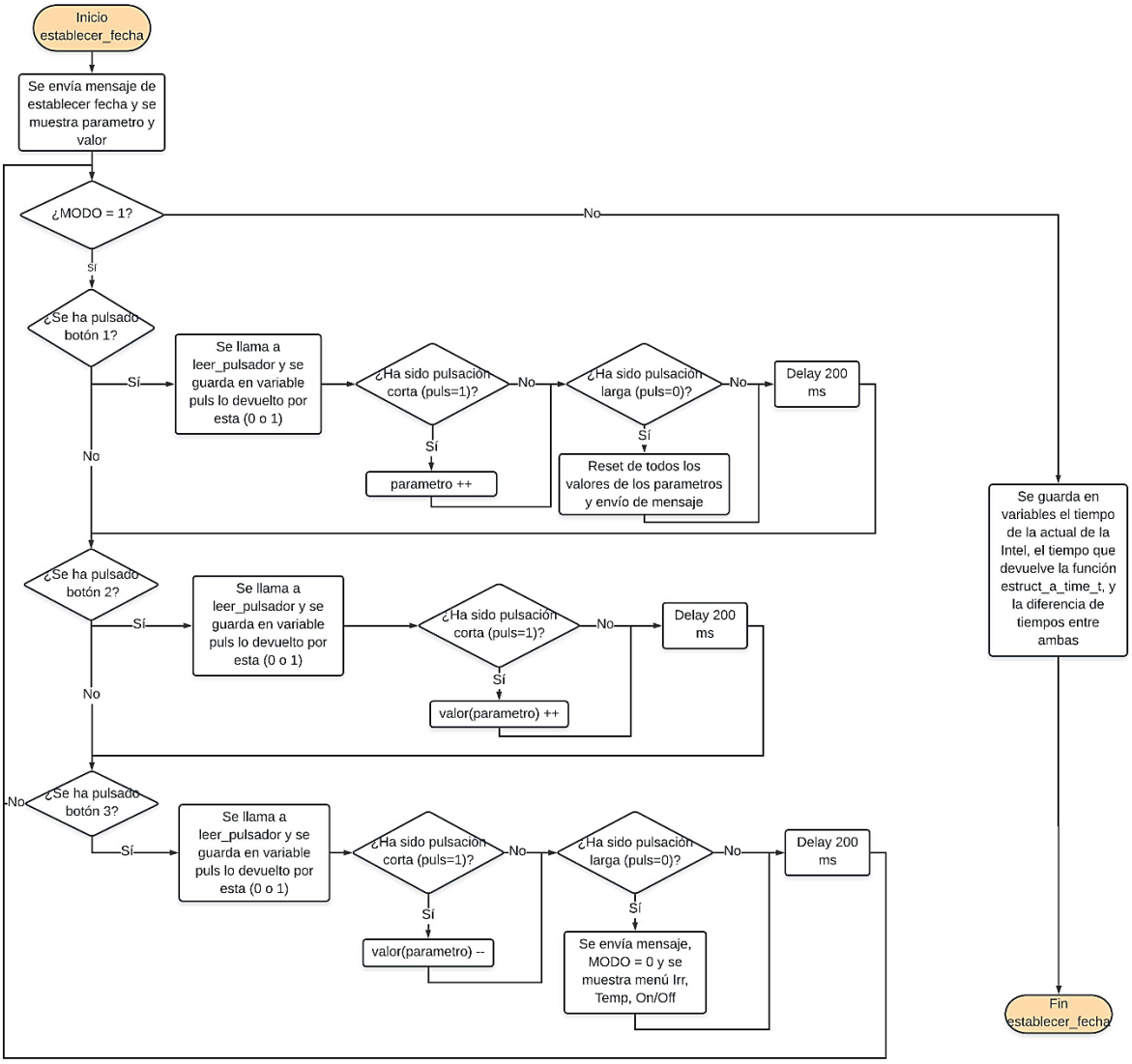

Figura 48. Diagrama de la función "establecer_fecha"

4.1.2 Función leer_pulsador

Cada vez que se detecta una pulsación dentro del bucle de comprobación, se llama a la función *leer_pulsador* (figura 49) pasándole el parámetro *pinpuls* x (valor de x según el pulsador que haya sido pulsado). Esta función es la que comprueba mediante un *if*, y la función *millis* si se ha realizado una pulsación corta o larga. De esta forma se toma el tiempo en el que se produjo la primera pulsación, y se comprueba con un *while* si el botón sigue pulsado.

En caso de haber sido mantenido el botón durante más de 2 segundos, la función devuelve un 0. En caso contrario, ésta devolverá un 1 indicando que la pulsación ha sido corta.

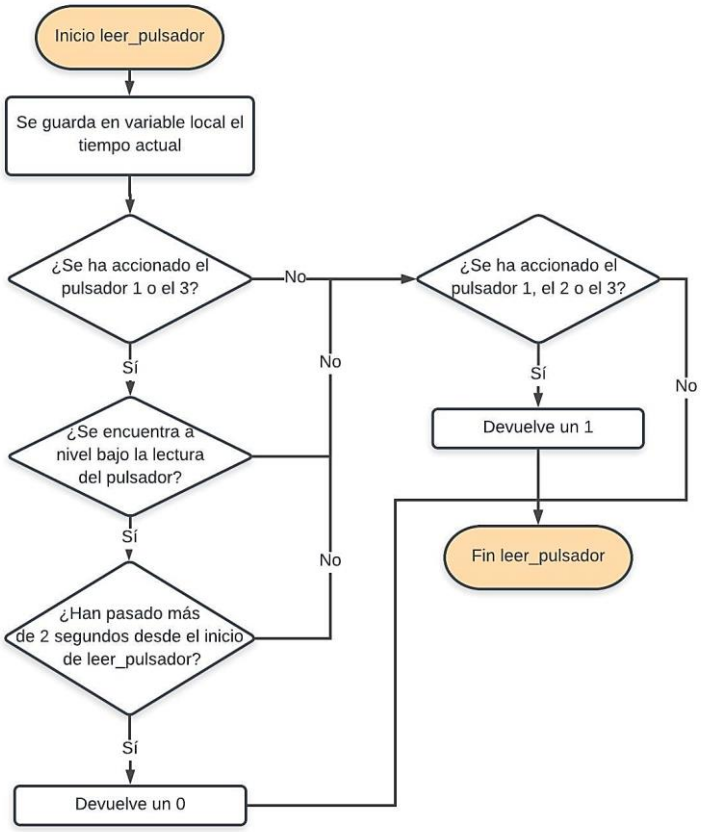

Figura 49. Diagrama de la función "leer_pulsador"

4.1.3 Función info_parametro

Cada vez que se realiza una pulsación corta, con la finalidad de que el usuario sepa el valor del parámetro y el parámetro en el cual se encuentra para fijar la fecha, se llama a esta función *info_pametro* (figura 50) a la que se le pasa el parámetro y el valor del parámetro.

Esta imprime en el monitor serie y en el LCD el parámetro en «letra» (no en número como se utiliza en la variable) y el valor de cada parámetro en ese momento.

Se realiza por medio de un switch-case.

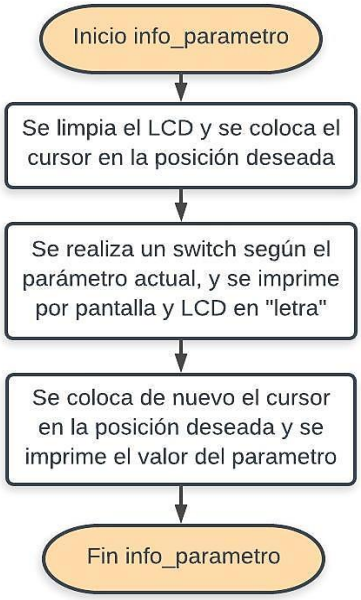

Figura 50. Diagrama de la función "info_parametro".

4.1.4 Función estruct_a_time_t

Esta función es llamada al finalizar *establecer_fecha*, y es la encargada de guardar en una estructura predeterminada del estándar de Linux (*struct tm*) denominada *timeinfo*, la fecha y hora con los valores introducidos por el usuario (en formato separado). Una vez realizado, los convierte mediante la función *mktime* en formato condensado (en segundos) en un entero *time_t* y los guarda en la variable local *tiempo*. Seguidamente este valor se devuelve a la función para su posterior uso (calcular la constante de diferencia de tiempo).

4.2 Loop

La función *Loop* como ya se ha dicho, es la que se ejecuta cíclicamente. Ésta es en la que se realiza la toma de datos y la que actualiza todas las variables según los intervalos de tiempo requeridos.

Primeramente, mediante un *if* y la función *millis* se comprueba si se ha producido el intervalo de tiempo requerido entre cada toma de datos (en este caso 40 ms = 25 tomas por segundo). Después se comprueba si alguno de los pulsadores ha sido pulsado, asignando el modo de funcionamiento. Si se ha pulsado el botón 1, se asigna el MODO=2, que se corresponde con el modo que muestra por el LCD la irradiancia en ese momento. En caso de que se haya apretado el pulsador 2, se asigna el MODO=3, que muestra la temperatura del momento por el LCD y, por último, si se ha pulsado el tercer botón se asignará el MODO=4 y se apagará o encenderá el LCD según el estado en el que se encuentre.

A continuación, mediante otro *if*, se comprueba si también se ha producido el paso de 1 segundo desde la última actualización de los demás datos. En caso de ser así, se procede a las llamadas de las funciones *actualizar_fecha, conversion_sensores, LED_RGB, mostrar_irradiancia, mostrar_temperatura, onOff* y *almacenamiento*.

A las tres funciones nombradas, *mostrar_irradiancia, mostrar_temperatura* y *onOff,* se accederá por medio de un *switch-case* según el MODO de funcionamiento que haya elegido el usuario.

Con la finalidad de que no se almacenen en la SD la irradiancia y temperatura con un valor de 0 (valores iniciales de las variables) en la primera toma, puesto que ya ha pasado un segundo desde que se inició el programa, se hace uso de un contador el cual cada vez que se toma un dato, se incrementa en uno. Así, por medio de un if, hasta que *cont* no supere 1, no puede almacenar, haciendo que el valor 0 no se almacena en el primer *loop.*

Una vez realizadas esas funciones, puesto que ya se han realizado las 25 tomas de datos (ha pasado el segundo), se ponen a cero las variables *irr, temp* y *cont*, para empezar de nuevo a acumular los valores.

Por último, se realizan dos sumatorios de los valores obtenidos por los sensores, uno de la irradiancia y otro de la temperatura. Estos se guardarán en las variables *irr* y *temp* que más tarde se almacenarán en la memoria SD junto con la fecha. También se incrementará un contador que cuenta el número de tomas realizadas.

Para intentar simplificar la explicación se ha creado el diagrama de la figura 51.

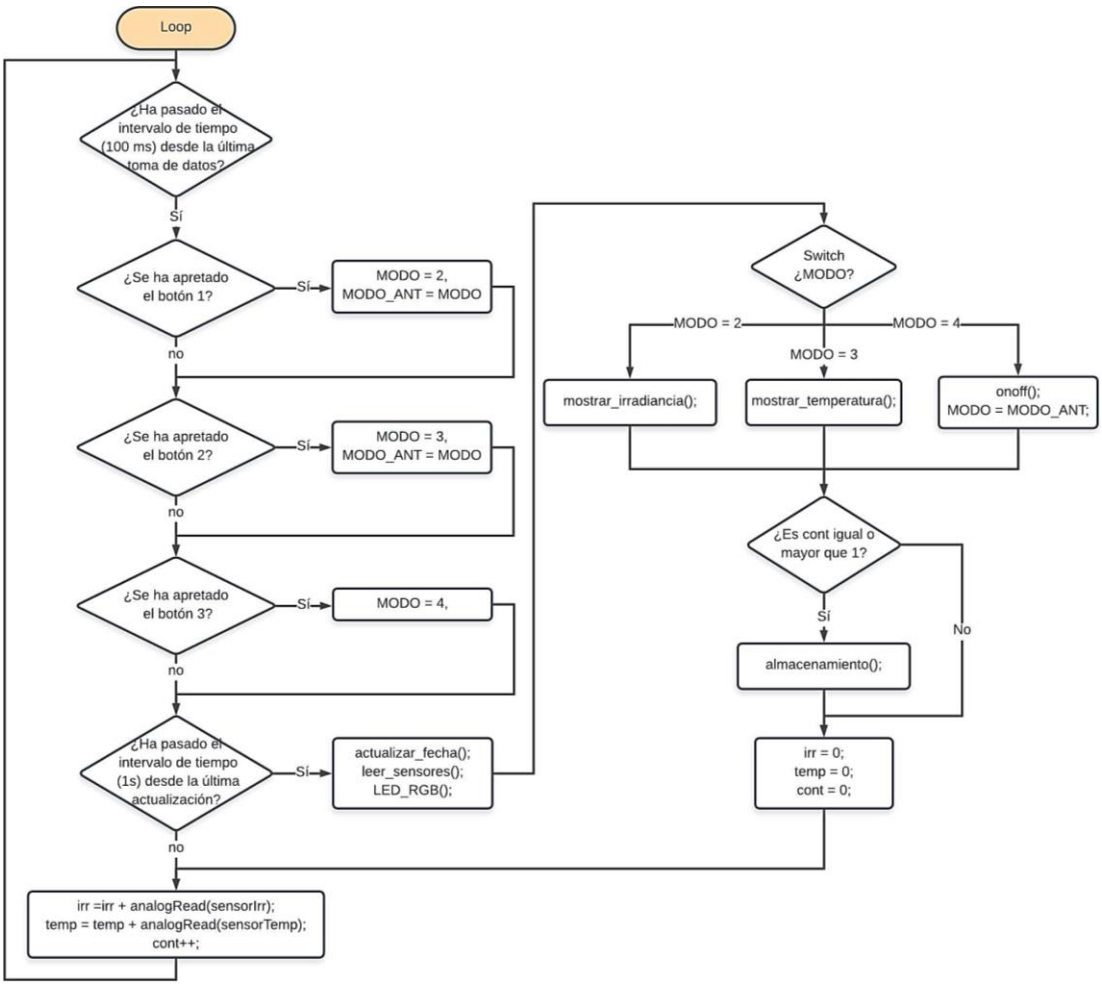

Figura 51. Diagrama de la función Loop.

4.2.1 Función actualizar_fecha

Esta función es la que se encarga de actualizar la fecha del sistema cada segundo. En ella se crea una variable local (*fecha_actual*) con la fecha interna de la Intel Edison, y a esta variable se le suma el valor de la constante

de tiempo calculada en la función *establecer fecha*. El resultado se guarda en la variable global *timestamp* que posee el valor actualizado de la fecha y hora en formato condensado. Después, mediante la función *localtime* se guarda el *timestamp* en valores que representan el tiempo correspondiente, expresado para la zona horaria local y se guarda en una estructura local *tm* (*tiempo_struct*).

Una vez realizado esto, por medio del comando *strtime*, se guardan los valores de la estructura en dos cadenas, una llamada *hora* en formato Hora:Minuto:Segundo, y otra llamada *fecha* en formato Días-Mes-Año.

4.2.2 Función conversion_sensores

En esta función, primeramente se realiza el guardado de las variables de los dos sensores leídos en el *loop* (irr y temp), promediándolos con el número de tomas por segundo. Estos valores, sabiendo que los convertidores analógico-digitales prestan una resolución de 10 bits, son convertidos a mili voltios y guardados en una variable local.

Una vez convertidos en voltaje, conociendo las especificaciones de los sensores y su relación entre la entrada y la salida una vez realizada la amplificación, calculamos los valores de irradiancia y temperatura correspondientes. Estos serán los que se muestren por el monitor serie y por el display LCD.

4.2.3 Función almacenamiento

Se trata de una función que se encarga guardar en un fichero *.txt* de la SD introducida en el adaptador de la placa, la fecha en formato condensado, el valor acumulado de la irradiancia y el valor acumulado de la temperatura, para su posterior procesamiento en un programa externo.

Primeramente, se crea el nombre del archivo en el que se guardan los datos. Mediante el comando *strcpy*, se copia en un vector de caracteres denominado *fichero*, la cadena *fecha* que contiene la fecha introducida por el usuario. Seguidamente mediante el comando *strcat* se concatena fichero con el texto *.txt* para dotar de una extensión al documento. También se podría haber puesto *.csv* o similares. De esta forma queda creado el nombre del archivo.

Ahora mediante la función *SD.open* que proporciona la librería, se abre el archivo creado, o en caso de no existir en la raíz de la SD, se crea. En caso de que se haya podido abrir este archivo, por medio de *.print* se escribe en el archivo la fecha y hora en formato condensado (*timestamp*), la irradiancia acumulada (*irr*) y la temperatura acumulada (*temp*), todas ellas separadas por

una coma, lo que permite a la hora de utilizar el archivo por otro programa, manejar correctamente los datos y tenerlos separados. Una vez escritos los valores, se cierra el archivo para que los datos se guarden correctamente.

En caso de no haberse abierto correctamente el archivo, se envía un mensaje de error por el monitor serie y por el LCD.

4.2.4 Función LED_RGB

Por medio de esta función se consigue iluminar un LED RGB con distintos colores según la irradiancia incidente en ese momento. Se ha simplificado de lo que será la función final, y en este caso sólo se mostrarán 3 colores correspondientes a tres intervalos de irradiancia, lo cual no quiere decir que estos sean los intervalos utilizados en el programa del proyecto.

Si la irradiancia es menor de 1000W/m^2, el led se iluminará en rojo. Si se encuentra en un rango entre 1000 y 2000 W/m^2, se iluminará en verde. Y si la irradiancia es mayor de 2000 W/m^2 se iluminará en azul. Todo ello se realiza con los comandos *if* y *else if* como se puede observar en el diagrama de la figura 52.

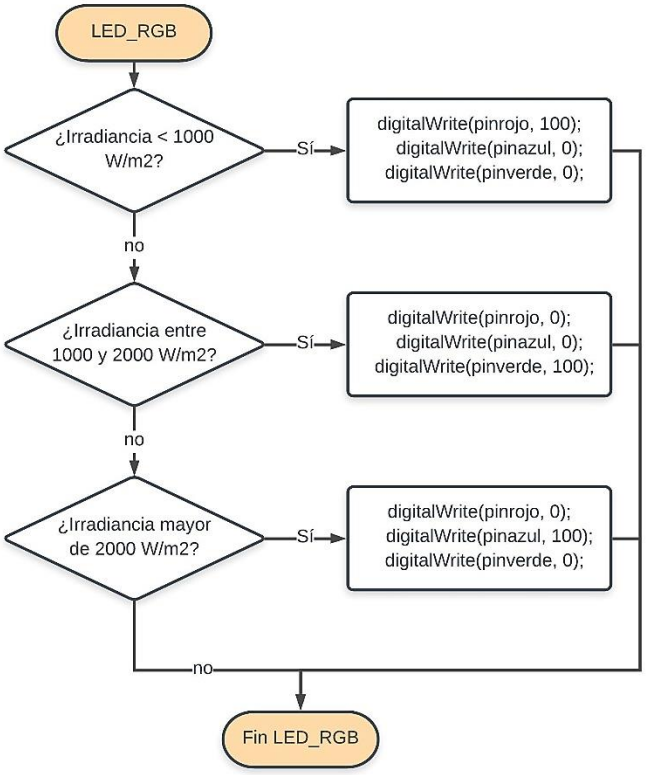

Figura 52. Diagrama de la función LED_RGB.

Si se desea introducir más rangos lo único que hay que hacer es dotar de más colores al RGB por medio de las salidas PWM que aporta la placa.

4.2.5 Función mostrar_irradiancia

Esta función mostrará la hora e irradiancia (en W/m^2) del momento por el monitor serie. Además, mostrará también la irradiancia por el display LCD. Utiliza los valores obtenidos en la función *leer_sensor*, y no los valores acumulados.

Se actualizará cada segundo.

4.2.6 Función mostrar_temperatura

Al igual que la anterior, esta función realiza el mismo trabajo con la excepción de que en vez de irradiancia lo que muestra es la temperatura en grados centígrados.

4.2.7 Función onOff

Esta función se encarga de encender o apagar la retroiluminación de la pantalla LCD, y dejar o no de mostrar lo que está mostrando.

Cuando se llama a la función, en caso de que la pantalla esté encendida (variable luz = 1), se cambia el valor de luz y se actualiza la señal de salida del pin, que lo apaga. Si no, es decir, si está apagada (luz = 0) se muestra de nuevo por el display la irradiancia o temperatura según la variable MODO_ANT (indica el modo anterior en el que se encontraba el sistema), y se enciende la retroiluminación. La figura 53 muestra el funcionamiento.

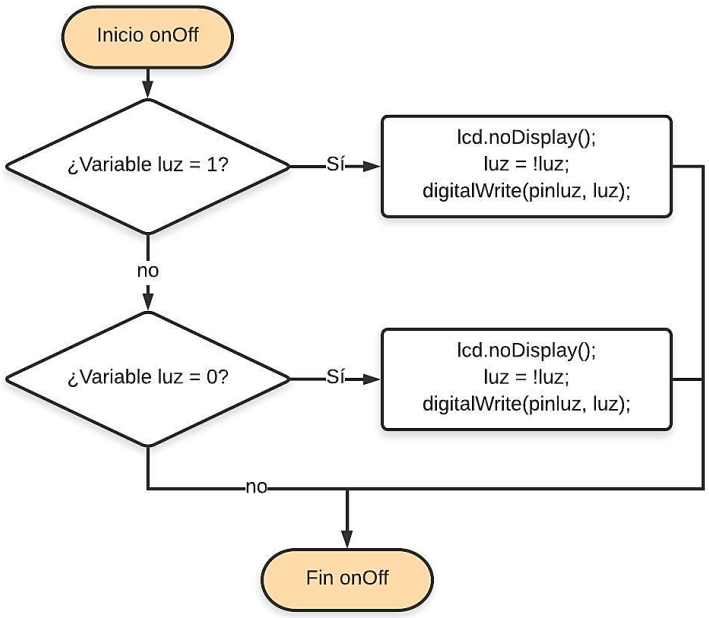

Figura 53. Diagrama de la función onOff.

Capítulo 5: Construcción del dispositivo y pruebas.

En este capítulo se va a detallar como se ha diseñado y construido el dispositivo medidor. Primeramente se hablará del prototipado en una «breadboard», y después del montaje final. También se mostrarán los resultados de las pruebas finales realizadas, incluyendo las gráficas correspondientes de irradiancia y temperatura.

5.1 Prototipado en Breadboard

Con el fin de realizar un montaje de prueba del dispositivo, se pensó en utilizar una placa de pruebas con los diferentes elementos que componen el dispositivo. Puesto que la célula solar no se podía conectar directamente, se decidió sustituir ésta por un fotoresistor, más conocido como LDR (ligth-dependant resistor) y una resistencia, únicamente para observar cómo se comportaba el sistema. Respecto a la toma de temperatura, se ha utilizado un sensor LM35 como el instalado en la célula solar.

Primeramente, se comenzó con el montaje de los botones (figura 54), conectándolos a los pines analógicos A3, A4 y A5 (que pueden trabajar como digitales), y comprobando con lo programado que funcionaba correctamente. Se observó que había veces que se producían rebotes provocando que al pulsar una única vez el botón, el sistema tomara como si se hubiese pulsado más de una vez. Para solucionarlo se añadió a cada pulsador, un condensador en paralelo de 100nF.

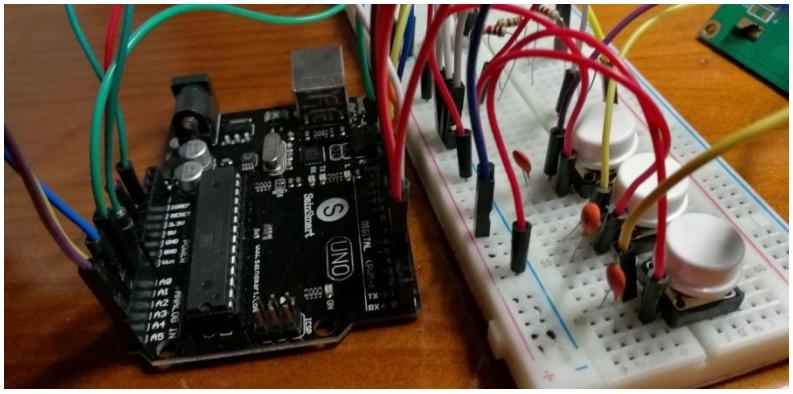

Figura 54. Montaje de los botones.

Una vez comprobado que los botones realizaban su función, se pasó a incorporar la LDR en forma de divisor de tensión con otra resistencia de 10 kΩ con el fin de simular lo que podría ser la célula solar. El hecho de no utilizar la célula en un primer caso fue por comodidad a la hora de realizar las pruebas y debido a que la conexión en esta placa de pruebas era complicada.

Se comprobó que los valores obtenidos tenían sentido, por lo que se pasó a montar el sensor de temperatura. Este sensor se conectó a un pin

analógico y a los correspondientes 5V y GND, verificando que los datos que proporcionaba eran correctos. En la figura 55 se muestra el montaje de ambos sensores sobre la placa.

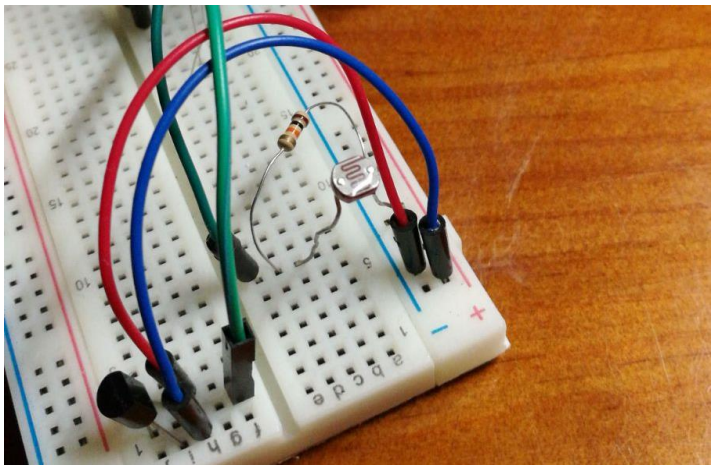

Figura 55. Montaje de los sensores.

Con los sensores ya montados, se procedió a instalar el display LCD (figura 56). Éste utiliza de una gran cantidad de pines, y su conexión se ha explicado en el capítulo 3, aunque hay que destacar la inclusión de 2 resistencias, una al pin del led que controla la retroiluminación del LCD para limitar la corriente por este, y otra al pin Vo que controla el contraste del LCD. Sus valores son de 220 Ω y 2200 Ω respectivamente.

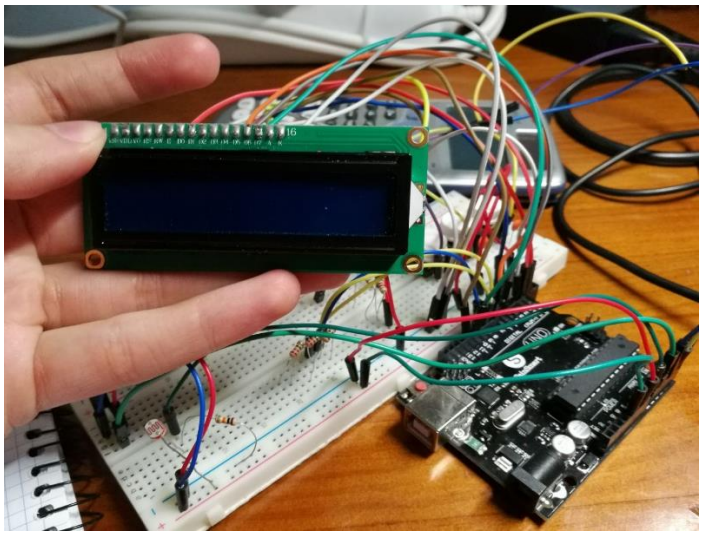

Figura 56. Montaje del LCD.

Puesto que el resultado fue positivo, se procedió al montaje del led RGB (figura 57). Éste se ha conectado a los pines de la placa que proporcionan una señal PWM (modulación por anchura de pulsos), la cual permite regular la intensidad de luz que emite el led RGB. También, se ha conexionado cada pin correspondiente a los colores rojo, verde y azul a una resistencia de 220 Ω que

regula la corriente que circula por él sin dañarlo. El cátodo común se conecta a tierra.

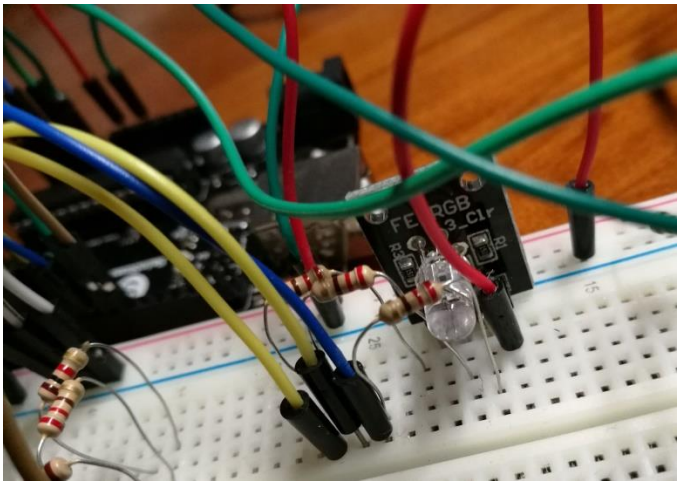

Figura 57. Montaje del led RGB.

Comprobado que el led emite la tonalidad correspondiente según lo establecido en el programa, se da por finalizado el prototipado en la placa de pruebas, ya que se ha observado que todos los elementos junto con sus conexiones funcionan como se esperaba. El conjunto total quedaría como se muestra en la figura 58.

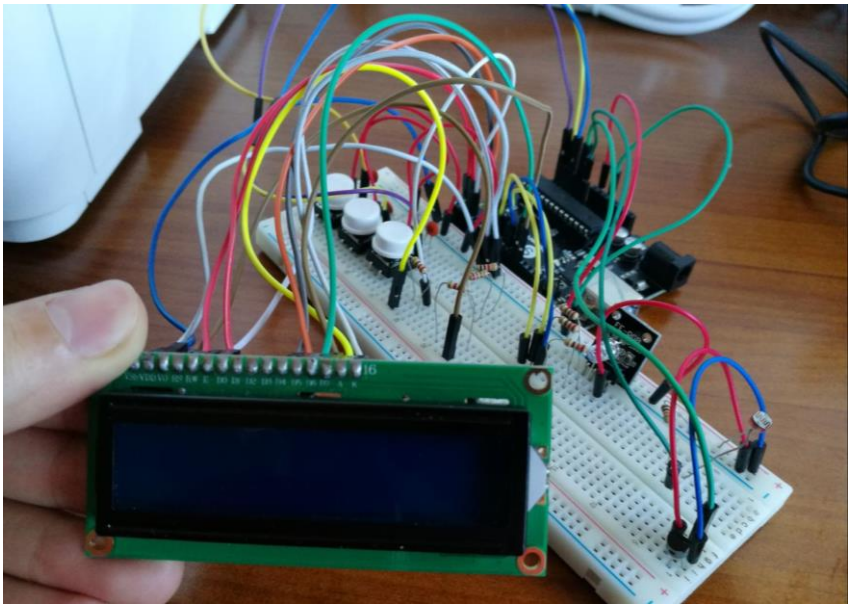

Figura 58. Prototipado final.

A partir de aquí, se realizan todos los cambios y pruebas añadidos que se necesiten en el programa, con el fin de realizar las simulaciones que mejor se ajusten al dispositivo final.

5.2 Montaje final

En primer lugar, para empezar a construir el dispositivo se ha perforado la parte trasera de la caja. Se han realizado tres agujeros de los tamaños correspondientes al interruptor de encendido y apagado, y a los dos conectores respectivos que se van a utilizar; un conector DIN de 4 contactos para las señales de los sensores de temperatura e irradiancia, y el conector de alimentación DC. Una vez realizados los agujeros, se han insertado y fijado los elementos mencionados como puede verse en la figura 59.

Figura 59. Parte trasera del dispositivo.

En segundo lugar, una vez instalados los elementos mencionados, se ha fijado la placa Intel Edison en el centro de la base de la caja. Se ha realizado por medio de tornillos, tuercas y una pistola termofusible, fijando unos pequeños tacos en los agujeros de la placa para evitar el contacto con la base. Después, se han conectado los cables mediante un soldador, de tal forma que el interruptor permita el paso de la corriente desde el conector exterior DC al conector de la placa computadora como se puede observar en la figura 60.

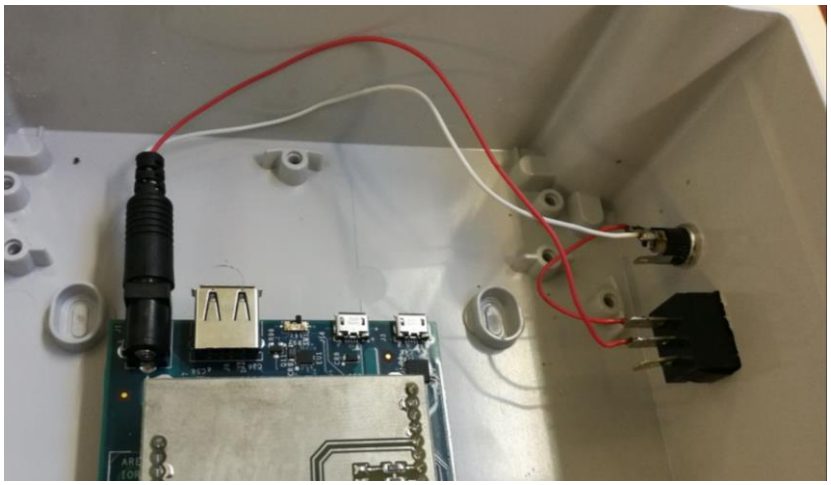

Figura 60. Conexión de la alimentación.

Con el fin de simplificar el montaje, se decidió realizar una placa de circuito impreso (PCB) con el diseño de los pines del Arduino, de tal forma que ésta encajara en la placa Intel Edison. De esta manera se han podido soldar los cables mejorando su fiabilidad, evitando introducirlos como en la placa de pruebas, lo que provocaba que pudieran no hacer bien contacto debido a la holgura y que con algún movimiento o vibración, estos se desconectaran. Además, en la placa se han introducido las resistencias necesarias para los leds tanto del LCD como del led RGB y los condensadores que evitan el rebote en los pulsadores. Todos estos elementos son de montaje SMD (Surface Mounted Device).

El diseño de la placa (figura 61) se ha realizado con el programa de software libre Eagle.

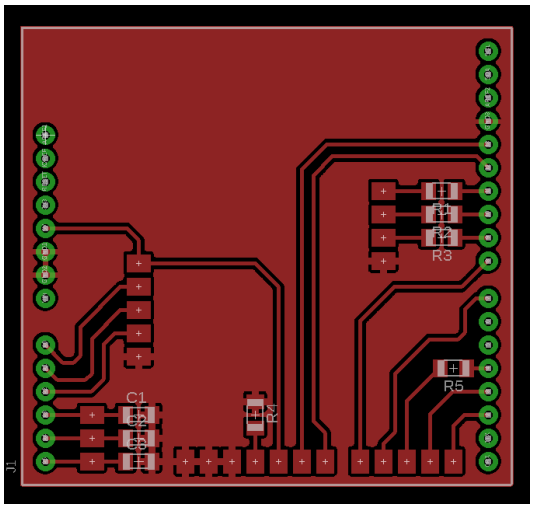

Figura 61. Diseño placa PCB.

Una vez diseñado, se ha taladrado la placa, insolado, revelado y atacado con ácido. Después se le han soldado unos pines para introducirlos en los de la placa Intel, dando el resultado de las figura 62 y 63.

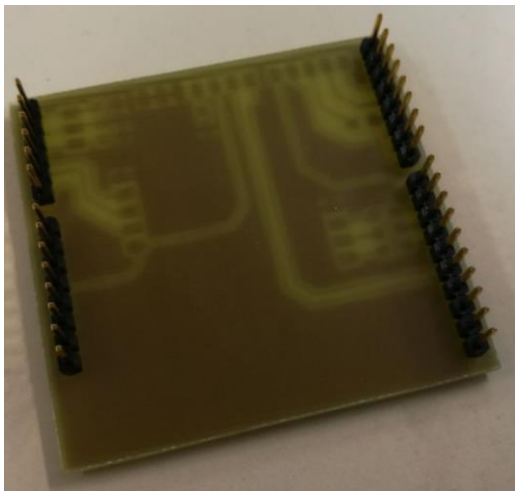

Figura 62. Cara inferior de la PCB.

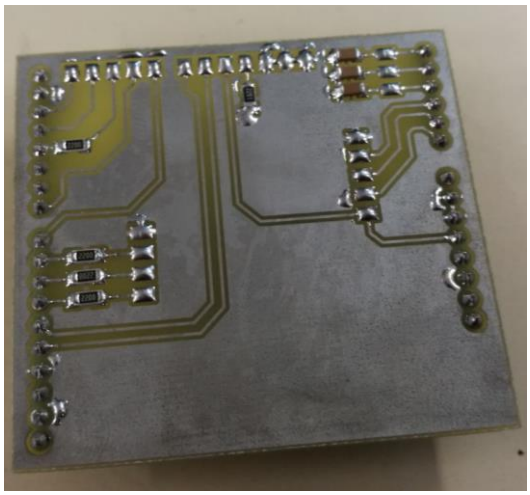

Figura 63. Cara superior de la PCB.

Ahora se realiza en la tapa de la caja los agujeros correspondientes para la fijación de la pantalla LCD, del led RGB y de los tres botones con los que se maneja el dispositivo. En las figuras 64 y 65 se puede ver el proceso y como quedan instalados estos elementos.

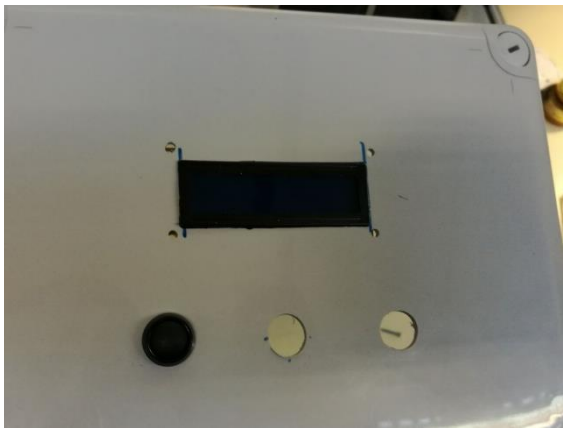

Figura 64. Parte superior de la tapa.

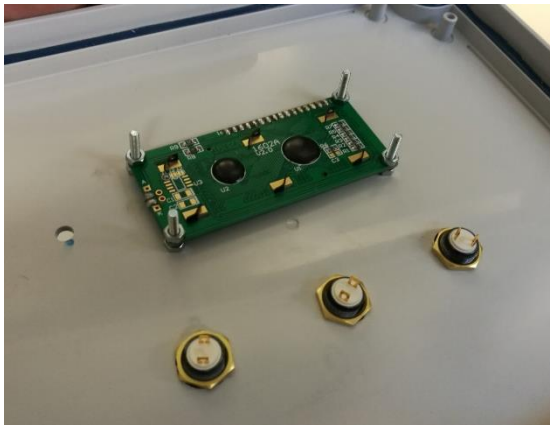

Figura 65. Parte inferior de la tapa.

La fijación del LCD se realiza por medio de tornillos y tuercas, la de los botones por unas tuercas y unas gomas aislantes que traen incluidas en su compra, y el led RGB mediante la termoselladora.

Una vez se tienen todos estos elementos instalados, sólo hay que soldar los cables a sus correspondientes pines (figura 66), dándose por finalizado el montaje de esta parte del dispositivo.

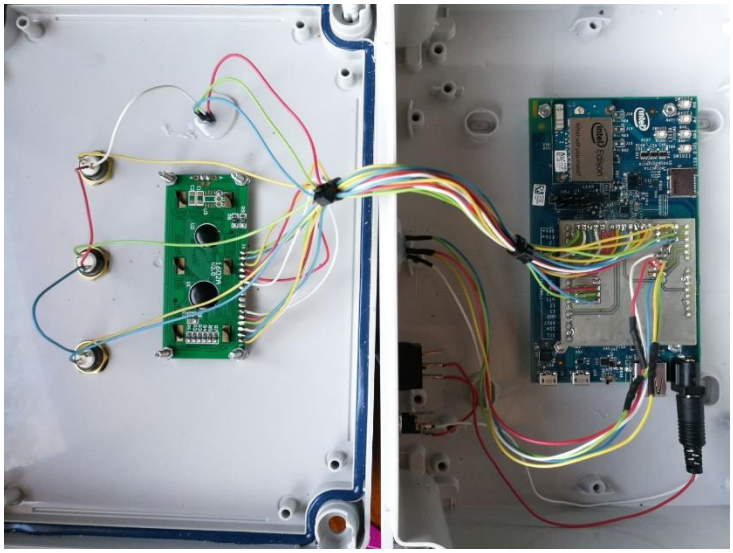

Figura 66. Conexionado del cableado.

Por otro lado, se ha de conectar la célula solar junto con el sensor de temperatura a la caja por medio del DIN que se instaló en la parte trasera. Primeramente, mediante la pistola termofusible y una pasta térmica

conductora se ha pegado el sensor LM35 a la célula por la parte de atrás como se ve en la figura 67.

Figura 67. Montaje del sensor LM35.

Observando las características del sensor de temperatura, su relación entre la entrada y la salida (10mV/°C), y teniendo en cuenta el rango de voltaje que permite la placa Intel Edison, se consideró la amplificación de la señal de salida de éste. Puesto que la placa Intel acepta tensiones entre 0 V y 5 V, que se traducen a dígitos entre 0 y 1023, el valor de cambio de cada dígito es de 4,83 mV. Sabiendo de la relación entrada/salida, significaría que un cambio de 5 mV detecta un cambio de 0,5°C y puesto que tiene un rango de 200°C, solo se estaría aprovechando 1 V del rango del Arduino. Por ello, se decide amplificar la señal de salida con un amplificador operacional con una ganancia de 3. Así, la relación ahora es 30 mV/°C, es decir, que cada 5 mV que varía la señal, el sensor detecta 0,17°C y no 0,5°C que detectaba anteriormente, mejorando su sensibilidad a pesar del offset del amplificador.

A su vez, como se comentará en el capítulo 7, se instalarán 2 diodos y una resistencia de 18 kΩ con la finalidad de que el sensor pueda medir en todo su rango de temperaturas.

Por otra parte, al igual que pasa con el sensor de temperatura, la célula solar proporciona una señal pequeña (1000 W/m^2 por cada 100 mV = 50 W/m^2 por cada5 mV). Esto hace que el dispositivo preste una resolución baja, por lo que se decide amplificar la señal con una ganancia de 4, de tal forma que ahora los 5mV corresponden a 12,5 W/m^2 aumentando su fiabilidad.

Por todo ello, se ha creado otra placa de circuito impreso (figura 68) con los correspondientes circuitos amplificadores para cada sensor, así como el circuito que permite leer temperaturas negativas en el LM35.

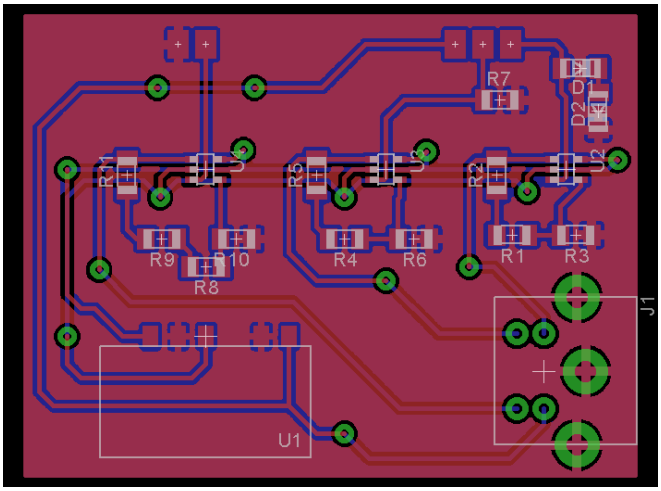

Figura 68. PCB para los sensores.

Las pistas en azul se corresponden a la cara superior de la placa y las rojas a la cara inferior. U1 es un convertidor DC-DC que proporciona la señal negativa de -5 V para el correcto funcionamiento de los amplificadores, ya que la placa Intel, al igual que las Arduino, no suministran tensiones negativas. J1 es el conector DIN de 4 pines que se conecta a la caja donde se encuentra la Intel, los U2, U3 y U4 son los amplificadores operacionales, y los demás elementos son los diodos y resistencias que permiten la amplificación y la lectura de temperatura negativa. La placa una vez finalizada se ve como la de la figura 69, siendo los cables de la izquierda los de la célula solar, y el gris que incluye 3 cables, los del sensor de temperatura.

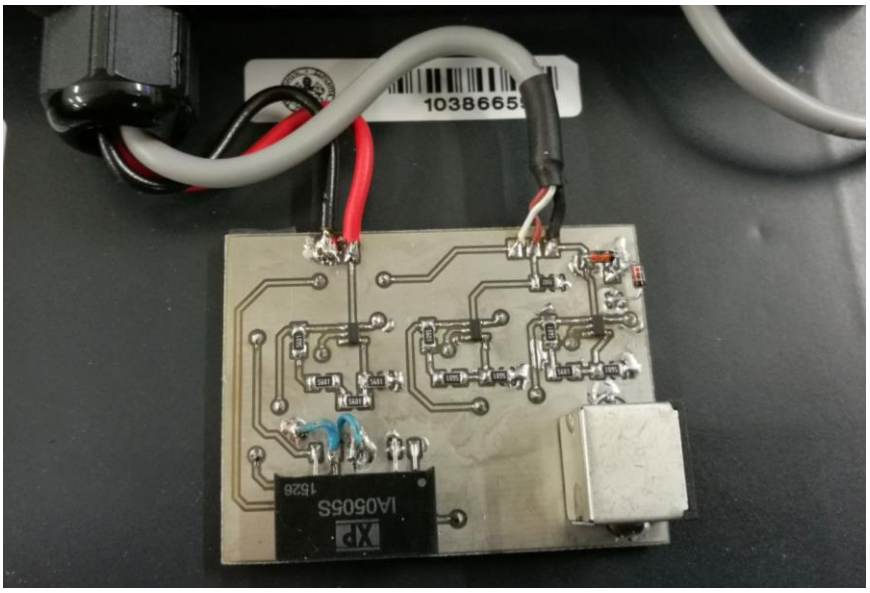

Figura 69. PCB de la célula solar y sensor de temperatura.

De esta forma, uniendo únicamente con un cable miniDIN de 4 vías la célula solar con la caja, el dispositivo quedaría montado completamente. Este se puede observar en la figura 70.

Figura 70. Montaje final.

5.3 Pruebas

Con el dispositivo final ya montado, se ha realizado a lo largo de 4 días la captación de las medidas de irradiancia solar y de temperatura de la célula. Esto, se ha realizado en la terraza de una casa particular en el barrio de Parquesol. Las gráficas 71 y 72 que se muestran a continuación, se corresponden con las medias móviles, con intervalos de 60 datos, de los días 10 y 11 de febrero de 2018 respecto a la irradiancia, y las gráficas 73 y 74 se corresponden a las de esos mismos días, pero respecto a la temperatura. Todas ellas entre las 14:30 y las 23:59

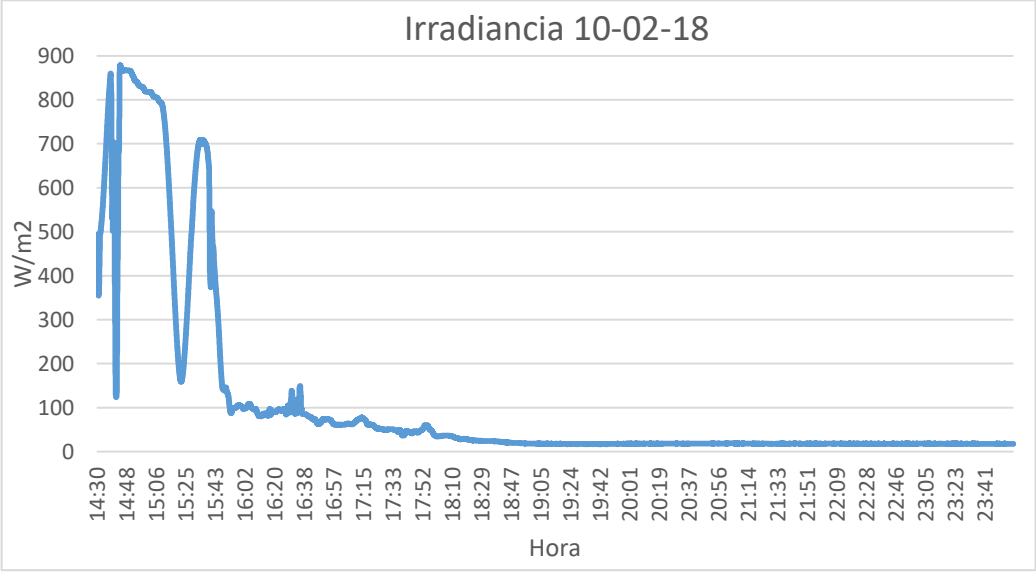

Figura 71. Gráfica de la irradiancia a 10-02-18.

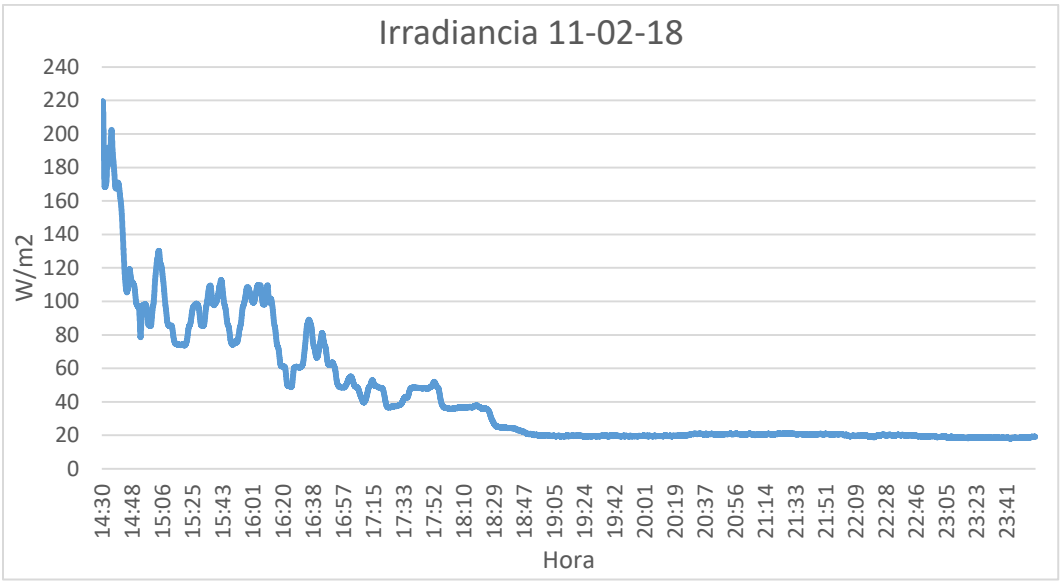

Figura 72. Gráfica de la irradiancia a 11-02-18.

Se observa que existe un pequeño offset (aproximadamente de 20 W/m^2) cuando la célula solar no recibe energía. Esto es debido al offset y error de los amplificadores operacionales y del convertidor DC-DC que se han incorporado. Se puede observar también la diferencia de magnitudes entre un día y otro, llegando a casi 900 W/m^2 el día 10, y solo a 220 W/m^2 el 11. Esto es debido a que en ese intervalo de toma de datos, el día 10 fue soleado con alguna nube, y el día 11 fue nublado.

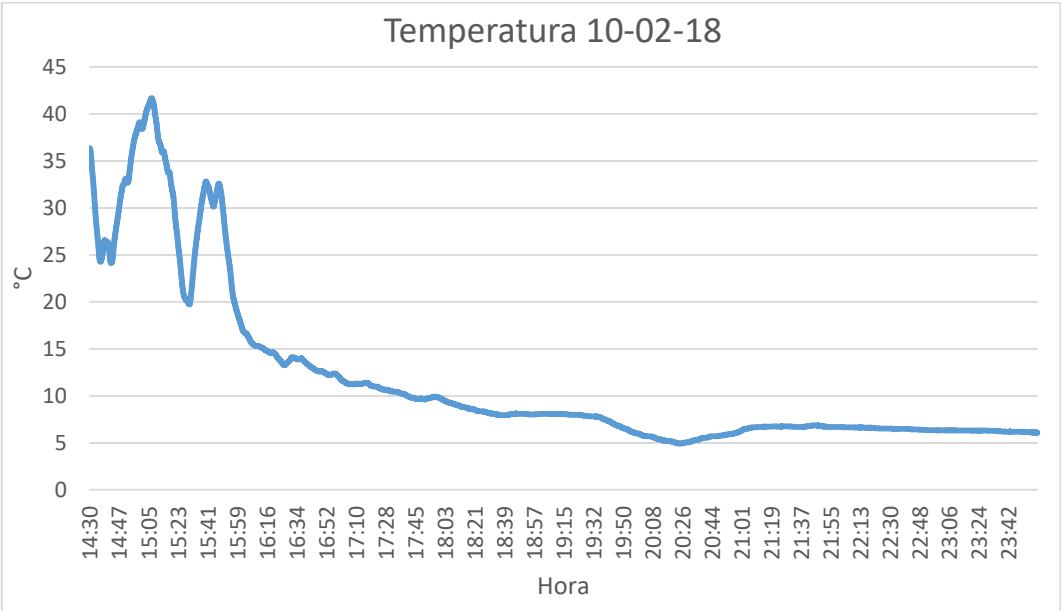

Figura 73. Gráfica de la temperatura a 10-02-18.

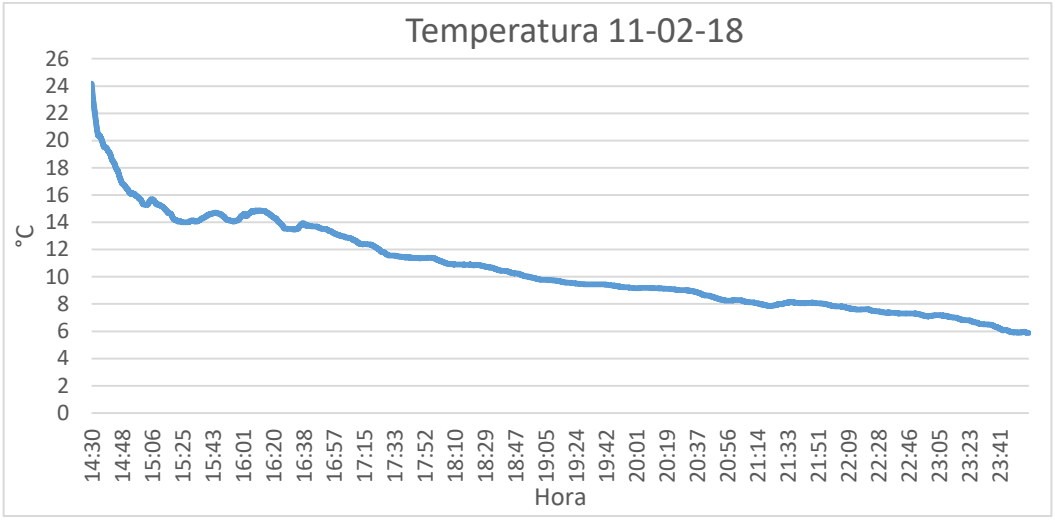

Figura 74. Gráfica de la temperatura a 11-02-18.

Respecto a las gráficas de temperatura de la célula solar, decir que se corresponden con las gráficas de la irradiancia, ya que cuanta mayor temperatura se ha registrado, ha sido en los momentos en que mayor irradiancia había, que coinciden cuando la célula se encontraba expuesta directamente al sol. A medida que se acerca la noche, la temperatura baja debido a la ausencia del sol y al enfriamiento atmosférico. Cuando se produce una bajada repentina de irradiancia por una nube o alguna sombre en la célula, la temperatura varía de forma más gradual, incluso no llegando a variar si la sombra sobre la célula dura poco tiempo.

Con la finalidad de comparar otra horquilla de horas, se han realizado otras mediciones los días 12 y 13 de febrero de 2018 entre las 00:00 y las 15:00 horas. Las gráficas 75 y 76 muestran la irradiancia registrada.

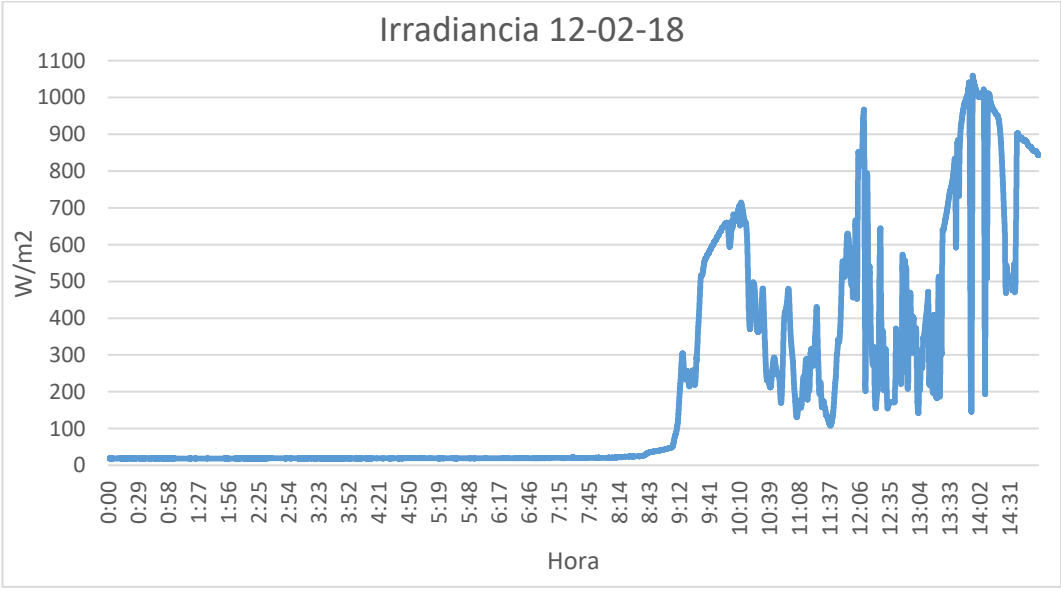

Figura 75. Gráfica de la irradiancia a 12-02-18.

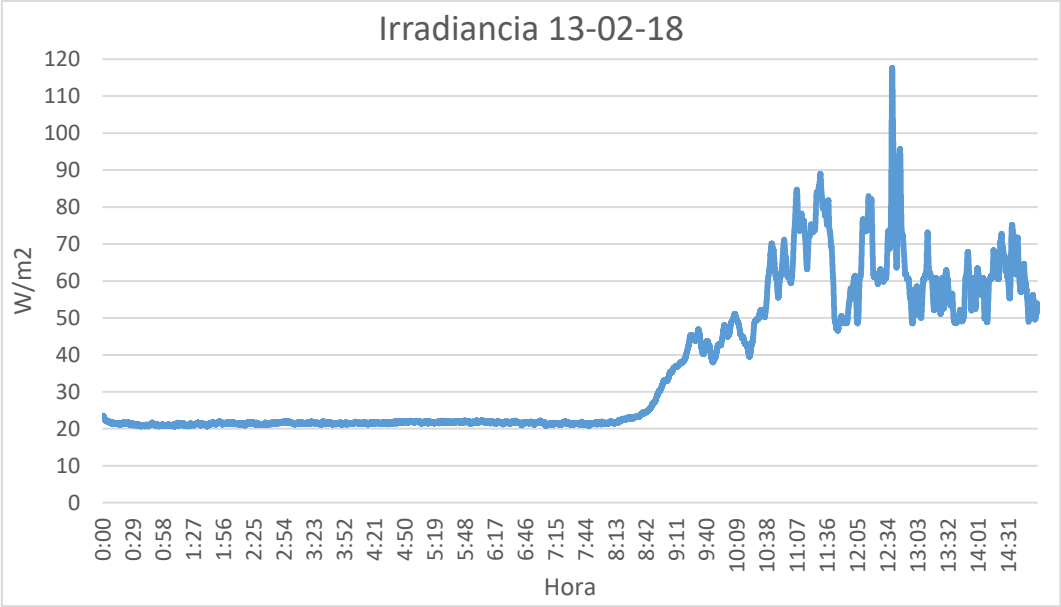

Figura 76. Gráfica de la irradiancia a 13-02-18.

De las gráficas, observando los altibajos en la irradiancia, se puede extraer que el día 12 fue un día soleado y con nubes, llegando a los 1050 W/m^2. Respecto al día 13, viendo que la máxima irradiancia no llega a los 120 W/m^2, se deduce que fue un día nublado.

Respecto a la temperatura, observando la gráfica de la figura 77, se corrobora lo dicho anteriormente, a medida que aumenta la irradiancia, aumenta la temperatura en la célula. Al disminuir la irradiancia, la temperatura también lo hace. Por la noche la temperatura se iguala aproximadamente a la temperatura ambiente.

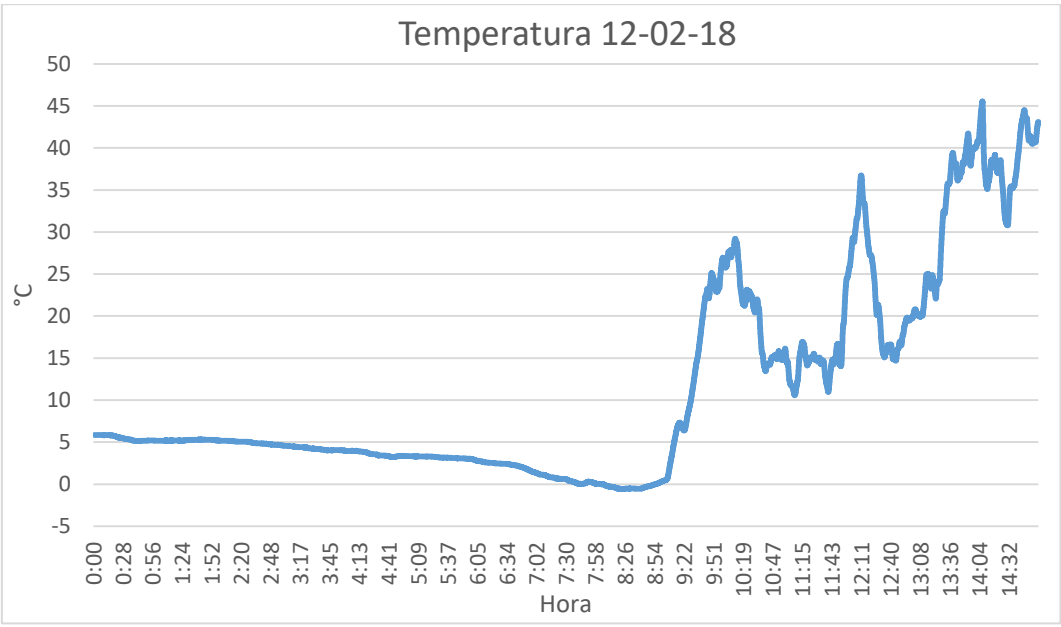

Figura 77. Gráfica de la temperatura a 12-02-18.

A su vez, en la figura 78 la temperatura se comporta como era de esperar, aunque cabe destacar que ésta se hace negativa a lo largo de la noche y que el dispositivo la registra correctamente.

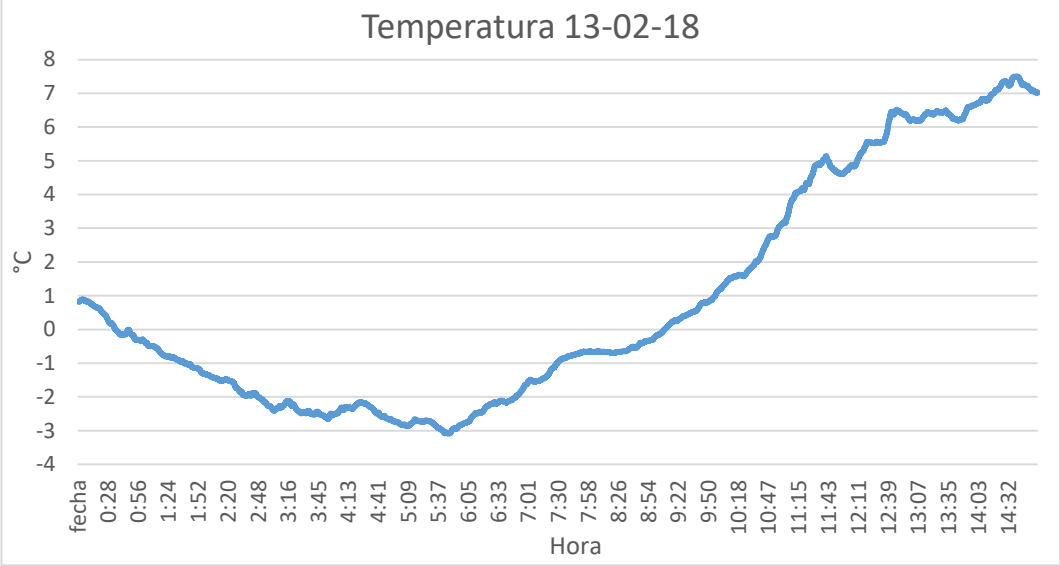

Figura 78. Gráfica de la temperatura a 13-02-18.

Una vez observado el comportamiento de las gráficas y comparado con la situación climatológica de esos días, se dan por concluidas las pruebas del dispositivo considerando un resultado positivo.

Capítulo 6: Costes del proyecto.

En todo proyecto, se necesita conocer su viabilidad, tanto de los aspectos técnicos, los teóricos como los económicos. Hasta ahora, se han estudiado los dos primeros, pero en ningún caso se puede desarrollar un proyecto sin antes haber realizado una aproximación económica de lo que puede suponer realizarlo.

Por lo que, en este capítulo se realiza una estimación del coste del proyecto, analizando los recursos utilizados, los costes directos e indirectos asociados y el total de todos ellos.

6.1 Recursos empleados

Al realizar este proyecto, se han utilizado diversos recursos. Se pueden diferenciar entre bienes o recursos tangibles (aquellos recursos físicos, que pueden ser tocados) e intangibles (aquellos que no se pueden tocar). Todos estos recursos, no son de un uso único, es decir, que no se van a utilizar solo en este proyecto, sino que a lo largo de su vida útil se van a emplear en más situaciones. Por tanto, habrá que tener presente que su coste no será íntegro, se tendrá en cuenta el tiempo de utilización para calcular la amortización correspondiente.

Los recursos tangibles utilizados, tanto equipos informáticos, como equipos electrónicos, como herramientas destinadas a la fabricación y sus respectivos precios se encuentran en la tabla siguiente:

Tabla 7. Recursos tangibles.

RECURSOS TANGIBLES	PRECIO
Ordenador portátil Lenovo Z50-70	599 €
Taladro de columna	90 €
Cizalla para corte de PCB	40 €
Taladro Proxxon Micromot 50/E	33 €
Soldador	15 €
Pistola termofusible	10 €
Osciloscopio	300 €
Polímetro	15 €
Fuente de alimentación (Adaptador)	10 €
Placa de pruebas (Breadboard)	5 €
Destornilladores, alicates, limas	15 €
Cúter, regla	5 €
TOTAL COSTE	1.029 €

A la hora de hablar de los bienes intangibles, hay que decir que están compuestos por todos los programas informáticos utilizados (ver tabla 8). La mayoría de estos se pueden descargar de forma gratuita.

Tabla 8. Recursos intangibles.

RECURSOS INTANGIBLES	PRECIO
Microsoft Office Profesional Plus 2013	499 €
Eagle	Gratuito
Arduino IDE	Gratuito
Sistema operativo (SO) Windows 10	Incluido en el PC
TOTAL COSTE	499 €

6.2 Costes directos

Los costes directos son aquellos costes que pueden asociarse de forma clara a la realización de este proyecto.

Dentro de estos costes pueden diferenciarse cuatro tipos diferentes que se desarrollaran a continuación:

- Costes de personal.
- Coste de amortización de los recursos.
- Coste de materiales directos empleados.
- Costes de consumibles.

6.2.1 Costes de personal

En todo proyecto, se necesita de alguien que lo lleve a cabo, ya sea una única persona o un grupo de trabajo. El emplear a una persona conlleva un coste, y este coste es el coste de personal.

Por ello, en la realización de este TFG se ha supuesto un ingeniero industrial que llevará a cabo todas las tareas, incluyendo el estudio del problema, la recopilación de información, la programación de los códigos necesarios a implementar, la construcción y el montaje del dispositivo, la realización de pruebas necesarias y ejecución de la memoria.

A la hora de calcular el coste se habrá de tener en cuenta el sueldo bruto anual medio de un ingeniero y la cotización a la seguridad social de la que se hará cargo la empresa (un 35% del sueldo bruto). Por lo que el coste anual del ingeniero queda resumido en la tabla 9:

Tabla 9. Coste anual de un ingeniero industrial.

COSTE ANUAL DE UN INGENIERO	
Sueldo bruto medio anual	35.000 €
Cotización a la seguridad social (SS)	10.500 €
TOTAL COSTE	45.500 €

Ahora, una vez conocido el coste anual del ingeniero, se han de calcular las horas atribuibles a la realización del trabajo en ese mismo año.

Por lo tanto, se ha de conocer la cantidad de horas de una jornada de trabajo, el número de días festivos y el periodo vacacional. Para ello, siguiendo el «Estatuto de los trabajadores» se obtiene que en España, el número máximo de horas semanales que se puede trabajar son 40 horas. A su vez, establece el derecho de 30 días de vacaciones, lo que equivale a 22 días de trabajo quitando fines de semana. También se han de descontar los días festivos que se corresponden a 14 días [36].

Por lo tanto conociendo todo estos se procede a calcular el número teórico de horas anuales que debería realizar un trabajador:

52 semanas anuales x 5 días x 8 horas = 2080 horas

A las que se descuentan los días de periodo vacacional y los días festivos quedando:

2080 horas – (22 días + 14 días) x 8 horas = 1792 horas

Ahora, sabiendo las horas anuales de un trabajador se puede calcular el coste por hora trabajada de este:

Coste/hora = 40.500 € / 1792 horas = 22,6 euros por hora de trabajo

A continuación, en la tabla 10 se realiza una aproximación de las horas de trabajo que pueden suponer la realización de este proyecto para un ingeniero de las características anteriormente nombradas.

Tabla 10. Distribución de las horas de trabajo.

DISTRIBUCIÓN DE LAS HORAS DE TRABAJO	
Estudio del problema	10 horas
Recopilación de información	80 horas
Programación de los códigos	70 horas
Fabricación y montaje del dispositivo	25 horas
Realización de pruebas	30 horas
Ejecución de la memoria	110 horas
TOTAL HORAS	320 horas

Por lo tanto, en el caso de este proyecto, el coste directo de personal sería:

320 horas x 22,60 euros/hora = 7.232 euros

6.2.2 Costes de amortización de los recursos

En este apartado se calculan los costes de amortización de los recursos empleados descritos en el apartado 6.1.

Siguiendo la Tabla de coeficientes de amortización de la Agencia Tributaria, se han de dividir los recursos anteriormente nombrados en 3 grupos, ya que estos grupos poseen diferentes coeficientes lineales. En la tabla 11 se indican los elementos que forman los grupos, el coeficiente de amortización y los costes de estos [37].

Tabla 11. Tabla de amortizaciones.

GRUPO	MATERIAL AMORTIZABLE	COEFICIENTE (%)	COSTE	TOTAL	TOTAL AMORTIZADO
Equipos informáticos	Ordenador portátil Lenovo Z50-70	25%	599 €	599 €	150 €
Programa informático	Microsoft Office Profesional Plus 2013	33%	499 €	499 €	165 €
Herramientas	Cizalla para corte de PCB	25%	40 €	208 €	52 €
	Taladro Proxxon Micromot E/40		33 €		
	Taladro de columna		90 €		
	Soldador		15 €		
	Destornilladores, alicates y limas		15 €		
	Pistola termofusible		10 €		
	Cúter, regla		5 €		
Equipo electrónico	Osciloscopio	20%	300 €	330 €	66 €
	Placa de pruebas (Breadboard)		5 €		
	Polímetro		15 €		
	Fuente de alimentación (Adaptador)		10 €		
				TOTAL	432 €

Los programas informáticos restantes no se han añadido ya que son gratuitos o vienen incluidos en el dispositivo a la hora de la compra.

Una vez calculado el coste total de amortización correspondiente a un año de trabajo (1792 horas), se procede a calcular lo proporcional al tiempo utilizado del proyecto:

432 € x 320 horas / 1792 horas = 77,14 €

6.2.3 Coste de materiales directos empleados

En esta sección se van a asignar los costes a los materiales que serán utilizados directamente en el dispositivo, y solo van a ser utilizados para ese fin. El valor de estos se puede observar en la tabla 12.

Tabla 12. Coste de materiales directos empleados.

MATERIAL	UDS	COSTE UNITARIO	COSTE TOTAL
Intel Edison Kit Arduino	1	115,00 €	115,00 €
Célula fotovoltaica calibrada compensada	1	128,00 €	128,00 €
Sensor de temperatura LM35	1	2,00 €	2,00 €
Pilas y portapilas	6	2,00 €	12,00 €
Caja contenedora IP66	1	12,10 €	12,10 €
Display LCD 2x16	1	10,00 €	10,00 €
LED RGB	1	3,00 €	3,00 €
Condensador 100nF	3	0,50 €	1,50 €
Condensador 1uF	1	0,08 €	0,00 €
Resistencia 220Ω	3	0,03 €	0,09 €
Resistencia 2.2kΩ	1	0,03 €	0,03 €
Resistencia 5.6kΩ	10	0,00 €	0,02 €
Resistencia 18kΩ	1	0,03 €	0,03 €
Convertidor DC-DC	1	4,33 €	4,33 €
Amplificador operacional OPA171	3	1,66 €	4,98 €
Diodos 1N914	2	0,75 €	1,50 €
Placa de prototipado 100x160 mm	1	8,79 €	8,79 €
Conecto DC Caja	1	0,88 €	0,88 €
Conector DC Placa	1	1,24 €	1,24 €
Cable miniDIN	1	2,24 €	2,24 €
Conector DC	1	0,82 €	0,82 €
Pulsador	3	3,46 €	10,38 €
Interruptor	1	1,35 €	1,35 €
Cables y pines	1	5,00 €	5,00 €
Tornillos, tuercas, arandelas	16	0,20 €	3,20 €
	TOTAL COSTE		328,48 €

El valor total de los costes de materiales directos empleados es de 328,48 €.

6.2.4 Coste de consumibles

En todo proyecto existen unos materiales los cuales se van utilizando y consumiendo, y que son igualmente necesarios. Son elementos tan simples como bolígrafos, lapiceros, gomas, folios, cuadernos, pegamento, cinta adhesiva, cartuchos de tinta, estaño para soldar... A estos se les denomina consumibles.

El coste de estos consumibles se ha tasado en 50 €.

6.2.5 Coste directos totales

En resumen, los costes totales directos se desglosan en la tabla siguiente:

Tabla 13. Costes directos totales.

COSTES DIRECTOS TOTALES	
Coste de personal	7.232 €
Coste de amortización de los recursos	77,14 €
Coste de materiales directos empleados	328,48 €
Costes de consumibles	50 €
COSTES DIRECTOS TOTALES	7.575 €

6.3 Costes indirectos

Los costes indirectos, son aquellos costes que no se pueden imputar directamente a un objeto, material o programa, es decir, que no pueden considerarse costes directos. Estos son gastos necesarios que permiten la realización del proyecto, como pueden ser el consumo de energía eléctrica (la alimentación de los dispositivos electrónicos, la iluminación), el consumo de gas (calefacción), consumo telefónico (internet), y otros servicios varios. Un desglose de estos gastos se puede observar en la tabla 14.

Tabla 14. Costes indirectos totales.

COSTES INDIRECTOS TOTALES	
Consumo eléctrico	100 €
Consumo de gas	50 €
Consumo telefónico (Internet)	60 €
Otros servicios	30 €
COSTES INDIRECTOS TOTALES	240 €

6.4 Costes totales

Por último, para finalizar con este capítulo, en la tabla 15 se muestran los costes totales de la realización del proyecto. En ella se incluyen todos los costes anteriormente citados.

Tabla 15. Costes totales.

COSTES TOTALES	
COSTES DIRECTOS	7.688 €
COSTES INDIRECTOS TOTALES	240 €
COSTES TOTALES	7.928 €

Capítulo 7: Problemas encontrados.

En este capítulo se va a proceder a describir todos los problemas encontrados durante la realización del proyecto y la forma en la que se han solucionado.

7.1 Actualización de la placa Intel Edison

Lo primero que se ha de hacer al obtener la placa Intel Edison es configurarla. Para ello, se han de instalar los drivers o controladores en el ordenador con el que se vaya a utilizar y actualizar su firmware, es decir, el sistema operativo de su microcontrolador, que en este caso es una versión de Linux.

Para hacerlo, se acudió a la página web de Intel en la cual te dan las herramientas necesarias para la correcta configuración. El problema resultó cuando al descargar y actualizar la última versión que proporcionan, ésta hacía que la placa no funcionara correctamente. La funcionalidad WiFi no hacía su función ya que al buscar redes del entorno no encontraba ninguna. Además, una vez se había cargado el sketch o programa en el microcontrolador, si se desconectaba del USB, este dejaba de ejecutarse, y cuando se volvía a conectar no iniciaba el programa.

Ante ello, se recurrió al foro de Intel, en el cual trabajadores del soporte técnico daban unas pautas complicadas de seguir, y aun siguiéndolas, no funcionaba tampoco. El problema base radica en que Intel ha dejado de dar soporte a la plataforma Intel Edison, y no van a lanzar más actualizaciones software [38].

Finalmente, se decidió instalar una versión firmware más antigua y solucionar el problema de raíz, consiguiendo que funcionase el WiFi y que el programa se ejecutase correctamente, a pesar de no estar conectado mediante el USB al ordenador.

7.2 Incompatibilidad de librerías SD.h y TimeLib.h

Realizando la programación del proyecto en Arduino, con el fin de almacenar los datos en una tarjeta SD, se recurrió a la librería SD.h que facilita la gestión (apertura de ficheros, escritura y lectura de datos, cierre de ficheros…).

A su vez, para el establecimiento de la hora introducida por el usuario, se utilizó una librería denominada *TimeLib.h*, que es una mejora o ampliación de la librería *time.h* de la biblioteca estándar del lenguaje C, que contiene

funciones para manipular o formatear la fecha y hora del sistema. Con esta librería *TimeLib.h*, se podían fijar y actualizar la fecha y hora mediante unos comandos simples como son *setTime* y *now*.

Sin embargo, a la hora de compilar el programa se obtenía un error en el que se indicaba que había dos librerías que entraban en conflicto entre ellas, y no permitían la subida del programa al microcontrolador. Por más que se buscó en los códigos de las librerías y se intentó encontrar el error no se consiguió.

A consecuencia de ello, se eliminó la opción de utilizar la librería *TimeLib.h* y se decidió utilizar la de la biblioteca estándar (*time.h*). Esta librería no otorga directamente la opción de fijar una fecha por el usuario, por lo que se ha tenido que utilizar la fecha interna de la placa Intel y a partir de ella conseguir establecer la introducida por el usuario. Esta placa tiene establecida como fecha inicial 31 de marzo de 2015, a la hora 13:00:00 (establecida por su firmware en ese momento). Puesto que el usuario usualmente introducirá la fecha del momento en el que se encuentra, con la fecha introducida por él, se hará una resta sobre la fecha interna de la placa obteniéndose una constante. Esta constante será la que habrá que sumar en cada ciclo de un segundo a la fecha interna de la placa para mostrar la hora y fecha introducida por el usuario. Todo ello realizado con variables y estructuras dedicadas para el tratamiento del tiempo.

Se trata de un método más complicado que requiere mayor carga computacional, pero es la única forma que se ha encontrado para solucionar el conflicto entre librerías, y para evitar utilizar un reloj de tiempo real que aumentaría el coste del dispositivo y complicaría la construcción y necesidad de pines.

7.3 Problema del año 2038 para sistemas de 32 bits

En los sistemas de 32 bits, los programas en C y C++ utilizan un tipo de datos *time_t* para representar las fechas y tiempos internamente. Este tipo de dato es un entero de 32 bits con signo, por lo que puede representar números entre -2 147 483 648 y 2 147 483 647. Transcurre desde el 1 de enero de 1970 y el valor más alto que puede representar es el 19 de enero de 2038 a las 03:14:07 UTC. En ese momento el programa alcanzará su límite y el reloj desbordará, devolviendo un valor erróneo y provocando que muchos dispositivos dejen de funcionar correctamente. Este problema también es conocido como Unix Millenium Bug [39].

No existe una solución universal aún; la mayoría de las soluciones que se han dado son para algún software en particular.

En el caso del dispositivo empleado, para resolver el problema, se intentó cambiar la definición de time_t por un entero sin signo (unsigned int) con el fin de poder doblar la capacidad, pero esto no se ha conseguido debido a un error en la conversión a la hora de compilar el programa. Por ello se ha decidido limitar hasta el 31 de diciembre del año 2037 como lo hacen gran cantidad de dispositivos que utilizan sistemas de 32 bits, entre los que destacan la mayoría de los teléfonos móviles.

Habrá que esperar a que se acerque la fecha "crítica" para poder ver las soluciones que se aportan y sus implementaciones, aunque la aparición y el uso de los sistemas de 64 bits es la mejor opción.

7.4 Eliminación de la partición de la tarjeta MicroSD

Durante el prototipado del dispositivo, cuando se realizaban diferentes pruebas para observar si la programación funcionaba correctamente, se observó que la tarjeta de memoria proporcionada para la realización del trabajo, a pesar de poseer 16 Gb de capacidad solo tenía disponible 47,3 Mb. Se pudo ver que esto era debido a que tenía una partición. Para resolverlo se intentó formatear la tarjeta, pero al hacerlo, la memoria seguía en el mismo estado.

Se buscó la información necesaria en foros y páginas web, y finalmente para solventarlo, por medio de la utilización de PuTTY (cliente de acceso remoto a máquinas mediante SSH y otros, para plataformas UNIX y Windows 32 bits) se accedió a la Intel Edison, y se ejecutaron los comandos necesarios que permitían eliminar la partición y dotar a la tarjeta de toda su capacidad [40].

7.5 Lectura de temperaturas negativas con el sensor LM35

El sensor LM35 es capaz de leer un rango de temperaturas de -55°C a 150°C. El problema es que éste no es capaz de proporcionar voltajes negativos. Con una configuración simple (conexión a alimentación, una salida para la entrada analógica y una conexión a tierra) es capaz de medir entre 2°C y 150°C como puede verse en la figura 79.

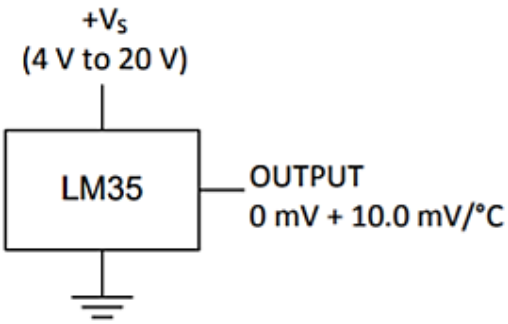

Figura 79. Montaje básico del LM35.

Para medir en todo el rango, se ha tenido que hacer un pequeño montaje externo que emplea dos diodos y una resistencia a mayores, además, se ha tenido que utilizar otro canal analógico de la placa (figura 80). De esta forma, no existe una conexión directa a GND, permitiendo que a pesar de que el voltaje no sea negativo, se puedan medir temperaturas negativas. Se realiza obteniendo la diferencia entre ente lecturas de los dos canales, siempre que ésta sea positiva, se obtienen temperaturas positivas en relación de 10mV/°C, mientras que si la diferencia es negativa, las temperaturas también lo serán [31].

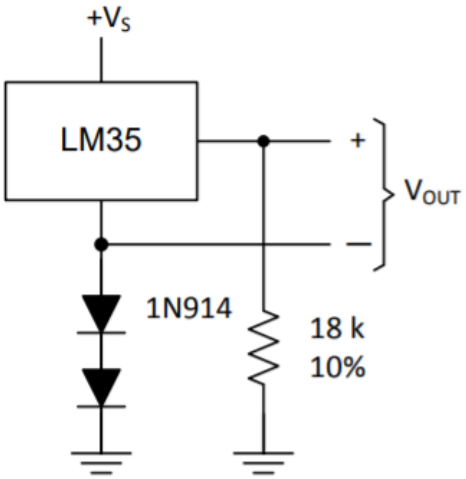

Figura 80. Montaje rango total de temperaturas LM35.

7.6 Toma de datos

En un primer momento se decidió que el dispositivo tomara 100 muestras por segundo, tanto de irradiancia como de temperatura y se programó para ello. En las pruebas se observó que en cada segundo, unas veces tomaba 96 tomas, otras 92, etc. También cuando se pulsaba alguno de los botones que cambia el estado del LCD, se alteraban aún más las tomas.

Se revisó la programación y no se encontró ningún error, por lo que se decidió reducir el número de tomas hasta que lo realizase de forma estable. Finalmente el valor del número de tomas ha sido fijado en 25, es decir cada 40 ms, considerando este número más que suficiente teniendo en cuenta el objetivo del proyecto.

7.7 Alimentación de la placa

A la hora de alimentar la placa Intel se decidió hacerlo mediante una batería de iones de litio (Li-ion) de 3,7V conectada al Jump2 de la placa, ya que esta entrada permite voltajes entre 3V y 4,3V. Una vez se fue a probar ésta, se observó que la placa era alimentada ya que se encendían los leds que indican que está siendo alimentada, pero no se encendía la pantalla LCD ni se comunicaba correctamente por el puerto serie. Investigando más profundamente en la documentación de la placa y consultando en los foros de Intel [41], se obtuvo la respuesta de que esta conexión solo alimenta el módulo de computación dejando sin alimentar los convertidores analógicos, los pines…

Por esta razón se desechó esa opción de alimentación y se escogió la opción de alimentarlo por el Jack (J1) o por el puerto microUSB (J16).

7.8 Ruido en la señal de los sensores

Realizando las pruebas una vez montado el dispositivo, se observó que los datos que leía, tanto de irradiancia como de temperatura variaban de forma notable. Respecto a la irradiancia, una vez promediada con las 25 tomas, cada segundo podía variar más de 40 W/m2, y la temperatura también promediada, un par de grados. Se observó la señal en un osciloscopio y se apreció que la señal que llegaba a la entrada de los pines analógicos del sensor de temperatura tenía bastante ruido, del orden de 100 mV (figuras 81 y 82).

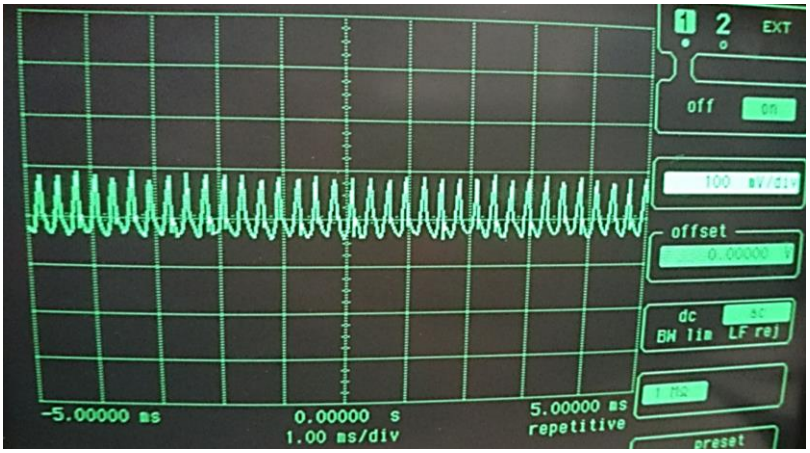

Figura 81. Señal temperatura del pin A1.

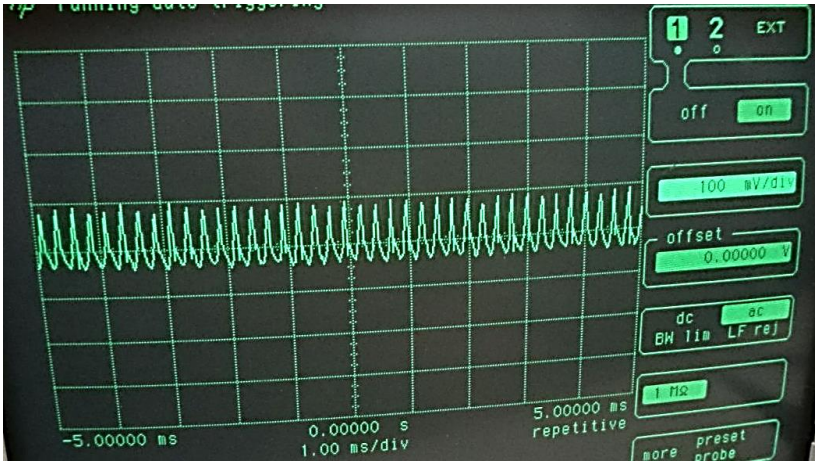

Figura 82. Señal temperatura del pin A2.

Analizando la señal de la célula solar, al igual que ocurría con el sensor de temperatura, ésta presentaba un ruido como el que se observa en la figura 83, del orden de 100 mV también.

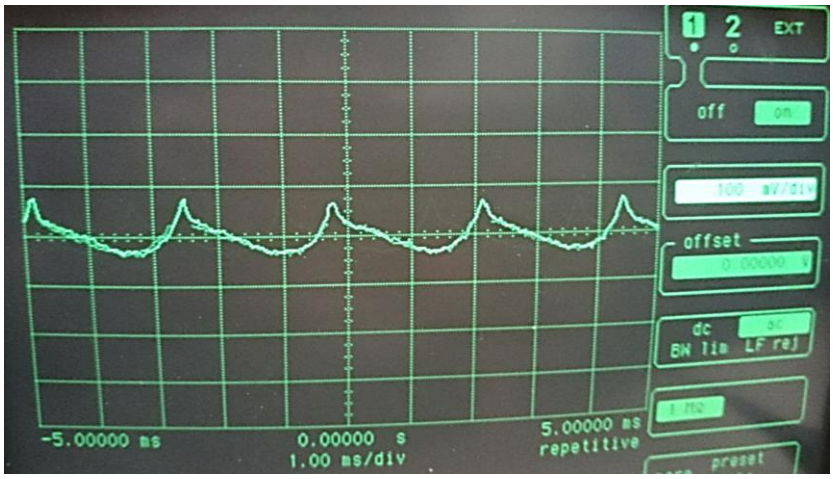

Figura 83. Señal irradiancia del pin A0.

Para solucionarlo, se decidió incluir antes del pin analógico un filtro paso bajo RC que limitara las frecuencias anteriores, a una frecuencia de corte (f_c) de 159,16 Hz, siendo el valor del condensador de 100 ηF y el de la resistencia de 10 kΩ. Esto hace que únicamente pasen las frecuencias por debajo de la f_c y que se elimine en gran medida el ruido existente. Estos montajes en cada canal no se tuvieron en cuenta a la hora de diseñar las PCB, por lo que se añadieron después, soldando las resistencias a los cables y los condensadores en la PCB de la caja.

Los resultados después de la instalación de estos filtros han sido satisfactorios, como puede observarse en las figuras 84, 85 y 86.

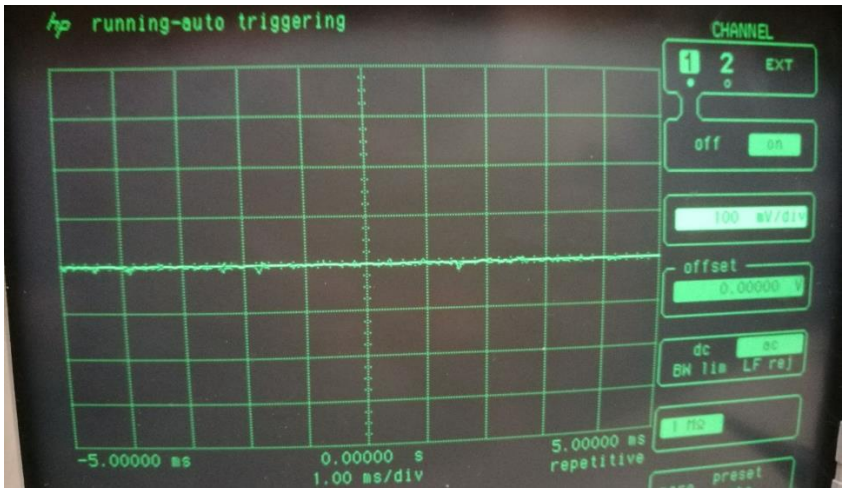

Figura 84. Señal temperatura del pin A1 filtrada.

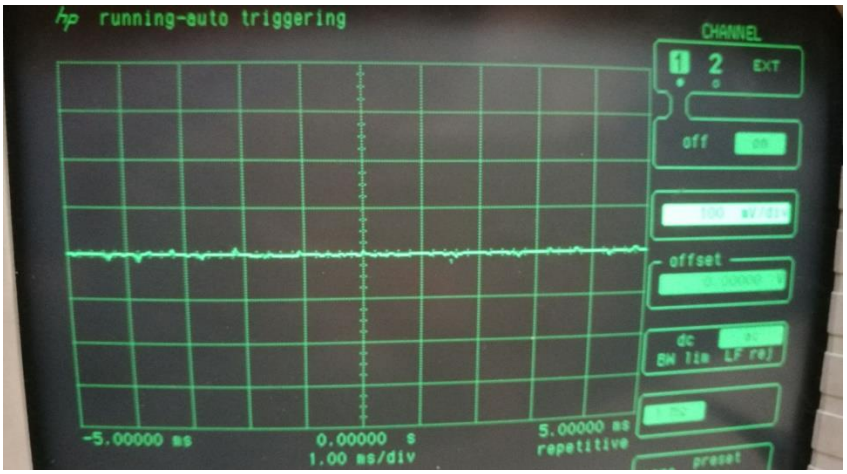

Figura 85. Señal temperatura del pin A2 filtrada.

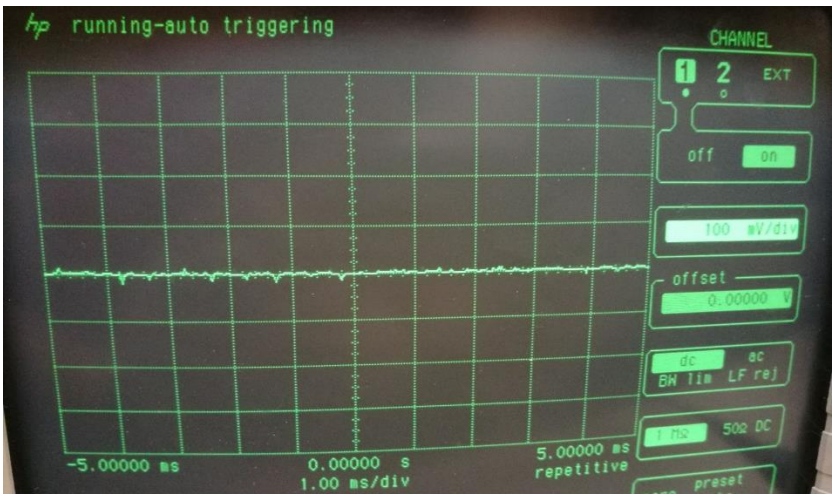

Figura 86. Señal irradiancia del pin A0 filtrada.

Capítulo 8: Conclusiones y posibles mejoras futuras.

En este capítulo se redactan las conclusiones obtenidas del trabajo una vez finalizado y las posibles mejoras que se pueden realizar en el futuro sobre el dispositivo.

8.1 Conclusiones

En este TFG se ha visto la importancia de la energía fotovoltaica y sus aplicaciones, que permiten mejorar la calidad del medio ambiente. La calidad de la energía eléctrica generada se ve afectada por las condiciones meteorológicas (irradiancia y temperatura) en las que trabajan las placas fotovoltaicas, y es por ello de la construcción de este dispositivo. Conocer las condiciones, ayuda a poder actuar de alguna forma para mejorar la producción, o controlarla, ya sea por inclinación de las placas, orientación…

La realización de este TFG ha permitido aprender en gran medida a cumplir los objetivos que se marcaron al inicio de la memoria, destacando el hecho de buscar las soluciones por uno mismo, y en caso de no encontrarlas, buscar las alternativas que permitieran la resolución de los problemas. También se ha aprendido a buscar la mejor alternativa respecto a lo requerido en el proyecto, teniendo en cuenta que a veces, éste viene condicionado por diversas circunstancias y que hay que adaptarse a ellas. Se ha ahondado en los conocimientos sobre sensores de irradiancia y temperatura, microcontroladores, funcionamiento de estos. Además, se han ampliado conocimientos respecto a las formas de alimentación de un dispositivo, se ha aprendido a gestionar y referenciar correctamente, redactar la bibliografía, realizar la programación del dispositivo mediante el IDE de Arduino, aprender a utilizar herramientas de Excel como la media móvil y realizar el montaje físico (mecánico y eléctrico), utilizando las respectivas herramientas.

Sería interesante además, realizar la calibración del sensor de temperatura con el fin de certificar que las medidas tomadas son permisibles y cercanas a la realidad. En el caso de la célula solar no es necesario, ya que ésta se encuentra calibrad, como puede observarse en las hojas de datos de los anexos.

A su vez, se ha obtenido experiencia a la hora de desarrollar proyectos en ingeniería, e integrar los conocimientos y capacidades adquiridos a lo largo del grado.

8.2 Posibles mejoras futuras

Entre las posibles mejoras caben a destacar las relacionadas con el internet de las cosas, y las mejoras físicas en el del dispositivo.

- Puesto que la placa Intel Edison posee un módulo de WiFi y Bluetooth, se ha pensado que en el caso de poseer una red WiFi o comunicación Bluetooth el lugar donde se instale el dispositivo, éste podría enviar y almacenar los datos en la nube. Todo ello se haría por medio de programación al no necesitar de más elementos físicos. De esta forma, la monitorización de los datos sería mucho más cómoda. A su vez, el hecho de estar conectado por WiFi permitiría a la placa Intel Edison conectarse a los servidores Network Time Protocol (NTP) y facilitar la obtención de la fecha en el dispositivo, ahorrando líneas de código con la consecuente mejora de la velocidad de ejecución del programa.
- Otra área de mejora sería la mejora de estanqueidad del dispositivo, sobretodo en la parte del LCD.
- La creación de un soporte para la célula, que permita posicionarla con diferentes inclinaciones según preferencias del usuario.

Bibliografía.

La bibliografía se encuentra referenciada por orden de aparición en la presente memoria. La mayoría de las referencias son de soporte online, aunque también se han utilizado recursos físicos como libros, revistas y apuntes proporcionados por los profesores de la universidad.

[1] Nelson, Jenny. Introduction: Brief history of the solar cell. *The Physics of Solar Cells.* London: Imperial Collegue Press, 2003, p. 2-3. ISBN: 1-86094-340-3

[2] Cassini, Alejandro y Levinas, Marcelo Leonardo. *La explicación de Einstein del efecto fotoeléctrico: un análisis histórico-epistemológico.* [online]. Buenos Aires: [2008], Revista latinoamericana. filos. Vol.34, n.1. [Consultado el 28 de octubre de 2017]. ISSN 1852-7353. Disponible en: <http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1852-73532008000100001&lng=es&nrm=iso>.

[3] Perlin, John. *From Space to Earth: The story of the solar electricity*. Ann Harbor, Michigan: Aatec Publications, 1999. ISBN: 0-674-01013-2

[4] U.S. Department of Energy. *The History of Solar.* [online]. [Consultado el 28 de octubre de 2017]. Disponible en: <https://www1.eere.energy.gov/solar/pdfs/solar_timeline.pdf>

[5] Grupo NAP (Nuevas Actividades Profesionales). *Energía Solar Fotovoltaica*. Madrid: Colegio Oficial de Ingenieros de Telecomunicación, 2002. ISBN: 978-84-935049-6-0

[6] Competencias del Grado en Ingeniería en Electrónica Industrial y Automática. Escuela de ingenierías Industriales de la Universidad de Valladolid. [Consultado el 3 de noviembre de 2017]. Disponible en: <https://eii.uva.es/titulaciones/grado.php?id=452&tema=comp>

[7] Efecto fotoeléctrico. Apuntes "*Tema 2 - Generadores Fotovoltaicos: la célula solar*" de la asignatura de 4º curso Electrónica de Potencia en Sistemas de Energía Alternativa de la Escuela de Ingenierías Industriales de la UVa.

[8] Camacho, René. "Arquitectura von Neumann y arquitectura Harvard" Computo integrado. 10 de abril de 2012. Bloc de René Camacho. [Consultado el 10 de diciembre de 2017]. Disponible en: <http://rcmcomputointegrado.blogspot.com.es/2012/04/arquitectura-von-neumann.html>

[9] Camacho, René. "Arquitectura RISC y CISC" Computo integrado. 17 de marzo de 2012. Bloc de René Camacho. [Consultado el 10 de diciembre de 2017]. Disponible en: <http://rcmcomputointegrado.blogspot.com.es/2012/03/arquitectura-risc-y-cisc.html>

[10] Valdés Pérez, Fernando y Pallás Areny, Ramón. Introducción a los microcontroladores. *Microcontroladores: Fundamentos y aplicaciones con PIC*. Barcelona: Marcombo Ediciones Técnicas, 2007, p. 11-17. ISBN: 84-267-1414-5

[11] Stallings, William. Memoria interna: Tipos de ROM. *Organización y arquitectura de computadores.* Madrid: Pearson Educación S.A., 2005, p. 154-155. ISBN: 84-8966-082-4

[12] Stallings, William. Memoria interna: DRM y SRAM. *Organización y arquitectura de computadores.* Madrid: Pearson Educación S.A., 2005, p. 151-153. ISBN: 84-8966-082-4

[13] Sánchez, Sergio. "Los microcontroladores de hoy en día" Microcontroladores y sus aplicaciones. Bloc de Sergio Sánchez. [Consultado el 09 de diciembre de 2017]. Disponible en: < https://microcontroladoressesv.wordpress.com/los-microcontroladores-de-hoy-en-dia/>

[14] Canto, Carlos. "Aplicaciones del microcontroladores" Arquitectura de los microcontroladores. [online]. Universidad Autónoma de San Luis Potosí. [Consultado el 09 de diciembre de 2017]. Disponible en: <http://galia.fc.uaslp.mx/~cantocar/microcontroladores/>

[15] Bizama Soto, Aníbal Alberto. "Empresas fabricantes de microcontroladores". 23 de noviembre de 2012. Bloc de A.A. Bizama. [Consultado el 11 de diciembre de 2017]. Disponible en: <http://anibalbizama.blogspot.com.es/2012/11/8-empresas-fabricantes-de.html>

[16] Bellido Díaz, Manuel Jesús. *Introducción al Diseño de SoC (Systems On Chip)*. [online]. Departamento de Tecnología Electrónica, Universidad de Sevilla. Febrero de 2017.

[17] Página web de Arduino. Consultado el 05 de enero de 2018. Disponible en: < https://www.arduino.cc/en/Guide/Introduction>

[18] "Raspberry Pi" Blog Historia de la Informática. 18 de diciembre de 2013. Escuela Técnica Superior de Ingeniería Informática de la Universidad Politécnica de Valencia. [Consultado el 05 de enero de 2018]. Disponible en: Disponible en: <http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/>

[19] Página web de Raspberry Pi. Consultado el 05 de enero de 2018. Disponible en: <https://www.raspberrypi.org/about/>

[20] *Intel Edison Kit for Arduino. Hardware Guide*. Marzo 2017. Consultado el 05 de enero de 2018.

[21] *Introduction to GPIO and physical computing on the Raspberry Pi*. Web de RaspBerry. Consultado el 05 de enero de 2018. Disponible en: <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/>

[22] Pérez Carrasco, Daniel. Proyecto Final de carrera "Procedimiento de mantenimiento y calibración de estación radiométrica". 2007. Universidad de Sevilla. Cap. 3 y 4.

[23] Hinckley, Alan. *Pyranometers: What you need to know*. [online]. 14 de junio de 2017. [Consultado el 16 de diciembre de 2017]. Disponible en: <https://www.campbellsci.com/blog/pyranometers-need-to-know>

[24] Bausà Aragonés, Jesús, García Gómez, Carlos y otros. Apuntes "Sensores de temperatura". Universidad Politécnica de Valencia (UPV).

[25] Kester Walt, Bryant James y Jung Walt. *Temperature Sensors*. Analog Devices. Disponible en: <http://www.analog.com/media/cn/training-seminars/design-handbooks/temperature_sensors_chapter7.pdf?doc=CN0281.pdf>

[26] Diosdado, Raúl. "La fuente de alimentación" Zona Maker. Bloc de Raúl Diosdado. [Consultado el 13 de diciembre de 2017]. Disponible en: <https://www.zonamaker.com/electronica/intro-electronica/instrumentacion/fuente-de-alimentacion>

[27] "Fuentes lineales vs Fuentes conmutadas" Ayuda electrónica. 11 de septiembre de 2009. [Consultado el 13 de diciembre de 2017]. Disponible en: < http://ayudaelectronica.com/fuente-lineal-vs-fuente-conmutada/>

[28] Marshall Brain, Charles W. Bryant y Clint Pumphrey. *How Batteries Work*. [online]. [Consultado el 11 de diciembre de 2017]. Disponible en: <https://electronics.howstuffworks.com/everyday-tech/battery.htm>

[29] Llamas, Luis. "Opciones para alimentar Arduino con baterías" Luis Llamas. 5 de marzo de 2016. [Consultado el 11 de diciembre de 2017]. Disponible en: <https://www.luisllamas.es/alimentar-arduino-baterias/>

[30] Manual de la célula compensada calibrada Atersa. Grupo Elecnor. Última revisión 17 de noviembre de 2011.

[31] Hoja de datos del sensor de temperatura LM35. Texas Instruments.

[32] Página web de compra de la EBL 2800. Disponible en: <http://www.eblmall.com/product/ebl-8-pack-aa-rechargeable-batteries-2800mah-ni-mh-1-2v-high-capacity-1200-cyclesbattery-case-included/>

[33] Hoja de datos del display LCD. Disponible en: <https://www.arduino.cc/documents/datasheets/LCDscreen.PDF>

[34] Hoja de datos del Led RGB. Disponible en: <https://www.arduino.cc/documents/datasheets/LEDRGB-L-154A4SURK.pdf>

[35] Hoja de datos de la envolvente Schneider Electric NSYTBS24198.

[36] Estatuto de los trabajadores. Artículo 34, 37 y 38.

[37] Tabla de coeficientes de amortización lineal de la Agencia Tributaria.

[38] Discontinuidad soporte Intel Edison. [Consultado el 04 de enero de 2018]. Disponible en: <https://communities.intel.com/docs/DOC-112093>

[39] S. Harshini, K.R. Kavyasri, P. Bhavishya y T. Sethukkarasi. *Digital World Bug: Y2k38 an Integer Overflow Threat-Epoch*. [online]. 31 de marzo de 2017. International Journal of Computer Sciences and Engineering. Vol. 5. [Consultado el 04 de enero de 2018]. ISSN: 2347-2693. Disponible en: <http://ijcseonline.org/pub_paper/22-IJCSE-01974-3.pdf>

[40] García, Alex. "Crear y eliminar particiones con fdisk en Linux" #rm-rf.es. 24 de julio de 2011. [Consultado el 04 de enero de 2018]. Disponible en: <http://rm-rf.es/crear-y-eliminar-particiones-con-fdisk-en-linux/>

[41] Alimentación por batería en J2. [Consultado el 04 de enero de 2018]. Disponible en: <https://communities.intel.com/message/295018#295018>

Anexos.

En este apartado se adjuntan los anexos correspondientes a:

- **Programación en Arduino.** Se añade el código de programación del dispositivo con la explicación correspondiente en comentarios.
- **Manual de usuario.** Se trata de un pequeño documento en el cual se explican las instrucciones para el manejo del dispositivo.
- **Hojas de datos.** Se adjuntan las hojas de especificaciones de los elementos esenciales del proyecto, así como el certificado de calibración de la célula.

Anexo 1: Código de programación

```
/*
Trabajo Fin de Grado (TFG)
Autor: Eduardo Bayón Alonso
Tutor: Daniel Moríñigo Sotelo
Se trata de la creación de un dispositivo con el fin de leer los valores que proporciona una célula solar calibrada, para convertirlos en W/m2 y obtener la irradiancia del lugar y posición en la cual se coloque. También se medirá la temperatura.
Estos datos se irán almacenando en una tarjeta microSD gracias a la placa Intel Edison que proporciona esta facilidad con la inclusión interna de un adaptador.
*/


/*Inclusión de las librerías*/

#include <LiquidCrystal.h> //Librería para el manejo del LCD
#include <SD.h> //Librería para la utilización de la microSD


/*Definición de los pines utilizados*/

//Definición de pines del LCD
const int rs = 2, en = 3, d4 = 7, d5 = 8, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//Definición de los pines de sensor de irradiancia y temperatura
#define sensorIrr A0
#define sensorTemp1 A1
#define sensorTemp2 A2

//Definicion de los pines del LED RGB y LED indicador del almacenamiento de datos en SD
#define pinrojo 9
#define pinverde 10
#define pinazul 11
#define LED_indicador 13

//Definición de los pulsadores y de la retroiluminación de la pantalla LCD
#define pinpuls1 A3
#define pinpuls2 A4
```

```
#define pinpuls3 A5
#define pinluz 4


/*Declaración de variables globales*/

//Variables indicadoras del tipo de pulsación en el botón
#define PULSCORTA 1
#define PULSLARGA 0

bool luz = LOW; //Variable que indica la retroiluminación del
LCD

char fecha[15], hora[15]; //Variables que almacenan el valor
de la fecha y la hora en formato (d:m:a , h:m:s)

int puls; //Variable que guarda el valor de la pulsación
(corta o larga)
int MODO = 1, MODO_ANT = 0; // Variables para indicar que
mostrar por el LCD

int parametro = 0; //Variable que indica el parámetro dia,
mes, año u hora, minuto, segundo en el que guardar el valor
de estos
int valor[6] = {1, 1, 0, 0, 0, 0};  //Vector en el que se
guardará el valor de la fecha y hora según el parámetro
int maxim[6] = {31, 12, 37, 23, 59, 59}; //Vector que indica
los valores máximos permitidos para cada parámetro
int minim[6] = {1, 1, 0, 0, 0, 0}; //Vector que indica los
valores mínimos permitidos para cada parámetro


int cont = 0; //Variable que cuenta el número de tomas por
segundo
int irr = 0, temp = 0;  //Variables que acumulan el valor "en
crudo" de las x tomas cada segundo
float irradiancia = 0, temperatura = 0; //Variables en las
que almacenar los datos una vez convertidos a sus
correspondientes en el SI

unsigned long millisLectura = 0; //Se declara la variable en
la que se guarda el tiempo en la que se produce la impresión
y el guardado de datos
unsigned long interv_global = 1000;  //Se declara la variable
que establece el tiempo entre cada actualización de datos
```

```
unsigned long millisTomaDatos = 0;  //Se declara la variable
en la que se guarda el tiempo en la que se produce la toma de
datos
unsigned long interv_datos = 40;  //Se declara la variable
que establece el tiempo entre cada toma de datos durante un
segundo de lectura
unsigned long const_time = 0; //Se declara la variable de la
diferencia de tiempo entre el introducido por el usuario y el
de la Intel

time_t timestamp; //Variable en la que se almacenará en
segundos la fecha que indique el usuario.

File archivoDatos; //Variable para el manejo de archivos en
la tarjeta microSD
/*Creación de símbolos para el LCD*/
byte grado[8] = //Símbolo para los grados Celsius
{
  0b00001100,
  0b00010010,
  0b00010010,
  0b00001100,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000
};

byte cuadrado[8] = //Símbolo de superíndice para los W/m2
{
  0b00001110,
  0b00000010,
  0b00001110,
  0b00001000,
  0b00001110,
  0b00000000,
  0b00000000,
  0b00000000
};

byte enie[8] = //Símbolo de la ñ
{
  0b00001110,
  0b00000000,
  0b00010110,
  0b00011001,
  0b00010001,
```

```
  0b00010001,
  0b00010001,
  0b00000000
};


/*************************************************************
 Función principal que se ejecuta al principio y una sola vez
*************************************************************/
void setup() {

  /*Modos en los que funciona cada pin*/
  //Entradas
  pinMode(sensorIrr, INPUT);
  pinMode(sensorTemp1, INPUT);
  pinMode(sensorTemp2, INPUT);
  pinMode(pinpuls1, INPUT_PULLUP); //Pulsador 1 . Activo a 0
por resistencia Pullup
  pinMode(pinpuls2, INPUT_PULLUP); //Pulsador 2 . Activo a 0
por resistencia Pullup
  pinMode(pinpuls3, INPUT_PULLUP); //Pulsador 3 . Activo a 0
por resistencia Pullup

  //Salidas
  pinMode(LED_indicador, OUTPUT);
  pinMode(pinluz, OUTPUT);
  pinMode(pinrojo, OUTPUT);
  pinMode(pinverde, OUTPUT);
  pinMode(pinazul, OUTPUT);

  Serial.begin(115200); //Se inicia la comunicación serie con
tasa de transferencia 115200 baudios

  lcd.begin(16, 2); //Se inicia la comunicación con el LCD
indicando el número de columnas y filas que utiliza

  //Se crean los caracteres definidos anteriormente
  lcd.createChar(0, grado);
  lcd.createChar(1, cuadrado);
  lcd.createChar(2, enie);

  lcd.clear(); //Se borra lo que hubiera en el display
  lcd.setCursor(0, 0);  //Se sitúa el cursor en la primera
fila y primera columna
  digitalWrite(pinluz, HIGH); //Se pone a nivel alto la
variable pinluz para encender la retroilumninación del LCD
```

```
  //Bucle comprobante de la inclusión de la microSD en el
adaptador
  while (!SD.begin()) { //Realizar continuamente si la SD no
se ha inicializado o no se ha introducido
    //Se mandan mensajes por monitor serie y LCD del error.
    Serial.println("Introduzca la tarjeta para empezar a
medir.");
    lcd.print("Introduce la SD");
    lcd.setCursor(0, 1);
    lcd.print(" para comenzar. ");
    delay(2000);  //Se espera 2 segundos para el siguiente
ciclo
  }
  //En caso de haberse iniciado correctamente se manda el
mensaje
  Serial.println("SD correctamente inicializada");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("SD inicializada.");
  delay(1000);  //Se espera un segundo para que se lea
correctamente el mensaje por LCD

  establecer_fecha(); //Llamada a la función que permite
establecer la fecha mediante botones
}


/*********************************************
  Función principal que se ejecuta cíclicamente
*********************************************/
void loop() {

  if (millis() - millisTomaDatos  >= interv_datos) {  //Si
han pasado 40ms(intervalo entre toma de datos) desde la última
toma de datos
    millisTomaDatos = millis();

    if (digitalRead(pinpuls1) == LOW) {   //Si el usuario
pulsa el botón 1, se guarda un 2 en la variable MODO
      MODO = 2;
      MODO_ANT = MODO;  //Se guarda la variable MODO en
MODO_ANT
    }
    if (digitalRead(pinpuls2) == LOW) {   //Si el usuario
pulsa el botón 2, se guarda un 3 en la variable MODO
      MODO = 3;
```

```
      MODO_ANT = MODO;   //Se guarda la variable MODO en
MODO_ANT
    }
    if (digitalRead(pinpuls3) == LOW) {    //Si el usuario
pulsa el botón 3, se guarda un 4 en la variable MODO
      MODO = 4;
    }

    if (millis() - millisLectura >= interv_global) {  //Si ha
pasado 1 segundo (intervalo de actualización de datos) desde
la última actualización
      millisLectura = millis();

      actualizar_fecha();   //Se   llama   a   la   función
actualizar_fecha

      conversion_sensores();   //Se   llama   a   la   función
conversion_sensores

      LED_RGB();   //Se  llama  a  la  función  LED_RGB  que
iluminará un led según el valor de la irradiancia

      switch (MODO) { //Según el modo que escoja el usuario,
se llamará a una función u otra
        case 2:
          mostrar_irradiancia();  //Si el usuario escoge ese
modo, se llama a la función mostrar_iradiancia que imprimirá
por LCD y monitor serie la irradiancia
          break;
        case 3:
          mostrar_temperatura();  //Si el usuario escoge ese
modo, se llama a la función mostrar_temperatura que imprimirá
por LCD y monitor serie la temperatura
          break;
        case 4:
          onOff();  //Si el usuario escoge ese modo, se llama
a la función onOff que apagará o encenderá el LCD
          MODO = MODO_ANT; //Vuelve al modo en el que estaba
anteriormente una vez realizada la función con el fin de que
el LCD no quede vacío
          break;
        default: //En caso de que el modo fuera otro, no se
ejecuta nada y sale.
          break;
      }
```

```
      if (cont >= 1) //En caso de que contador sea mayor o
igual que 1 se llama a la función almacenamiento para que en
la primera ejecución los valores de irradiancia y temperatura
no sean 0 al guardarlos.
        almacenamiento();

      //Cada vez que pasa el segundo se resetean los valores
para volver a acumularlos en el siguiente segundo y poder
tratarlos externamente
      irr = 0;
      temp = 0;
      cont = 0;
      Serial.println("************************************");
    }

    irr = irr + analogRead(sensorIrr);      //Sumatorio de
los valores de la irradiancia
    temp    =    temp    +    (analogRead(sensorTemp1)    -
analogRead(sensorTemp2)); //Sumatorio de los valores de la
temperatura

    cont ++;  //Se incrementa contador para controlar el
primer dato guardado
  }
}


/*************************************************************
  Función para que el usuario establezca la fecha y hora
actual, la cual se indicará en el fichero donde se guardarán
los datos de irradiancia y temperatura
*************************************************************/
void establecer_fecha() {

  Serial.println("Establezca fecha (d/m/a h:m:s)");
  lcd.print("Establezca fecha");
  info_parametro(parametro, valor); //Se llama a la función
info_parametro para que inicialmente en el LCD y el monitor
serie, una vez inicializada la SD aparezca el parámetro y su
valor correspondiente

  while (MODO == 1) { //Se realiza el bucle continuamente
mientras el MODO sea 1 (Bucle de comprobación de botones)
    //INICIO PULSADOR 1
    if (digitalRead(pinpuls1) == LOW) { //En caso de que se
haya apretado el pulsador 1...
```

```
      puls = leer_pulsador(pinpuls1); //Asigna a la variable puls, el valor que devuelve la función leer_pulsador del pulsador 1
      if (puls == PULSCORTA) {  //Si lo que se ha devuelto es un 0 (pulsación corta), se incrementa 1 el parámetro
        parametro++;
        if (parametro > 5)  //Si el párametro es mayor que 5 (segundo), vuelve a 0 (día)
          parametro = 0;
        info_parametro(parametro, valor); //Se llama a la función que actualiza la fecha y hora a mostrar
        puls = 2; //Se ajusta el valor de puls para que no interfiera en el resto de los if
      }
      if (puls == PULSLARGA) {  //Si lo que se ha devuelto es un 1 (pulsación larga), se resetea el valor de los parámetros y se vuelve al parámetro inicial (dia)
        Serial.print("Reseteando...");
        lcd.clear();
        lcd.print("Reseteando...");
        //Se asignan los valores iniciales
        parametro = 0;
        valor[0] = 1;
        valor[1] = 1;
        for (int i = 2; i <= 5; i++) {
          valor[i] = 0;
        }
        delay(1200); //Se deja un tiempo para leer en el LCD "Reseteando"
        info_parametro(parametro, valor); //Se llama a la función para actualizar el LCD y mostrar por monitor serie la fecha y hora
      }
      delay(200); //Se deja un tiempo de 200 milisegundos entre 2 posibles lecturas de pulsación
    }
    //FIN PULSADOR 1

    //INICIO PULSADOR 2
    if (digitalRead(pinpuls2) == LOW) { //En caso de que se haya apretado el pulsador 2...
      puls = leer_pulsador(pinpuls2); //Asigna a la variable puls, el valor que devuelve la función leer_pulsador del pulsador 2
      if (puls == PULSCORTA) {  //Si la pulsación ha sido corta (menor de 2 segundos)
```

```
        valor[parametro]++;   //Aumenta la variable valor del
parametro actual en 1
        if (valor[parametro] > maxim[parametro])  //En caso
de que se exceda el máximo valor de ese parámetro, se
establece el mínimo
          valor[parametro] = minim[parametro];
        info_parametro(parametro, valor); //Se llama a la
función para actualizar el LCD y mostrar por monitor serie la
fecha y hora
        puls = 2; //Se ajusta el valor de puls para que no
interfiera en el resto de los if
      }
      delay(200); //Se deja un tiempo de 200 milisegundos
entre 2 posibles lecturas de pulsación
    }
    //FIN PULSADOR 2

    //INICIO PULSADOR 3
    if (digitalRead(pinpuls3) == LOW) { //En caso de que se
haya apretado el pulsador 3...
      puls = leer_pulsador(pinpuls3); //Asigna a la variable
puls, el valor que devuelve la función leer_pulsador del
pulsador 3
      if (puls == PULSCORTA) {  //Si la pulsación ha sido
corta
        valor[parametro]--; //Disminuye la variable valor del
parametro actual en 1
        if (valor[parametro] < minim[parametro])  //En caso
de que se disminuir del mínimo valor de ese parámetro, se
establece el máximo
          valor[parametro] = maxim[parametro];
        info_parametro(parametro, valor); //Se llama a la
función para actualizar el LCD y mostrar por monitor serie la
fecha y hora
        puls = 2; //Se ajusta el valor de puls para que no
interfiera en el resto de los if
      }
      if (puls == PULSLARGA) {  //En caso de ser una pulsación
larga
        Serial.println("Guardando valores..."); //Se imprime
el mensaje por el monitor serie y por el LCD
        Serial.println("");
        lcd.clear();
        lcd.print("Guardando los");
        lcd.setCursor(0, 1);
        lcd.print("valores...");
```

```
        delay(1200);    //Se deja un tiempo para leer
corectamente el LCD
        //puls = 2; //Se ajusta el valor de puls para que no
interfiera en el resto de los if
        MODO = 0; //Se establece MODO == 0 para salir del
bucle while (ya que se ha establecido la fecha)
        //Se borra el LCD y se imprime el menú desde el que
elige el usuario qué desea que se muestre por el monitor serie
y el LCD
        Serial.println("Pulse el botón 1 para mostrar
irradiancia, 2 para mostrar temperatura y 3 para apagar o
encender el LCD");
        Serial.println("");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("MODO:  1-IRRAD");
        lcd.setCursor(0, 1);
        lcd.print("2-TEMP 3-ON/OFF");
      }
      delay(200); //Se deja un tiempo de 200 milisegundos
entre 2 posibles lecturas de pulsación
    }
    //FIN PULSADOR 3
  }

  time_t tiempo_intel = time(NULL);    //Se crea un time_t
(tipo entero) con la fecha inicial que tiene la Intel en ese
momento
  time_t tiempo_introducido = estruct_a_time_t();  //Se crea
otro time_t con la fecha introducida por el usuario
  const_time = (tiempo_introducido - tiempo_intel); //Se
calcula la diferencia entre la fecha introducida y la actual
de la Intel en una variable que será constante

  /*******************************************************/
  Serial.println("################################");
  Serial.println(tiempo_intel);
  Serial.print(asctime(localtime(&tiempo_intel)));
  Serial.println("################################");
  Serial.println(tiempo_introducido);
  Serial.print(asctime(localtime(&tiempo_introducido)));
  Serial.println("################################");
  /*******************************************************/
}


/*************************************************************
```

```
  Función que determina el tipo de pulsación que se ha
realizado en los botones
*************************************************************/
int leer_pulsador(int pin) {

  unsigned long tiempopulsado = millis(); //Se almacena el
tiempo actual (cuando se ha pulsado)
  //PULSACIÓN LARGA
  if (pin == pinpuls1 || pin == pinpuls3) { //Si se han
pulsado los botones 1 o 3 (que permiten la pulsación larga)...
    while (digitalRead(pin) == LOW) { //Se realiza el bucle
mientras esté a nivel bajo la lectura de pulsación (mientras
esté pulsado)
      if ((millis() - tiempopulsado) >= 2000) { //Si desde
que se ha pulsado el botón hasta ahora han pasado 2 segundos,
la función devuelve 1
        return 0;
      }
    }
  }
  //PULSACIÓN CORTA
  if (pin == pinpuls1 || pin == pinpuls2 || pin == pinpuls3)
{ // Si se han pulsado 1, 2 o 3 (menos de 2 segundos)
    return 1;
  }
}


/*************************************************************
  Función que imprime por el monitor serie y el LCD el
parámetro actual, así como su valor durante la ejecución de
la función establecer_fecha
*************************************************************/
void info_parametro(int parametro, int valor[]) {

  lcd.clear();
  lcd.print("Establezca fecha");
  lcd.setCursor(2, 1);
  //Según sea un parámetro u otro, imprimirá lo
correspondiente
  switch (parametro) {
    case 0:
      Serial.print("Dia:  ");
      lcd.print("Dia:");
      break;
    case 1:
      Serial.print("Mes:  ");
```

```
      lcd.print("Mes:");
      break;
    case 2:
      Serial.print("Año:  ");
      lcd.print("A");
      lcd.write(byte(2)); //Se escribe el símbolo creado para
el LCD
      lcd.print("o:");
      break;
    case 3:
      Serial.print("Hora: ");
      lcd.print("Hora:");
      break;
    case 4:
      Serial.print("Minuto: ");
      lcd.print("Minuto:");
      break;
    case 5:
      Serial.print("Segundo:  ");
      lcd.print("Segundo:");
      break;
  }
  lcd.setCursor(12, 1);
  if (valor[parametro] < 10) {  //Si el valor es menor que
10, se añade un cero delante para que quede estético
    lcd.print("0");
    Serial.print("0");
  }
  lcd.print(valor[parametro]);
  Serial.println(valor[parametro]);
}


/*************************************************************
   Función que guarda en una estructura del estándar de linux
la fecha y hora con los valores introducidos por el usuario,
y lo convierte a time_t
*************************************************************/
time_t estruct_a_time_t() {

  time_t tiempo;  //Se define variable en la que se guardará
en segundos, el tiempo introducido por el usuario
  struct tm timeinfo; //Se define la estructura en la que se
guardan los valores introducidos por el usuario en dia-mes-
año hora:minuto:segundo
```

```
  timeinfo.tm_year = valor[2] + 100 , timeinfo.tm_mon =
valor[1] - 1, timeinfo.tm_mday = valor[0],
           timeinfo.tm_hour = valor[3], timeinfo.tm_min =
valor[4], timeinfo.tm_sec = valor[5]; //Se asignan los
valores a cada parámetro de la estructura
  tiempo = mktime(&timeinfo); //Convierte el valor de las
variables de la estructura de formato separado a su
correspondiente en formato condensado (en segundos)
  return tiempo; //Se devuelve el valor tiempo
}


/**********************************************************
  Función que se encarga de la actualización de la fecha en
cada pasada del bucle en donde se la llama
**********************************************************/
void actualizar_fecha() {

  time_t fecha_actual = time(&fecha_actual);  //Se crea otra
variable local con la fecha actual de la tarjeta Intel
  timestamp = fecha_actual + const_time;  //Se asigna a la
variable timestamp global el valor de fecha_actual y la
diferencia entre la introducida por el usuario y la de la
Intel dando el valor final del tiempo
  struct tm* tiempo_struct = localtime(&timestamp);  //Se
crea una estructura para dar formato a los segundos del
timestamp con días, meses...

  // Se da formato a los valores contenidos en la estructura
colocándolos en los array de caractéres fecha y hora
  strftime(hora, sizeof hora, "%H:%M:%S", tiempo_struct);
  strftime(fecha, sizeof fecha, "%d-%m-%Y", tiempo_struct);
}


/**********************************************************
   Función que procede a la conversión de los datos de
irradiancia y temperatura acumulados, para su posterior
muestra
**********************************************************/
int conversion_sensores() {

  float valorSensorIrr = irr / 25.0;  //Se guarda en la
variable de irradiancia el valor promediado de la irradiancia
acumulada
```

```
  float voltaje0 = (valorSensorIrr * 5000) / 1024.0; // Se
convierte la lectura analógica (Rango 0 - 1023) a voltaje (0
- 5000mV)
  irradiancia = voltaje0 * 1000 / 400.0;   // Se convierte a
W/m2 por medio del datasheet de la placa solar  que indica:
100mV igual a 1Kw/m2 = 1000W/m2
  //Debido al circuito de amplificación que aumenta la
ganancia 4 veces, se divide entre 400 en vez de 100

  float valorSensorTemp = temp / 25.0;  //Se guarda en la
variable de temperatura el valor promediado de la temperatura
acumulada
  float voltaje1 = (valorSensorTemp * 5000) / 1024.0; // Se
convierte la lectura analógica (Rango 0 - 1023) a voltaje (0
- 5000mV)
  temperatura = voltaje1 / 30.0;  //Se convierte a °C ya que
según datasheet LM35, la ganancia es de 10 --> 10mV es 1 °C
  //Debido al circuito de amplificación que aumenta la
ganancia 3 veces, se divide entre 30 en vez de 10
}


/*************************************************************
   Función que se encarga de crear un fichero en la SD con el
nombre de la fecha introducida por el usuario, y guardar la
fecha, el valor de la irradiancia y la temperatura en crudo,
para su posterior procesamiento en un programa externo.
*************************************************************/
void almacenamiento() {

  //Creación del nombre del archivo para guardar los datos
  char fichero[20];
  strcpy(fichero,  fecha);  //Se  copia  en  el  vector  de
caractéres "fichero" el valor de la cadena fecha
  strcat(fichero,  ".txt");  //Se  concatena  fichero  con  el
texto ".txt" para la creación del archivo posterior

  archivoDatos  =  SD.open(fichero,  FILE_WRITE);  //Se  abre
fichero  o  se  crea  en  caso  de  no  existir  en  modo  lectura-
escritura
  if (archivoDatos) {   //Si se ha podido abrir el fichero...
    digitalWrite(LED_indicador, HIGH);  //Se enciende el LED
indicador
    archivoDatos.print(timestamp);   //Se  escribe  en  el
fichero el tiempo en formato condensado
    archivoDatos.print(",");
```

```
    archivoDatos.print(irr); //Se escribe la lectura de la
irradiancia acumulada (las 10 tomas) en el fichero
    archivoDatos.print(",");
    archivoDatos.print(temp);  //Se escribe la lectura de la
temperatura acumulada (las 10 tomas) en el fichero
    archivoDatos.println("");
    archivoDatos.close(); //Se cierra el fichero para que se
guarden correctamente los datos
  }
  else {  //En caso de no haberse podido abrir el fichero, se
envía un mensaje de error
    Serial.println("Error de apertura del archivo, reinicie
dispositivo.");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Error apertura." );
    lcd.setCursor(0, 1);
    lcd.print("Reinicie aparato" );
  }
  digitalWrite(LED_indicador,  LOW);  //Se  apaga  el  LED
indicando la finalización del archivado de los datos
}


/*********************************************************
   Función que iluminará un LED RGB con diferentes colores
según sea la medición de la irradiancia en ese momento
*********************************************************/
void LED_RGB() {

  if  (irradiancia  <  1750)  {  //Los  intervalos  están
establecidos de 1750 en 1750. Color rojo
    digitalWrite(pinrojo, 100);
    digitalWrite(pinazul, 0);
    digitalWrite(pinverde, 0);
  }  else if (irradiancia < 1750 && irradiancia >= 3500)
{  //Color amarillo
    digitalWrite(pinverde, 100);
    digitalWrite(pinazul, 100);
    digitalWrite(pinrojo, 0);
  } else if (irradiancia < 3500 && irradiancia >= 5250)
{  //Color verde
    digitalWrite(pinverde, 100);
    digitalWrite(pinazul, 0);
    digitalWrite(pinrojo, 0);
  } else if (irradiancia < 5250 && irradiancia >= 7000)
{  //Color morado
```

```
    digitalWrite(pinverde, 0);
    digitalWrite(pinazul, 100);
    digitalWrite(pinrojo, 100);
  } else if (irradiancia >= 7000) {    //Color azul
    digitalWrite(pinazul, 100);
    digitalWrite(pinverde, 0);
    digitalWrite(pinrojo, 0);
  }
}


/*********************************************************
   Función que mostrará la irradiancia del momento tanto por
el monitor serie como por el display LCD, así como la fecha
de la toma de esos datos por el monitor serie
*********************************************************/
void mostrar_irradiancia() {

  Serial.print("Fecha y hora: ");
  Serial.print(fecha);
  Serial.print("  ");
  Serial.println(hora);
  Serial.print("El valor de la irradiancia es: ");
  Serial.print(irradiancia);
  Serial.println(" W/m2");

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" La irradiancia " );
  lcd.setCursor(0, 1);
  lcd.print(" es: " );
  lcd.print(irradiancia);
  lcd.print("W/m");
  lcd.write(byte(1));
}


/*********************************************************
   Función que mostrará la temperatura del momento tanto por
el monitor serie como por el display LCD, así como la fecha
de la toma de esos datos por el monitor serie
*********************************************************/
void mostrar_temperatura() {

  Serial.print("Fecha y hora: ");
  Serial.print(fecha);
  Serial.print("  ");
```

```
  Serial.println(hora);
  Serial.print("El valor de la temperatura es: ");
  Serial.print(temperatura);
  Serial.println("°C");

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" La temperatura " );
  lcd.setCursor(0, 1);
  lcd.print("  es: " );
  lcd.print(temperatura);
  lcd.write(byte(0));
  lcd.print("C");
}

/*********************************************************
   Función que se encargará de apagar o encender el LCD según
se pulse el tercer botón del dispositivo
*********************************************************/
void onOff() {

  if (luz == 1) { //En caso de estar encendido
    lcd.noDisplay();  //El LCD deja de mostrar lo que está
mostrando
    luz = !luz;    //Cambia el valor de la variable de
retroiluminación
    digitalWrite(pinluz, luz);    //El LCD apaga la
retroiluminación
  } else if (luz == 0) {  //Si no, si se encuentra apagado
    lcd.display();  //El LCD muestra de nuevo lo que estaba
mostrando
    luz = !luz;  //Cambia el valor de la variable de
retroiluminación
    digitalWrite(pinluz, luz);    //El LCD enciende la
retroiluminación
  }
  Serial.println(luz);
}
```

Anexo 2: Manual de usuario

Descripción del dispositivo.

El dispositivo se trata un instrumento autónomo y de bajo coste que permita capturar y guardar la información que proporciona una célula solar calibrada y un sensor de temperatura. A su vez, permite que los datos sean guardados en un elemento de memoria (MicroSD), y mostrados por un LCD. Dispone también de un led RGB que dependiendo de la irradiancia del momento se ilumina en un color u otro.

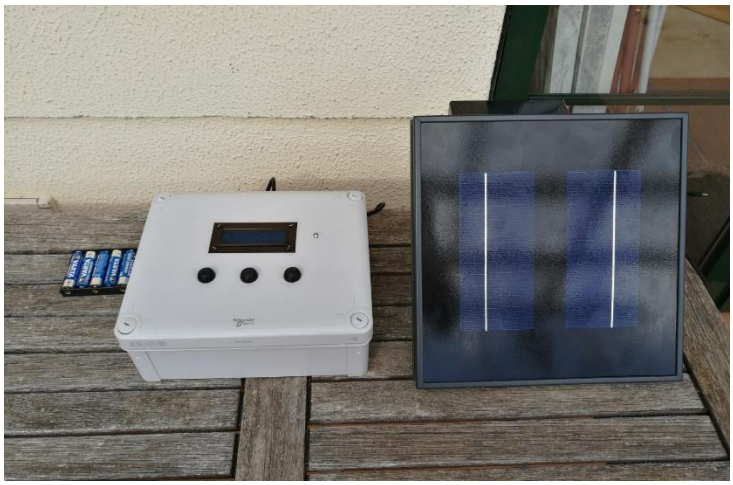

Figura 87. Dispositivo.

Encendido y apagado del dispositivo.

Para encender el dispositivo, éste se debe conectar a una fuente de alimentación, bien por un adaptador de red conectado al Jack DC o bien por una batería conectada al microUSB o al Jack DC.

Una vez se encuentra conectado, basta con poner el interruptor situado en la parte trasera de la caja en la posición “ON” para que comience su funcionamiento. Se han de esperar unos segundos a que la placa se encienda correctamente y cargue el programa.

Para el correcto apagado del dispositivo basta con mover el interruptor a la posición “OFF” primeramente, y en caso de estar conectado a la red, desconectarlo.

Control del dispositivo.

El sistema está diseñado para que su control sea fácil e intuitivo. Este control se realiza por medio de los 3 pulsadores localizados en la cara superior de la caja.

Una vez encendido el dispositivo, el LCD mostrará un mensaje que indicará al usuario que introduzca la tarjeta MicroSD para comenzar. Hasta que no se haya introducido el dispositivo no comenzará. Cuando se ha inicializado correctamente la tarjeta, el dispositivo pide al usuario que establezca la fecha y hora actual. Esto se realiza por medio de los pulsadores.

- Botón 1. Este botón permite el cambio de parámetro del sistema (Día – Mes – Año – Hora – Minuto – Segundo)
  Una pulsación corta, significa el paso de un parámetro a otro.
  Una pulsación larga (más de dos segundos), resetea los valores de todos los parámetros y vuelve al parámetro Día.
- Botón 2. Este botón permite el cambio de valor del parámetro en el que se encuentra. Una pulsación aumenta en uno el valor.
- Botón 3. Este botón permite el cambio de valor del parámetro en el que se encuentra. Una pulsación corta, disminuye en uno el valor del parámetro. Una pulsación larga (más de dos segundos), guarda los valores de los parámetros y da paso al menú de selección de modo.

Una vez se han guardado los valores, el dispositivo comienza a medir y a guardar los valores. A su vez, el LCD muestra un menú en el que el usuario puede elegir entre 3 opciones distintas:

- Botón 1. Pulsando este botón el dispositivo mostrará por el LCD la irradiancia incidente en la célula en ese momento.
- Botón 2. Pulsando este botón el dispositivo mostrará por el LCD la temperatura ambiente en ese momento.
- Botón 3. Pulsando este botón, se podrá apagar o encender el LCD según el usuario lo desee.

Monitor serie.

El dispositivo también se puede alimentar conectándolo a un ordenador por medio de microUSB. Una vez conectado se ha de abrir el IDE de Arduino para poder modificar parámetros de programación como puedan ser el ciclo de guardado y toma de datos, el rango de irradiancia para la iluminación del led, etc. Con la apertura de la pestaña del monitor serie, se visualizará la misma información que la mostrada por la pantalla LCD.

Mantenimiento.

El dispositivo no requiere de un mantenimiento excesivo, ya que la envolvente se encarga de proteger los elementos más críticos. Puesto que el dispositivo estará expuesto durante un tiempo al aire libre, se recomienda pasar un paño en cada uso de éste.