ANEJO I: COMPONENTES DEL SISTEMA HARDWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

En el actual anejo se encuentran detallados todos los elementos de hardware necesarios para llevar a cabo la solución propuesta. De todos ellos se realiza una justificación de su necesidad, explicando la función que desempeñan en el equipo, así como el motivo de su elección y sus características técnicas específicas.

El documento se encuentra dividido en dos apartados. Por un lado, se tratan todos los componentes necesarios para llevar a cabo el dispositivo de monitorización diseñado como solución. Por otro, se tratan otros componentes complementarios al equipo principal necesarios para llevar a cabo el testeo e implementación del dispositivo solución. Entre los componentes de este segundo grupo se encuentran el sensor, los componentes necesarios para su alimentación y los necesarios para el acondicionamiento de la señal del mismo al equipo. Por tanto, el anejo se dividirá en componentes del equipo principal de monitorización y componentes complementarios al mismo.

1. COMPONENTES EQUIPO DE MONITORIZACIÓN SISTEMA DAQ

A continuación, se desarrolla el documento en torno a los elementos de hardware necesarios para llevar a cabo el dispositivo de monitorización resultante de la solución propuesta anteriormente.

1.1. SISTEMA DE ALIMENTACIÓN

El equipo funciona mediante corriente continua, con un voltaje operativo de 5V y un voltaje de entrada de 7-12 V, siendo el óptimo 9V. Para su alimentación se recomienda por tanto emplear un voltaje de entrada de 9V. Esto es debido a que con voltajes de entrada menores (7-9V) pueden surgir problemas de mal funcionamiento por escasez de voltaje en momentos puntuales. Y con voltajes mayores (9-12V) se pueden dar problemas por sobrecalentamiento al tener que disipar mayor cantidad de energía en forma de calor ya que el voltaje operativo del microcontrolador es de 5V. En ningún caso se podrán sobrepasar los voltajes de entrada límites para su alimentación, siendo estos de 6-20V.

Teniendo en cuenta que el dispositivo necesita una corriente continua de alimentación de 9V y que su instalación se lleva a cabo en campo, donde se encuentra aislado y sin una posible conexión a red, es inevitable que el mismo disponga de un sistema propio e independiente de alimentación para su funcionamiento.

Tomando en consideración las premisas anteriormente descritas el sistema de alimentación del equipo tiene una parte dedicada al suministro de la energía eléctrica de alimentación, y otra a su acondicionamiento para una correcta alimentación del equipo. A continuación, se describen y analizan las distintas partes del sistema de alimentación.

1.1.1. Suministro de energía eléctrica de alimentación

Como anteriormente se ha comentado es necesario que el equipo disponga de un sistema propio e independiente de alimentación que permita la instalación y perfecto funcionamiento del equipo en cualquier punto del campo. Teniendo en cuenta este aspecto surgen las siguientes opciones:

La primera opción es alimentar el dispositivo directamente de la red eléctrica mediante el uso de un adaptador de corriente. Con ello el equipo se alimentaria constantemente sin la necesidad de almacenar electricidad en baterías. Dada la dependencia que tendría entonces el equipo a un lugar cercano del que obtener energía, el cual habitualmente en campo no existe, esta opción es descartada.

La siguiente es que la alimentación se realice mediante el uso indirecto de la red eléctrica con el uso de baterías. Para ello, las baterías se cargan de la red eléctrica y se llevan a campo donde se conectan al equipo. Una vez descargadas se cambian por otras baterías cargadas y se cargan las baterías descargadas. A pesar de que esta opción permite la instalación del dispositivo en cualquier lugar, este debe ser accesible y la unidad sigue siendo dependiente al necesitar un continuo cambio de baterías. Además, tiene el riesgo de agotarse la batería y que el equipo no funcione, por lo que se descarta su uso.

La tercera opción es obtener la energía eléctrica de fuentes alternativas a la red empleando energías renovables. Para ello se utiliza un panel solar, el cual carga una batería en la que se almacena la energía para los periodos en los que no haya luz solar que genere la electricidad necesaria. Con ello se consigue que el dispositivo tenga una fuente continua, propia e independiente de energía permitiendo su instalación en cualquier lugar. El sistema se ha dimensionado para mantener su autonomía durante varios días con baja radiación solar. Además, el equipo cuenta con sensores de voltaje tanto en la placa solar como en la salida de la batería con los que se conocen el estado de los mismos y se da aviso si surge algún problema.

De todas las opciones de alimentación de la unidad esta última, en la que se emplean paneles solares y baterías, es la más adecuada y la empleada como sistema de suministro energético. Ello debido a que permite la instalación del equipo en cualquier lugar del campo, sin dependencia a ningún elemento externo y sin la necesidad de un mantenimiento frecuente.

Para llevar a cabo este sistema de suministro de energía eléctrica son necesarios los elementos a continuación descritos.

1.1.1.1. Panel solar 12V

Como ya se ha visto, la energía eléctrica necesaria para la alimentación del equipo se obtendrá mediante el uso de placas solares ya que estas permiten obtener dicha energía en cualquier lugar sin necesidad de nada más que luz solar, lo que hace posible que el equipo sea autónomo e independiente. Las placas dan esta posibilidad ya que son capaces de obtener energía eléctrica de la luz solar mediante el conjunto de células fotovoltaicas de que están formadas ya que al incidir a través de ellas los fotones de la luz solar, se genera una corriente eléctrica. Las células se montan sobre el panel solar en serie para alcanzar la tensión deseada a la salida del módulo fotovoltaico y posteriormente se van conectando en paralelo hasta llegar a la corriente deseada. Las células fotovoltaicas pueden ser de varios materiales, aunque el más común es el silicio cristalino. Según la fabricación de las células nos encontramos con módulos fotovoltaicos monocristalinos o policristalinos (VIDAL, 2018).

Monocristalinos: Las células están formadas de un único cristal de sílice. Suelen tener un rendimiento de 14-16%.

Policristalino: formado por varios cristales de sílice o de otros materiales con lo que el rendimiento suele ser próximo al 12-14%.

Ambos tipos son muy utilizados. Además de estos dos tipos de paneles solares también existen los paneles silicio amorfo (capa fina). La tecnología del silicio amorfo a-Si tiene una eficiencia considerablemente menor que las basadas en silicio cristalino, debido principalmente a la mala calidad del silicio utilizado, cuya estructura interna dificulta la recolección de los portadores fotogenerados. Sin embargo, son especialmente adecuadas para uso en ambientes de luz parcial o indirecta y en atmósferas con mucho polvo.

El módulo fotovoltaico elegido para obtener la energía eléctrica de alimentación del equipo es el panel de silicio policristalino SHS modelo ESPMC050 fabricado por Era Solar, Figura 1.

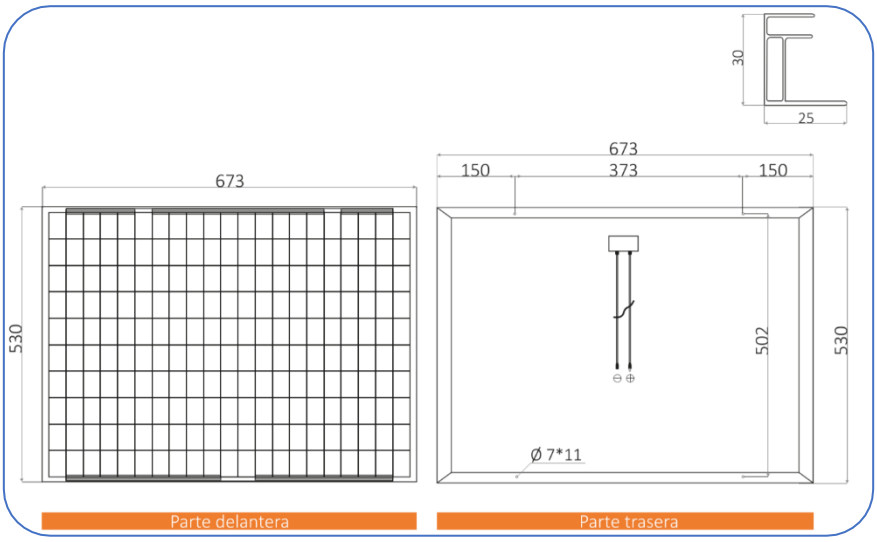


Figura 1. Plano acotado del panel solar seleccionado, modelo ESPMC050 fabricado por Era Solar.

Se ha escogido este panel solar debido a que el fabricante da una garantía de 25 años y a que da una potencia de 50W, que es la potencia recomendada para el correcto funcionamiento del regulador de carga y por ello para la carga de la batería, véase apartado 1.1.1.3 Módulo 18650 controlador de carga mppt. También se ha escogido por ser un módulo de silicio policristalino, que es el tipo de panel de mejor relación rendimiento/coste.

A continuación, se muestra en la Tabla 1 las principales características eléctricas del panel.

Tabla 1. Características eléctricas en condiciones de prueba estándar del módulo solar ESPMC050 fabricado por Era Solar.

Potencia máxima (Pmax)	50 W
Tolerancia de potencia de salida	±3%
Voltaje máximo de potencia (Vmp)	18,3 V
Corriente máxima de potencia (Imp)	2,73 A
Voltaje en circuito abierto (Voc)	22,7 V
Corriente de cortocircuito (Isc)	2,9 A
Temperatura normal de funcionamiento	45°C
Voltaje máximo del sistema	DC600V
Fusible de serie del modulo	10A
Peso	4,1 kg
Dimensiones	532x674x30mm

Los datos se refieren a las Condiciones Estándar de Medida, una aproximación del funcionamiento a plena luz solar (STC: irradiación de 1000W/m^2, temperatura de la célula a 25°C, espectro AM:1.5).

1.1.1.2. Batería litio

En toda instalación aislada, como sucede en este caso, es necesaria la presencia de un acumulador. Debido a que la naturaleza de la radiación solar es intrínsecamente variable en el tiempo, sometida por un lado al ciclo diario de los días y las noches; por otro, al ciclo anual de las estaciones y, por último, a la variación aleatoria del estado de la atmósfera. Como consecuencia de todo ello, son muchos los momentos en los que la potencia eléctrica que puede entregar el módulo solar difiere, por exceso o defecto, de la que demanda el dispositivo. Para solventar este problema en los sistemas fotovoltaicos se emplea la acumulación electroquímica de la energía obtenida de los módulos solares, es decir, se hace uso de baterías recargables, que

se definen como una asociación continua formada por un conjunto de vasos electroquímicos interconectados, que presentan la propiedad de que se pueden cargar (acumular energía eléctrica en forma de energía química) y que se pueden descargar (convierten la energía química en eléctrica), entregando energía eléctrica a una carga conectada a los dos electrodos de que disponen. Todas las baterías constan de un par de electrodos (uno positivo y otro negativo) y un electrolito(BORDÓN, 2010). Con la batería recargable se consigue una entrega estable de energía eléctrica a lo largo del tiempo independientemente de las condiciones ambientales.

El equipo desarrollado, al encontrarse aislado y basar su alimentación en la energía solar, tiene este problema. Por ello resulta necesario la instalación de una batería que proporcione una entrega estable de energía a lo largo del tiempo. Existen distintos tipos de baterías en el mercado como las de plomo, níquel-hierro o níquel-cadmio y Litio entre otras. De entre todos los tipos de baterías existentes en la actualidad las baterías de Litio recargables son las más usadas en los dispositivos portátiles, ya que proporcionan una mayor densidad de carga siendo mucho más ligeras y pequeñas, no poseen efecto memoria, su coste de fabricación es considerablemente menor, mantienen más tiempo la carga cuando no se usan y proporcionan un elevado número de ciclos de carga (ABREU-CUSTODIO, 2015).

Por ello se ha instalado en el equipo una batería de Litio. Teniendo en cuenta lo anteriormente mencionado y que el sistema de suministro de la alimentación ha de tener una tensión suficiente para su posterior acondicionamiento para alimentar el equipo a 9V, la batería seleccionada es el modelo 12V12A-1D2X de la marca Vari Core, Figura 2. Se ha elegido este modelo en concreto respecto al resto de baterías de litio debido a que suministra una tensión de 12V suficiente para ser acondicionada posteriormente sin ser excesiva, así como por su bajo coste, tamaño comedido y alta capacidad de almacenaje (12Ah). Además, la capacidad de almacenaje permite alimentar el equipo unas 20 horas sin energía solar, así como una rápida carga, ya que se puede cargar por completo en menos de 4,5 horas con la placa solar instalada. Este hecho hace que la batería seleccionada tenga una capacidad de acumulación de energía óptima. Esto es debido a que con capacidades de servicio mayores se requieren tiempos de carga mayores a la vez que aumenta el espacio ocupado y el peso del equipo, no siendo necesaria mayor autonomía. Con capacidades de servicio menores, aunque requieran menores tiempos de carga, menos espacio y peso, son insuficientes en momentos de climatología adversa. A continuación, se detallan en la Tabla 2 sus características principales.

Figura 2. Batería seleccionada de Li-Ion, marca Vari Core de 12V.

Tabla 2. Características principales de la batería de Li-Ion modelo 12V12A-1D2X de la marca Vari Core.

Parámetros Nominales	Tipo	Li-Ion, 3S6P, celdas 18650
	Voltaje	12V
	Capacidad	12Ah
	Capacidad	144Wh
Dimensiones (Largo x Ancho x Alto)		112x70x56(mm)
Peso		0,9Kg
Corriente de descarga estándar		0,2C (2,4A)
Corriente de descarga máxima		1C (12A)
Resistencia interna		250mohm
Protección sobrecorriente, sobrecarga, sobredescarga y cortocircuito		SI
Voltaje de carga		12.6V
Corriente de carga		2A
Corriente de carga máxima		5A
Temperatura funcionamiento		0-60ºC

Las baterías de litio tienen grandes ventajas como ya se ha indicado anteriormente respecto a otros tipos de baterías, pero también tiene ciertas desventajas o aspectos negativos a tener en cuenta. Este tipo de baterías son sensibles a las altas temperaturas y a las tensiones muy altas o muy bajas que pueden provocar que se dañe la batería e incluso que sean peligrosas dado que pueden llegar a explotar por sobretensión o sobrecalentamiento. Para evitar estos problemas se ha de evitar la exposición de las mismas a altas temperaturas y se ha de realizar la carga y descarga de las mismas de forma adecuada, véase Figura 3.

En cuanto a la descarga no se ha de superar la corriente máxima de descarga a fin de evitar calentamientos. En este caso el equipo tiene una intensidad de corriente para su funcionamiento bastante inferior a la que puede ofrecer la batería por lo que este aspecto no será un problema. Además, es recomendable no superar el 80% de la capacidad de descarga puesto que a partir de ese punto la descarga es mucho más rápida y por ello difícil de controlar y en baterías de varias celdas como es el presente caso se pueden descargar unas celdas antes que otras, lo que puede producir un deterioro prematuro de la batería. Para evitar este problema se ha instalado un módulo controlador de carga MPPT y, además, la batería seleccionada tiene incorporada protección frente a sobredescargas.

Si analizamos el proceso de carga resulta que no se deben superar los 5 A de intensidad de corriente de carga, ni se debe sobrecargar, puesto que, si no se dañaría la batería e incluso podría sobrecalentarse y explotar. Dado que el panel solar instalado ofrece una corriente máxima de 2,73 A y la batería tiene protección frente a sobrecarga estos inconvenientes no pueden presentarse. Además, la tensión de alimentación también ha de ser constante y no superior a la corriente máxima de carga del acumulador, que en este caso es de 12,6V, puesto que se dañaría la batería y podría incendiarse, así como tampoco puede ser cercana a la tensión mínima de la batería puesto que se podría dañar. Otro inconveniente son las distintas fases de carga que necesitan este tipo de baterías, en las que en la primera fase es necesario una tensión constante y en la siguiente fase una carga a pulsos. Estos inconvenientes quedan resueltos con la instalación del regulador tipo MPPT que es el único capaz de tener en cuenta el estado de la batería y cargarla de forma adecuada a su estado y fase de carga de cada momento.

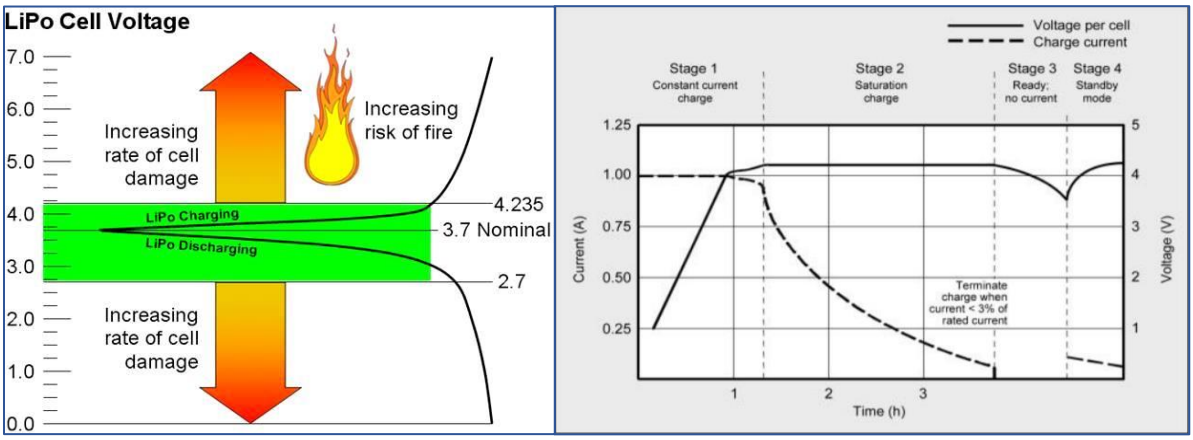


Figura 3.Diagramas de carga de baterías de litio. A la izquierda se muestra el riesgo de daño y fuego en función del voltaje de las celdas de la batería. A la derecha las distintas fases de carga con la intensidad a aplicar en cada una de ellas a lo largo del tiempo.

1.1.1.3. Modulo 18650 controlador de carga mppt

El controlador de carga, o también conocido como regulador de carga, es el componente de la instalación encargado de dirigir y controlar la energía que circula entre los módulos solares fotovoltaicos y la batería, en definitiva, el regulador de carga evita las sobrecargas y sobredescargas de las baterías cuando están recibiendo energía fotovoltaica de los paneles solares de la instalación. Por tanto, el regulador de carga se sitúa entre los módulos fotovoltaicos y las baterías. Dado que la batería instalada tiene procesos de carga y descarga delicados, como se ha explicado en el apartado anterior, este componente es de gran importancia en la instalación.

En el mercado se pueden encontrar principalmente dos tipos de controladores de carga, siendo estos los reguladores de carga pulse-width modulation (PWM), y los reguladores de carga Maximum Power Point Tracker (MPPT). Ambos se encargan de lo mismo, controlar el flujo de energía entre el campo fotovoltaico y las baterías, pero difieren en la tensión de funcionamiento y por tanto en las aplicaciones en las que deben ser usados.

-Controlador de carga PWM:

Los reguladores PWM son reguladores sencillos que actúan como interruptores entre las placas fotovoltaicas y la batería y se caracterizan por hacer que los módulos trabajen a la tensión de la batería. La energía a un lado y al otro del regulador es la misma, con los valores de tensión y corriente iguales también. En su funcionamiento estos introducen la carga de forma gradual, a pulsos de tensión, y cuando se alcanza la etapa de absorción en la carga de la batería, el regulador modifica la intensidad de los pulsos, corta varias veces por segundo el contacto entre los módulos y la batería, evitando que la batería se sobrecargue. Todo ello hace que los módulos no trabajen en su punto de máxima potencia, sino en el que impone la batería según su estado de carga, produciendo una pérdida de energía, que puede llegar hasta el 25-30%.

Entre las ventajas de estos reguladores podemos encontrar su sencillez, su reducido peso y su bajo precio, así como una vida útil bastante larga.

En cuanto a los inconvenientes, encontramos que el voltaje nominal debe ser el mismo que el del banco de baterías, por lo que no se pueden emplear paneles de voltaje muy superiores a los de las baterías y limitan el crecimiento del sistema de alimentación siendo menos versátiles. Además, también tienen un menor rendimiento frente a los reguladores MPPT.

-Controlador de carga MPPT:

Respecto a un regulador de carga MPPT se caracteriza por incluir un controlador de punto de máxima potencia, de ahí sus siglas Maximum Power Point Tracking y un transformador CC-CC, encargado de convertir la corriente continua de alta tensión en corriente continua de una tensión inferior para cargar la batería. Además, a diferencia del regulador PWM que considera solo la corriente al final de la carga, los reguladores MPPT consideran tanto la tensión como la corriente del panel solar y carga de la batería, transformando en corriente la diferencia de tensión entre ambos puntos.

Estos mecanismos permiten que el regulador de carga MPPT pueda obtener en cada momento la máxima potencia y un rendimiento óptimo, a la vez que es capaz de controlar el estado y fases de carga y descarga de la batería. Gracias a ello, el regulador es capaz de optimizar dichos procesos, haciéndolos más seguros y evitando que se estropee la batería, ya que en ellos es capaz de variar tanto la tensión como la corriente que alimentan la batería adaptándolos y optimizándolos en cada momento en función del estado y fase de carga en que se encuentre. También debido a sus características estos reguladores permiten emplear paneles de voltaje muy superiores a los de las baterías, pudiendo crecer el sistema de alimentación haciéndolos más versátiles, hecho que no es posible realizar con reguladores PWM.

Asimismo, este tipo de reguladores son idóneos para realizar la regulación de carga de baterías de litio. Esto es debido a que, por una parte, son los únicos capaces de proporcionar una corriente eléctrica de carga constante (no a pulsos como los PWM) necesaria para cargar dicho tipo de baterías. Por otra, como se ha visto en el apartado dedicado a la descripción de la batería, las baterías de litio son muy delicadas y requieren de un control de su estado y de sus procesos de carga y descarga específicos que tan solo los reguladores MPPT son capaces de proporcionarles como se ha visto anteriormente dadas las características de los mismos.

Entre sus desventajas tenemos su mayor coste.

De entre estos dos tipos de controladores por tanto se selecciona uno de tipo MPPT. Esto es debido principalmente a que la batería del equipo es de litio y como se ha visto anteriormente los reguladores MPPT son los únicos capaces de realizar la regulación de la carga y descarga de baterías de litio de forma adecuada. Además, este tipo de reguladores son más seguros, eficientes y capaces de obtener más potencia energética a la vez que son capaces de controlar el estado y fases de carga y descarga de la batería en cada momento de forma óptima.

Se ha seleccionado el controlador MPPT de carga solar BQ24650 de 12V para carga de baterías de litio 3s de celdas tipo 18650, del vendedor Walfront, Figura 4.

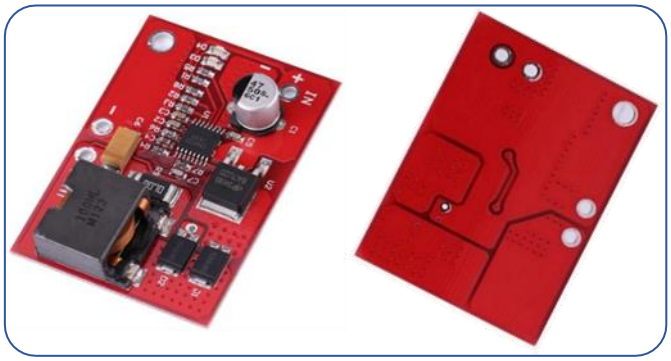


Figura 4. Regulador de carga solar MPPT seleccionado, modelo BQ24650 de 12V.

Se ha seleccionado este controlador de carga por tratarse de un regulador MPPT específico para el control de carga de baterías de litio con celdas de tipo 18650 dispuestas en 3 bloques en serie a partir de la energía obtenida de un panel solar, como es el caso de nuestro equipo. Este hecho le hace capaz de proteger mejor a la batería frente a sobrecargas y sobredescargas así como identifica, controla y gestiona mejor los procesos de carga y descarga frente a otros reguladores MPPT. Esto es debido a que este regulador impide que específicamente este tipo de baterías se cargue a una velocidad superior a 1C puesto que la corriente constante máxima de salida es de 3 A, así como que se cargue a tensiones inadecuadas, al tener una tensión de funcionamiento de 12 V. Además tiene función de gestión de carga por goteo, corriente constante y tensión constante también específicas. Estas le confieren gran precisión en el proceso de adaptación a las distintas situaciones y fases de carga y descarga, siendo capaz de proporcionar tensión y corriente constantes en la primera fase de carga, así como cambiarlas y adaptarlas a la segunda fase de carga en el momento más preciso dada su especificidad. Con ello se consigue una carga óptima y segura que favorece a la vida útil de la batería del equipo, que es uno de los componentes más caros del dispositivo. Este hecho es de gran importancia ya que las baterías instaladas son de litio y tienen un proceso de carga delicado, como se explica en el apartado dedicado a la batería (ABREU-CUSTODIO, 2015).

A ello podemos sumar su pequeño tamaño respecto a los MPPT más comunes, así como su bajo precio comparado con otros similares.

En la Tabla 3 se muestran las características principales del controlador MPPT seleccionado.

Tabla 3. Características del controlador de carga solar MPPT BQ24650 de 12V

Batería	Litio Li-ion 18650, 12V, 3s
Voltaje nominal sistema batería	12V
Potencia entrada recomendada	40/60W
Potencia Máxima Entrada	60W
Voltaje circuito abierto fotovoltaico	25V
Voltaje de entrada	18 V (tensión nominal de panel solar)
Voltaje de salida Nominal	12 V (10,8-11,1 V)
Voltaje de salida en carga máxima	12,6 V
Corriente constante máxima de salida	3 A
Dimensiones(mm) (Largo x Ancho x Alto)	45x31x23
Luz indicadora	Rojo=cargando, Rojo + Verde=cargado
Refrigeración	Pasiva

1.1.1.4. Sensor de tensión

Los sensores de tensión empleados en el dispositivo sirven para medir la tensión en puntos de interés de los circuitos de alimentación tanto del procesador como del LVDT. Estos datos son leídos a través de los sensores por el procesador, que los guarda en la SD y envía a la nube. Con ello se pretende monitorizar el estado de la alimentación para así poder enviar advertencias por falta de energía, comprobar el correcto funcionamiento del sistema de carga de batería y diagnosticar fallos en la alimentación.

Los sensores de tensión empleados son módulos FZ0430, Figura 5, que permiten medir hasta 25V en circuitos de corriente continua con procesadores tipo Keyestudio Mega. El FZ0430 consta principalmente de 2 clemas de conexión de entrada y terminales de salida para conexión rápida al microprocesador. Para su funcionamiento consta de un divisor de tensión con resistencias de 30KΩ y 7,5KΩ, lo que supone que la tensión de salida sea igual a la de entrada del módulo

dividida por un factor de 5. A continuación, se muestran las características principales en la Tabla 4 y sus conexiones con el microcontrolador en la Tabla 5.

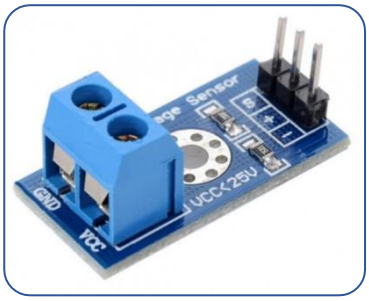


Figura 5. Sensor de tensión seleccionado, modelo FZ0430.

Tabla 4. Características del módulo sensor de tensión FZ0430

	Alimentación 5 Vcc	Alimentación 3,3 Vcc
Rango de entrada de voltaje	0 – 25 V	0 – 16,5 V
Voltaje detección entrada máximo	25 V	16,5 V
Rango de detección de voltaje	24,41mV – 25 V	
Resolución analógica de tensión:	0,00489 V en CC	
Voltaje detección entrada mínimo	24,45mV (4,89mV x 5 = 24,45mV)	

Tabla 5. Conexiones del procesador con el módulo sensor de tensión FZ0430

PROCESADOR	MÓDULOS FZ0430
5v	Vcc
Gnd	Gnd
ADC, Pin A8, A9 y A10	Señal

Este módulo ha sido elegido por su sencillez y rapidez de montaje. Destaca también su bajo coste y la buena precisión en la medida de tensión.

En el equipo principal se emplean dos de estos sensores, uno ubicado en la alimentación del controlador de carga y otro en la salida de la batería. Con el primero de ellos podemos conocer el voltaje que emite la placa solar y que recibe el controlador de carga. El segundo de ellos nos indica la carga de la batería.

En el circuito de alimentación del L.V.D.T. se han instalado también dos módulos FZ0430 uno ubicado en la alimentación del controlador de carga y otro en la salida de la batería con los mismos propósitos que los instalados en el equipo principal.

1.1.1.5. Indicador nivel de carga

A fin de conocer el nivel de carga de la batería del equipo de forma rápida, sencilla y presencial se ha instalado un pequeño modulo que indica el nivel de carga de la misma. Se trata del módulo indicador de nivel de carga para baterías de Litio 18650 3S, modelo xw228dkfr4 de la marca Xtori, Figura 6. Este módulo indica el nivel de carga mediante la iluminación progresiva de la figura de una batería, la cual se encuentra seccionada en cuatro partes. Mediante la iluminación progresiva de cada sección según la tensión recibida de la batería conocemos el estado de carga de la misma. A continuación, se detallan los distintos niveles de iluminación para cada tensión:

-Tensión batería menor de 10,8: Todo apagado.

-Tensión batería de 10,8 a 11,25: 1 bloque iluminado.

-Tensión batería de11,25 a 11,7: 2 bloques iluminados.

-Tensión batería de 11,7 a 12,15: 3 bloques iluminados.

-Tensión batería >12,15: Todos bloques iluminados.

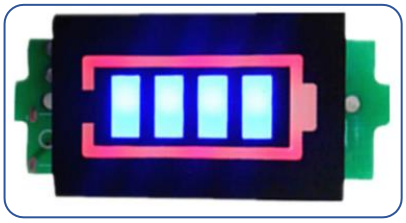


Figura 6. Indicador de carga, modelo xw228dkfr4 de la marca Xtori.

Se ha seleccionado este módulo puesto que es específico para mostrar el estado de carga de baterías de litio 18650 3S, como es el caso, tiene un muy bajo coste y es muy compacto, ocupando poco espacio dentro de la caja del equipo. Sus dimensiones son 4x2x0.8 cm (Largo x Ancho x Alto).

1.1.1.6. Interruptor 3 pines

Para el encendido y apagado tanto de la alimentación del equipo como del indicador de nivel de carga se han montado dos interruptores. Los interruptores seleccionados son de tres pines, tipo SPST, redondo basculante y con led indicador de paso de corriente, Figura 7. Además, están preparados para su montaje directo en el chasis del equipo. Se ha seleccionado este tipo de interruptores por su facilidad de montaje, bajo coste y porque lleva indicador de paso de corriente de muy bajo consumo. A continuación, en la Tabla 6 se pueden observar sus características principales

Figura 7. Interruptor de 3 pines.

Tabla 6. Características interruptor de tres pines instalado

Tipo	SPST, Basculante, 3Pines (LED)
Corriente	Continua
Intensidad corriente	20A
Tensión	12V
Dimensiones (Largo x Ancho x Alto)	2.4 x2.4x 3 (cm)

1.1.2. Acondicionamiento alimentación del equipo

Es necesario llevar a cabo un acondicionamiento de la señal eléctrica de alimentación a fin de que esta sea adecuada al equipo, además de protegerlo. Esto es debido a que la posible aparición de cortocircuitos puede dañar los componentes del dispositivo. Por otra parte, es necesario conseguir una corriente eléctrica optima, sin rizados y estable a lo largo del tiempo con la que obtener un buen funcionamiento del microprocesador, ya que grandes variaciones en la tensión de alimentación pueden producir fluctuaciones en las lecturas de los sensores con lo que no se obtendrían medidas estables, fiables ni precisas. Además, una alimentación con tensión excesiva o escasa puede provocar problemas de sobrecalentamiento o mal funcionamiento.

Para llevar a cabo el acondicionamiento de la energía eléctrica de alimentación del equipo se montarán en una placa los siguientes elementos.

1.1.2.1. Fusible

Para proteger los elementos del circuito de posibles daños en caso de que se produzca un cortocircuito y evitar que se produzcan situaciones peligrosas en la red de alimentación se instala un fusible rápido de 1A, Figura 8, ya que esa es la intensidad de corriente máxima admitida por el microcontrolador. Este elemento se instala en un portafusible por lo que es fácil de cambiar en caso de rotura. Cuando la intensidad de corriente es superior a la soportada por el fusible (1A en este caso) este se romperá impidiendo así que circule la corriente eléctrica por el circuito y protegiendo con ello tanto a los elementos de alimentación como al microcontrolador frente a cortocircuitos.

Figura 8. Fusible seleccionado

1.1.2.2. Regulador de tensión

La función de los reguladores de tensión es proporcionar una salida de tensión continua constante a un valor prefijado e independiente de la tensión que alimenta el regulador. Hay que tener en cuenta que los reguladores pueden ser fijos, variables y de tensión positiva o negativa (respecto a tierra). Otro factor importante es que pueden alimentarse con un amplio margen de tensiones, pero siempre deben ser superiores a la tensión de salida que se pretende obtener (siempre consumen algo de tensión). En este proyecto se emplea un regulador de tensión fija y positiva LM7809 con salida de 9 V, Figura 9, ya que esta es la tensión idónea para alimentar el procesador. Con este regulador se consigue evitar sobretensiones y que el microcontrolador sea alimentado de forma constante a la tensión optima.

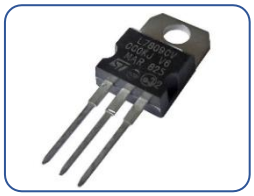


Figura 9. Regulador de tensión LM7809

1.1.2.3. Condensadores

A pesar de que el regulador de tensión estabiliza la tensión de alimentación y consigue que esta sea constante existe tensión de rizado que este no es capaz de estabilizar. Para atenuar esta tensión de rizado que produce fluctuaciones en la tensión de alimentación a lo largo del tiempo es necesario instalar otro componente denominado condensador. Con él se consigue que la tensión de alimentación no tenga pequeñas variaciones debidas a la tensión de rizado con lo que se obtiene una tensión más estable. Para ello se instalan dos condensadores, uno a la entrada y otro a la salida del regulador de tensión. Los condensadores seleccionados son un condensador de 0,33 µF, Figura 10, instalado a la entrada del regulador de tensión y otro de 0,1µF ubicado a la salida del mismo.

Figura 10. Condensador de 0.33 µF

1.2. SISTEMA DE ADQUISICIÓN Y ALMACENAMIENTO DE DATOS

A continuación, se describen los elementos de hardware necesarios para la adquisición de datos de los sensores y para su posterior almacenamiento.

1.2.1. Placa microcontrolador keyestudio mega 2560 r3

Un microcontrolador es un circuito integrado de pequeñas dimensiones, programable, capaz de ejecutar las órdenes grabadas en su memoria y capaz de interactuar de forma bidireccional con elementos electrónicos externos, es decir, que permite tanto que sea ingresada información desde un sistema externo, como emitir información a partir de ese sistema. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica, incluyendo en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida. Estas características hacen de los microcontroladores el sistema idóneo para gestionar el equipo ya que tienen todo lo necesario para controlar todos los sensores y módulos y con ellos realizar la toma de datos y su posterior procesado. Por tanto, para la gestión del equipo se empleará un microcontrolador, siendo este el encargado en el equipo de gestionar todos los sensores y módulos necesarios para la toma de datos y su posterior almacenamiento tanto en una tarjeta SD como en la nube.

Para ello se ha seleccionado la placa Keyestudio Mega 2560 R3, Figura 11, la cual integra un microcontrolador ATmega2560 y todos los componentes necesarios para su funcionamiento, pudiéndose conectar periféricos a sus entradas y salidas, añadir shields con facilidad, así como se puede programar fácilmente con un ordenador a través de la comunicación serial. Se ha seleccionado esta placa debido a las características que tiene por ser de una plataforma de tipo Arduino, véase el apartado 1.Introducción de la memoria, lo que la hace capaz de conectar periféricos a las entradas y salidas de su microcontrolador fácilmente y con ello siendo capaz de leer entradas tanto analógicas como digitales y convertirlas en una salida de forma sencilla, pudiéndose programar el microcontrolador de la tarjeta fácilmente, disponiendo de gran cantidad de módulos capaces de ampliar sus capacidades y funciones y disponiendo de gran cantidad de documentación para su uso, todo ello con un bajo coste. Además, la placa seleccionada dispone de gran cantidad de pines de comunicación, así como un coste inferior al

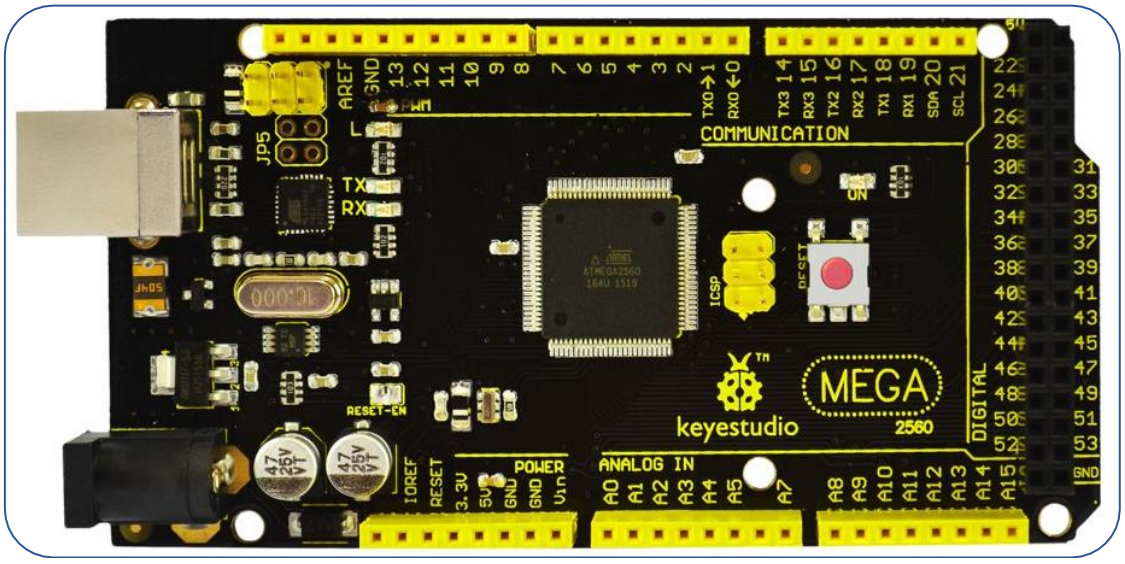


Figura 11. Placa microcontrolador Keyestudio Mega 2560 R3

de la plataforma Arduino oficial.

A continuación, se detallan las características del microcontrolador Keyestudio Mega 2560 R3.

El microcontrolador dispone de 16 canales analógicos preparados únicamente para recibir entradas de señales, así como 54 canales digitales de entrada y salida. Sin embargo, hay que destacar que algunos de estos pines tienen un uso reservado para otros fines, como es el caso de los pines digitales 0 y 1, destinados a la comunicación serie mediante cable USB. Si fuera necesario establecer una comunicación serie con más de un dispositivo, existen otras parejas de pines que pueden cumplir con este fin (los pines 18-19 forman el Serial1, el 16-17 el Serial2, y el 14-15 son el Serial3). También cuenta con 4 UARTs (puertos serial por hardware), cristal oscilador de 16 Mhz, conexión USB, jack de alimentación, conector ICSP, bus I2C y botón de reset. El procesador dispone de un conversor analógico digital de 10 bit que le confiere una precisión de 4,88mV a los pines de entrada, lo que supone una precisión relativa respecto a la señal de entrada de 0,1% (1/1024).

Por otro lado, los pines digitales del 2 al 13, pueden ser usados como salidas de PWM, es decir ofrecer como salida un valor de tensión entre 0 y 5 V.

En cuanto a la manera de alimentar eléctricamente a la placa, existen 3 métodos para poder hacerlo. El primero es la conexión a través del cable USB directamente al ordenador. El segundo es mediante la conexión de una fuente de alimentación externa al conector tipo Jack de alimentación. El último método es a través de un cable que esté a 5V conectado en el pin V_{in}.

En este proyecto la alimentación se lleva a cabo es a través del Jack de alimentación. Para su uso debemos tener en cuenta que este voltaje no supere los 20V ni sea inferior a 6V. El fabricante recomienda valores entre 7 -12 V, siendo el óptimo 9 V.

Además, el microcontrolador dispone de dos pines de puesta a tierra, etiquetados como GND, además de otros dos pines de 5V y 3.3V, que pueden emplearse, como se hace en este proyecto, para alimentar circuitos exteriores.

En cuanto al botón de reset de la placa, cuando este es pulsado, el programa que tenga cargado el microcontrolador en su interior vuelve a comenzar desde la primera línea de código.

El microcontrolador, además, tiene tres leds indicadores. Los dos inferiores (TX y RX) corresponden a la información digital que viaja a través del puerto serie formado por los pines 0 y 1. El led superior (marcado como L) es un led de prueba que conecta con el pin digital 13, y puede usarse tanto para ver el valor que toma la entrada conectada a él, como para verificar que la salida ordenada por el programa cargado es realmente la deseada.

Cuando se decide emplear los pines digitales como salidas o entradas debe tenerse en cuenta que la corriente continua que circula por estos pines es de unos 40 mA, lo cual limita las aplicaciones a baja potencia y sería necesaria una etapa de amplificación si la aplicación requiriera potencias e intensidades mayores.

Por último, Keyestudio MEGA 2560 tiene 256 KB de memoria flash en la que se guardan los programas cargados. También incluye una memoria SRAM de 8KB y una EEPROM de 4KB, la cual puede ser escrita y leída mediante funciones de la librería EEPROM.

1.2.2. Módulo GSM/GPRS sim900

Los datos obtenidos de los sensores por el microcontrolador han de ser almacenados para su posterior estudio. Para un rápido y fácil acceso a ellos se ha optado por su almacenamiento en la nube. Con ello se consigue acceder en cualquier momento de forma sencilla, remota y actualizada a todos los datos obtenidos sin necesidad de desplazamiento hasta el lugar donde

este instalado el equipo a través de internet. Además, al almacenarse en la nube los datos quedan guardados de forma segura, por lo que siempre existe una copia de seguridad disponible de los mismos, evitando el problema que surge con los métodos físicos de almacenaje, que se pueden estropear y con ello perder la información. Como desventajas este método de almacenaje tiene la necesidad de conexión a internet y la dependencia del proveedor, si por algún motivo la conexión a internet o el proveedor fallan los datos se perderán. Para evitar estos problemas el equipo complementa este sistema de almacenamiento de la información con otro de tipo físico, para que en caso de fallo los datos queden almacenados y no se pierdan.

Para llevar a cabo este sistema de almacenamiento el procesador necesita de un módulo que le de acceso a internet, ya que este no es capaz por si solo de hacerlo. El módulo encargado de ello es el módulo GSM/GPRS SIM900. Este módulo está basado en el chip cuatribanda SIM900 de SIMCOM que le confiere al microcontrolador la función GSM/GPRS. Esto hace posible que gracias a este módulo y utilizando una tarjeta SIM el microcontrolador se conecte a la red de internet y telefonía móvil y envíe y reciba datos por GSM/GPRS, con lo que podrá enviar y almacenar los datos en la nube mediante protocolo TCP/UDP. También le hace capaz de recibir y enviar tanto llamas como SMS y MMS, por lo que se podrán enviar avisos desde el equipo si surge cualquier problema al número de teléfono que se desee.

Concretamente, el módulo seleccionado es el GSM/GPRS SIM900 ks0142 de Keyestudio, Figura 12, por su bajo coste y porque es compatible con el procesador empleado a la par que dispone de una fácil conexión y comunicación al mismo al ser del mismo fabricante ambos componentes. El chip cuatribanda SIM900 que incorpora le permite hacer conexiones en bandas de 850/ 900/ 1800/ 1900 MHz. El procesador se comunica con el módulo a través del puerto serie mediante comandos AT y en la placa del módulo hay un interruptor para seleccionar el tipo de comunicación del puerto serie, haciendo que esta se controle bien a través de software o bien por medio del hardware. Además del chip SIM900 la placa dispone de otros de los elementos como son la ranura de conexión de tarjeta SIM; pines de interfaz PWM y ADC con el procesador; botones de control de encendido y apagado de la placa, el chip y conexión a internet con leds indicadores del estado de cada uno de ellos; conexión para antena y antena; reloj RTC y conexiones para micrófono y auriculares. En la Tabla 7 se muestran las características específicas del módulo y en la
Tabla 8 sus conexiones con el procesador.

Figura 12. Módulo GSM/GPRS SIM900 ks0142 de Keyestudio.

Tabla 7. Características módulo GSM/GPRS SIM900 ks0142 de Keyestudio.

Alimentación(V)	5V (autoadaptación a 3,3V) con procesador a 9V	
Bandas de conexión	Quad-Band GSM/GPRS 850/900/1800/1900MHz	
Protocolo conexión	TCP/UDP incorporado	
Interfaz de conexión	PWM y ADC	
Comunicación con procesador	Puerto serie por software y hardware	
Temperatura de funcionamiento	-40°C a +85 °C	
Consumo	Modo sleep	1,5 mA
	Llamada	250 mA
	Modo datos GPRS	440 mA
Dimensiones	85x67(mm)	

Tabla 8. Conexiones del procesador con el módulo GSM/GPRS SIM900 ks0142 de Keyestudio

PROCESADOR	MODULO GSM/GPRS SIM900
Gnd	Gnd
5V	Vcc
RX, Pin 10	TX
TX, Pin 11	RX

1.2.3. Módulo microSD

Los módulos Micro SD permiten que el microcontrolador almacene información en una tarjeta micro SD de forma sencilla. Se ha seleccionado este método de almacenamiento de datos físico para complementar el almacenamiento en la nube puesto que se ha convertido en un estándar, desplazando a otros medios de almacenamiento de datos debido a su gran capacidad y pequeño tamaño. También, tiene un muy bajo coste y su programación y conexión al microcontrolador son sencillos. Este sistema tiene la ventaja de proporcionar una amplia capacidad de memoria que además es no volátil (es decir, resiste cuando se elimina la alimentación), y puede ser extraída y conectada a un ordenador con facilidad, hecho de gran importancia en caso de que falle el almacenamiento en la nube o falle la alimentación del equipo. Como desventaja mencionar que este sistema supone una gran carga de trabajo para el procesador.

El módulo micro SD seleccionado, Figura 13, es comercializado por la empresa Catalex y tiene las características a continuación detalladas.

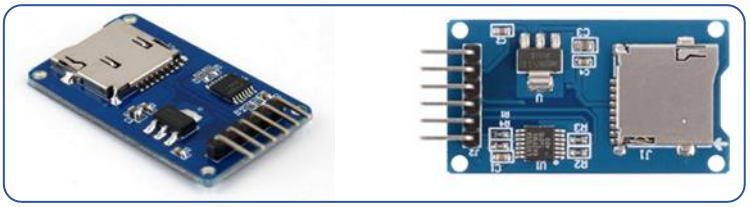


Figura 13. Módulo micro SD

La comunicación entre el microcontrolador y el módulo se lleva a cabo a través de bus SPI. Aunque se puede realizar a través de otros interfaces, como bus I2C o UART, siendo normalmente preferible emplear SPI por su alta tasa de transferencia.

Respecto a las tarjetas que se pueden emplear están las tarjetas SD o SDSC (Standard Capacity) o SDHC (High Capacity), pero no las SDXC (Extended Capacity). La tarjeta deberá estar formateada en sistema de archivos FAT16 o FAT32.

En cuanto a su alimentación es necesaria una tensión de 3.3V, aunque este módulo incorpora la electrónica necesaria para conectarlo de forma sencilla a la placa del equipo a través del pin de 5 V al incluir un regulador de voltaje.

La conexión del módulo a la placa del procesador es sencilla, simplemente se alimenta el módulo desde la placa del microcontrolador mediante 5V y Gnd. Por otro lado, conectamos los pines del bus SPI a los correspondientes del microcontrolador, véase Tabla 9.

Tabla 9. Conexiones del procesador con el módulo micro SD

PROCESADOR	MODULO MICRO SD
Gnd	Gnd
5V	Vcc
MISO, Pin50	MISO
MOSI, Pin 51	MOSI
SCK, Pin 52	SCK
CS, Pin 53	CS

1.2.4. Módulo DS3231

Puesto que la placa del microcontrolador no dispone de un reloj interno preciso que mantenga el valor del tiempo en caso de pérdida de alimentación con el que conocer la hora y la fecha exactas en que se realiza cada toma de datos de los sensores es necesario instalar un módulo que le proporcione esa información. Para ello se instalará el módulo DS3231 fabricado por Maxim el cual incorpora un reloj RTC y medición de temperatura, Figura 14.

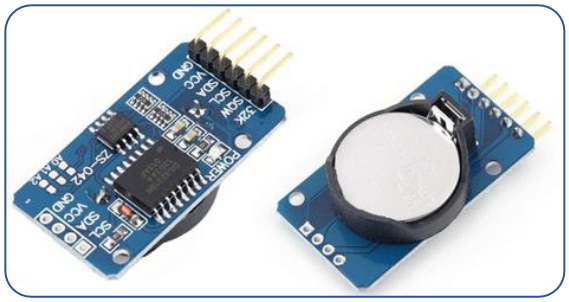


Figura 14. Módulo DS3231

Se ha seleccionado este módulo dada la sencillez de instalación y bajo coste que ofrece. También porque con él se consigue reducir el consumo de energía, gran precisión en la medida del tiempo y se libera al procesador de tener que realizar la contabilización del tiempo. Además, incorpora una batería CR2032 para mantener el dispositivo en hora al retirar la alimentación.

El reloj RTC que incorpora el módulo permite obtener mediciones de tiempo en las unidades temporales que empleamos de forma cotidiana teniendo en cuenta las peculiaridades de medir el tiempo, como por ejemplo el sistema sexagesimal, los meses con diferentes días, o los años bisiestos.

Dado que las variaciones de temperatura afectan a la medición del tiempo del reloj produciendo desfases temporales que pueden llegar a ser de 1 a 2 min al día este módulo incorpora medición y compensación de la temperatura garantizando una precisión de al menos 2ppm, lo que equivale a un desfase máximo 172ms/día o un segundo cada 6 días, aunque a efectos prácticos

normalmente se consiguen precisiones superiores, equivalentes a desfases de 1-2 segundos al mes.

En este caso también se aprovechará el hecho de que el módulo realice medición de temperatura para conocer la temperatura interna del equipo y con ello saber si está dentro de los márgenes de funcionamiento de sus componentes.

La comunicación entre el microcontrolador y el módulo se realiza a través del bus I2C, por lo que es sencillo obtener los datos medidos, véase Tabla 10. Y se alimenta mediante los pines 5V y Gnd de la placa del procesador.

Tabla 10. Conexiones del procesador con el módulo DS3231.

PROCESADOR	MODULO MICRO SD
Gnd	Gnd
5V	Vcc
SCL, Pin 21	SCL
SDA, Pin 20	SDA

2. COMPONENTES COMPLEMENTARIOS, IMPLEMENTACIÓN Y PUESTA A PRUEBA

A continuación, se detallan los elementos de hardware necesarios para llevar a cabo la implementación y puesta a prueba del equipo desarrollado mediante el uso de un dendrómetro basado en un sensor L.V.D.T.

2.1. SENSOR DE PUESTA A PRUEBA, DENDRÓMETRO

El dendrómetro es un sensor de alta precisión, normalmente basado en sensores de desplazamiento L.V.D.T (de las siglas en inglés Linear Variable Differential Transformer) que suele usarse, en el campo agronómico, para medir las variaciones en el crecimiento del tronco de los árboles o de los frutos. Se han definido varios parámetros que se determinan a partir de las medidas en continuo de este crecimiento que se relacionan con el estado hídrico de la planta. Por ello pueden ser utilizados, como indicador del estado hídrico de la planta, para la programación del riego (GONZALEZ-ALTOZANO, 1998).

2.1.1. Principio funcionamiento (RS COMPONENTS LTD., 2019)

Un sensor de desplazamiento lineal o transductor de desplazamiento inductivo LVDT se construye utilizando un transformador estático (devanado primario) y dos devanados secundarios, véase Figura 15. Los devanados conforman una bobina hueca a través de la cual puede desplazarse un núcleo magnético. El núcleo magnético proporciona un camino por el que unir las bobinas a través de un flujo magnético. Cuando el devanado primario está conectado a un suministro de corriente alterna se genera un flujo magnético que a través del núcleo magnético llega a los devanados secundarios. Este flujo magnético, a su vez, genera un flujo de corriente proporcional al mismo en las bobinas secundarias.

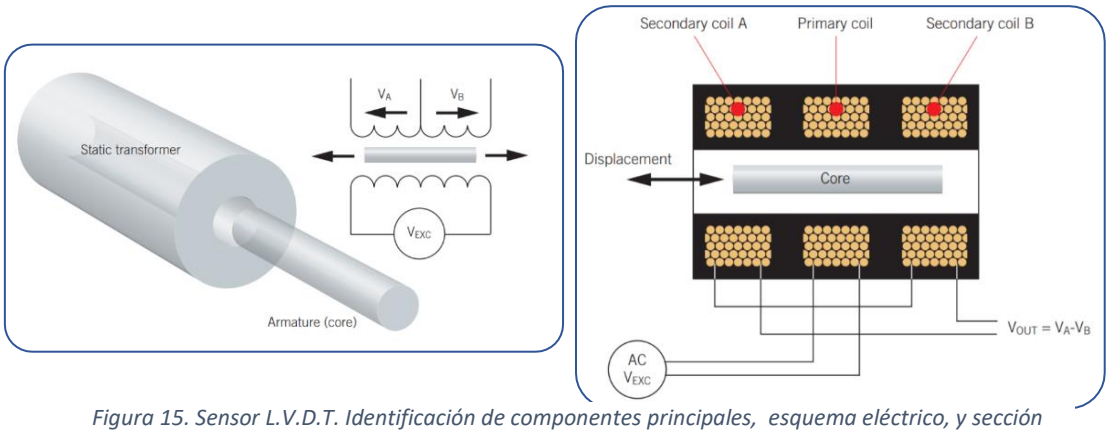


Figura 15. Sensor L.V.D.T. Identificación de componentes principales, esquema eléctrico, y sección transversal con la identificación de sus partes

Las bobinas secundarias A y B están conectadas en oposición en serie, de modo que los dos voltajes V_A y V_B tienen una fase opuesta y la salida del transductor es V_A - V_B. Si el núcleo está en la posición central, se inducirán tensiones de igual magnitud, pero fase opuesta en cada bobina secundaria y la salida neta es cero. A medida que el núcleo se mueve en una dirección, la tensión en la bobina secundaria correspondiente aumenta mientras que la otra bobina experimenta una reducción de tensión complementaria. El efecto neto es una salida de voltaje del transductor que es proporcional al desplazamiento, véase Figura 16. El conocimiento de la magnitud y la fase de la salida con respecto a la señal de excitación permite deducir la posición y dirección del movimiento del núcleo desde la posición nula.

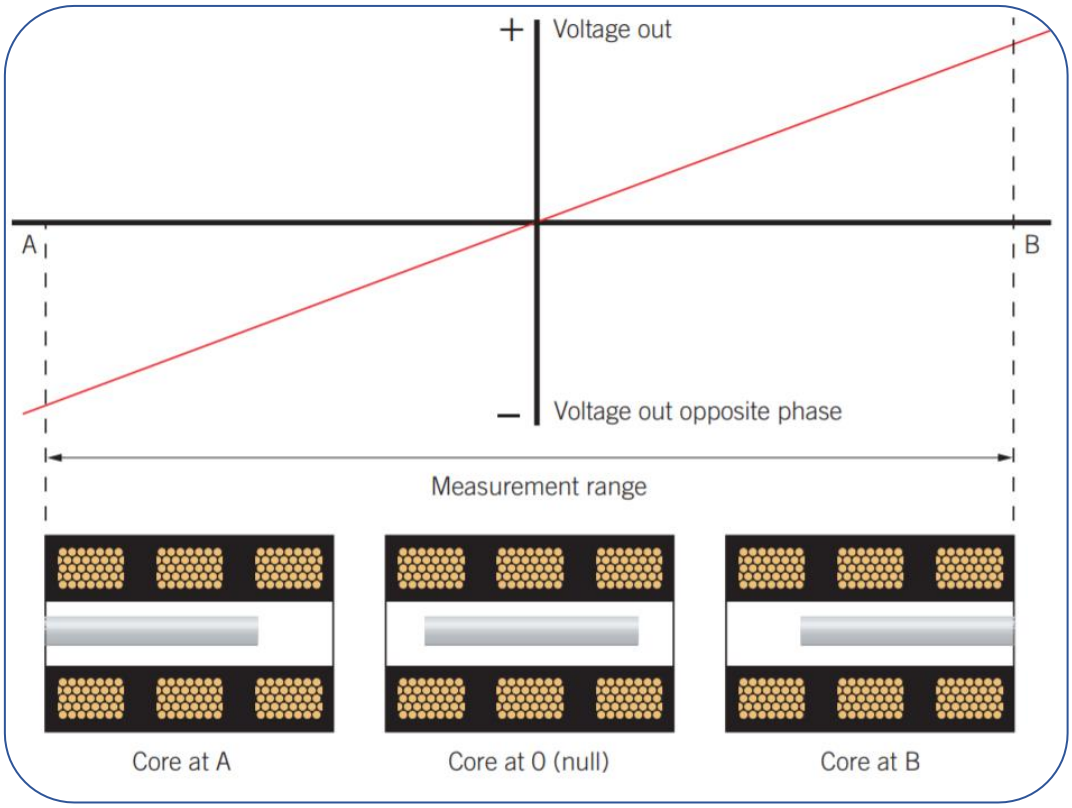


Figura 16. Funcionamiento de la señal del sensor L.V.D.T.

La salida de un LVDT es una función lineal de desplazamiento en su rango de medición calibrado. Más allá de este rango, la salida se vuelve cada vez más no lineal. El rango de medición se define como ± distancia desde la posición nula del transductor.

2.1.2. Dendrómetro seleccionado

El dendrómetro empleado en la fase de implantación está basado en el sensor de desplazamiento lineal L.V.D.T., Solartron Metrology, modelo DFg ± 2,5 mm, Bognor Regis, UK, precisión ± 10 µm, Figura 17, instalado en portasensor construido con material invar (aleación de Ni y Fe) y aluminio (coeficiente de dilatación próximo a 0).

Figura 17. Sensor de desplazamiento lineal L.V.D.T., Solartron Metrology, modelo DFg ± 2,5mm.

Este modelo está dotado de oscilador, demodulador y filtro electrónicos integrados, que dan una señal d.c. de salida lineal y proporcional a la posición del núcleo. Además, presenta piezas independientes para la bobina y la electrónica, con un núcleo libre de fricción, ligero y con rodamiento acoplado de homopolímero de poliacetal, lo que permite al núcleo desplazarse de forma guiada por la parte interior de la pieza de la bobina sin apenas rozamiento. A continuación, se detallan en la Tabla 11 las características específicas del sensor seleccionado y en la Tabla 12 las conexiones del mismo con el procesador. Para mayor información sobre la conexión entre el sensor y el microcontrolador véase el Anejo II esquemas del sistema hardware.

Tabla 11. Características específicas del sensor de desplazamiento lineal L.V.D.T., Solartron Metrology, modelo DFg ± 2,5 mm

Recorrido DFg 2,5	±2,5mm
Sensibilidad	750 mV/mm a 10 Vd.c.
Consumo de corriente para activación de 10V	10-15mA
Tensión de entrada	10-24V d.c.
Rizado a la salida	<1% fondo de escala
Tiempo de respuesta	5ms
Respuesta en frecuencia	-3dB a 100Hz
Temperatura de funcionamiento	de -20°C a +80°C
Coeficiente de temperatura	<0,005%/°C
Sensibilidad	750 mV/mm a 10 Vdc
Error sensibilidad	<0,01%/°C
No linealidad	0,3%
Cable	3m, 5 núcleos y apantallado con PVC
Conexiones eléctricas	
Rojo	alimentación +ve
Azul	alimentación 0V
Blanco	salida +ve
Verde	salida 0V
Amarillo	N/C

Tabla 12. Conexiones del procesador con el sensor de desplazamiento lineal L.V.D.T.

PROCESADOR	SENSOR L.V.D.T.
Pin A0	salida +ve (Verde)
Pin A1	salida 0V (Amarillo)

2.1.3. Alimentación del sensor

El sensor necesita para su correcto funcionamiento ser alimentado con una tensión de entrada de entre 10 y 24 V en corriente continua.

Para poder alimentarlo sin que el equipo pierda independencia se ha optado por realizar su alimentación de igual modo que en el dispositivo. Por tanto, se alimentará con un sistema basado en la energía solar en el que se acumulará la energía obtenida en una batería que ira suministrando la energía al sensor. A continuación, se detallan los componentes necesarios para dicho sistema.

2.1.3.1. Placa solar

Para dicho sistema de alimentación se empleará el mismo modulo solar instalado para el equipo, concretamente el panel de silicio policristalino SHS modelo ESPMC050 fabricado por Era Solar, véase apartado 1.1.1.1 Panel solar 12V. En consecuencia, tanto el sensor como el dispositivo de monitorización compartirán el mismo módulo solar, ya que posee potencia suficiente para cargar las baterías de ambos componentes. Esto es debido a que, aunque la carga coincida en el tiempo, la placa solar les ofrecerá 2,7A (1,35A cada una), corriente suficiente para cargar correctamente las baterías, aunque sea a menor velocidad. Esta disminución de la velocidad de carga no supondrá ningún problema, puesto que en tal caso en menos de 2,5 horas la batería del sensor estará totalmente cargada, disponiendo de horas de luz más que suficientes para cargar la batería del equipo por completo, puesto que, necesita menos de 4,5 horas a 2,7A para ello. Este hecho sumado a que las baterías se recargan antes de descargarse por completo, disminuyendo los tiempos de carga, hace que el momento de carga de ambas baterías a penas coincida en el tiempo por lo que con un solo panel solar se cargaran en condiciones óptimas y de forma rápida e independiente ambas baterías en la mayoría de las ocasiones. Además, se minimizarán aún más las posibilidades de carga simultanea de ambas baterías para evitar posibles problemas de carga lenta haciendo uso únicamente del sensor en los momentos de medida ya que así se ahorrará energía y con ello se disminuirá la cantidad de cargas necesarias.

2.1.3.2. Batería

De igual modo que sucedía con el equipo, es necesaria la instalación de un acumulador de energía puesto que la radiación solar de la que se obtiene la energía es variable en el tiempo. Para solucionar los problemas que este hecho acarrea, ya comentados anteriormente, se instalará una batería de litio. Se ha seleccionado este tipo de acumulador por los motivos ya expuestos en el apartado 1.1.1.2 Batería litio. La batería seleccionada para la alimentación del sensor tiene características similares a la batería del equipo a excepción de la capacidad que es bastante menor, puesto que el sensor requiere de menor cantidad de energía para su funcionamiento. Concretamente se ha seleccionado la batería Li-Ion modelo Dii-12V3000 de la marca Liitokala. A continuación, se detallan las características de la misma en la Tabla 13.

Tabla 13. Características principales de la batería de Li-Ion modelo Dii-12V3000 de la marca Liitokala.

Parámetros Nominales	Tipo	Li-Ion, 3S1P, celdas 18650
	Voltaje	12V
	Capacidad	3Ah
	Capacidad	36Wh
Dimensiones (Largo x Ancho x Alto)		67x55x19(mm)
Peso		0,15Kg
Corriente de descarga estándar		0,2C (0.6A)
Corriente de descarga máxima		1C (3A)
Resistencia interna		250mohm
Protección sobrecorriente, sobrecarga, sobredescarga y cortocircuito		SI
Voltaje de carga		12.6V
Corriente de carga		1A
Corriente de carga máxima		3A
Temperatura funcionamiento		0-60°C

Se ha seleccionado este modelo de batería de litio por tener características similares a la batería del equipo. Esto simplifica tanto el sistema de carga como el diseño del mismo puesto que de esta forma se aprovecha la placa solar del equipo y no es necesario rediseñar un nuevo sistema de alimentación, ahorrando así tiempo y dinero. Además, se ha elegido por su bajo coste, pequeño tamaño y capacidad de almacenaje. Su capacidad de almacenaje de 3000mAh permite alimentar el sensor unas 20 horas seguidas sin energía solar, así como una rápida carga, ya que se puede cargar por completo en poco más de 1 hora con la placa solar instalada en pleno funcionamiento. Este hecho hace que la batería seleccionada tenga una capacidad de acumulación de energía adecuada.

2.1.3.3. Controlador de carga

El controlador de carga, o regulador de carga, es el componente de la instalación encargado de dirigir y controlar la energía que circula entre los módulos solares fotovoltaicos y la batería, encargándose de evitar las sobrecargas y sobredescargas de las baterías cuando están recibiendo energía fotovoltaica del módulo solar de la instalación. Dado que las baterías instaladas tienen procesos de carga y descarga delicados, como se ha explicado en el apartado 1.1.1.2 Batería litio, este componente es de gran importancia en la instalación.

Puesto que la batería seleccionada para los sensores son del mismo tipo que la del equipo, y con similares características de carga , ambos comparten la placa solar y el regulador de carga seleccionado es el mismo que se ha instalado en el dispositivo de monitorización. Véase apartado 1.1.1.3 Módulo 18650 controlador de carga mppt, para consultar la justificación puesto que ésta se cumple para las características de la batería del sensor. Concretamente, el controlador elegido es el controlador MPPT de carga solar BQ24650 de 12V para carga de baterías de litio 3s de celdas tipo 18650, del vendedor Walfront. Las características específicas se encuentran en la Tabla 3 del apartado 1.1.1.3 Módulo 18650 controlador de carga mppt .

2.1.4. Control de funcionamiento

Dado que la monitorización del sensor LVDT se realiza periódicamente en el testeo se producirán lapsos de tiempo entre toma de datos que consumirán energía en vano si este se encuentra permanentemente conectado. Ello supone un consumo excesivo de energía, sobre todo si se tiene en cuenta que de cada diez minutos tan solo se estarán tomando datos en uno.

Este hecho supone un problema en primer lugar porque el consumo de energía llevado a cabo por el sensor si este está continuamente conectado (10 min.) es diez veces mayor que el consumo del sensor si solo se conecta en el momento de la monitorización (1 min.). Por tanto, este hecho conlleva un gran desaprovechamiento de la energía acumulada en la batería.

En segundo lugar, y como consecuencia del primer problema planteado, el mayor consumo de energía de forma inútil por parte del sensor en cada monitorización de datos conlleva una mayor cantidad de ciclos de carga/descarga de la batería de forma innecesaria para un mismo número de tomas de datos por parte del equipo. Ello a su vez da lugar a dos problemas. Por una parte, produce una disminución de la vida útil de la batería, ya que el número de ciclos de carga/descarga que esta soporta es limitado. Por otra, al realizarse un mayor número de ciclos de carga/descarga, aumentara el tiempo en que se esté cargando la batería del sensor, aumentando así las probabilidades de que se carguen simultáneamente la batería del sensor y del equipo y por ello de que pueda surgir algún problema de carga en ellas.

Por tanto, el uso continuo del sensor supone un problema de desaprovechamiento de energía que, además, conlleva una disminución de la vida útil de la batería y aumento de la probabilidad de aparición de problemas por carga lenta en las mismas.

Para evitar este problema se ha optado por instalar un módulo relé controlado por el microprocesador. Dicho relé abrirá y cerrará el circuito de alimentación del sensor. Con ello se consigue controlar con el microprocesador los tiempos de funcionamiento del sensor. Así se puede conectar y desconectar el sensor solo en los momentos necesarios o deseados (cuando el equipo monitorice los datos del sensor), solucionando todos los problemas anteriormente mencionados. El módulo relé elegido para llevar a cabo dicho control es el modelo TE213 del fabricante XCSource, Figura 18. A continuación, se detallan sus principales características en la Tabla 14 y las conexiones del mismo con el procesador en la Tabla 15.

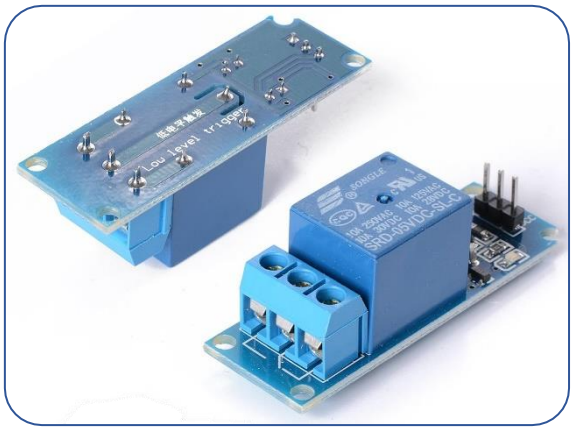


Figura 18. Modulo relé, modelo TE213.

Tabla 14. Características del módulo relé seleccionado modelo TE213 del fabricante XCSource.

Tensión alimentación cc	5V
Corriente estática accionador	4 mA
Corriente del accionador	2 mA
Tensión entrada	250Va.c.,30Vd.c.
Corriente entrada	10A
Terminal entrada	KF301
Dimensiones (Largo x Ancho x Alto)	53x18x18,5(mm)
Peso	81,6 g

Tabla 15. Conexiones del procesador con el módulo relé

PROCESADOR	MODULO RELÉ
Gnd	Gnd
5V	Vcc
Pin D13	Señal

Además, esta solución, aumenta en gran medida la autonomía del sensor puesto que así el sensor consume diez veces menos energía. Por tanto, se consigue pasar de una autonomía del sensor de unas 20 horas seguidas sin energía solar a cerca de unas 200 h haciendo uso de toda la capacidad de almacenaje de la batería instalada.

2.1.5. Acondicionamiento de alimentación

Al igual que sucede en el caso del equipo, en el sistema de alimentación del sensor es necesario llevar a cabo un acondicionamiento de la señal eléctrica a fin de que esta sea adecuada para el equipo, además de protegerlo por los motivos ya descritos anteriormente.

El acondicionamiento de la energía eléctrica de alimentación del sensor es muy similar al del equipo, montándose igualmente en una placa los mismos elementos, un fusible, un regulador de tensión y dos condensadores. Estos tres elementos se instalan con los mismos fines, de igual modo y por los mismos motivos que se montan en el equipo, véase apartado 1.1.2.Acondicionamiento alimentación del equipo.

Tanto el fusible como los condensadores tendrán las mismas características técnicas que en el caso de la alimentación del equipo dado que los motivos de su elección son iguales. En el caso del regulador de tensión se instalará un regulador similar. Dado que el sensor necesita una tensión de alimentación entre 10 y 24V no es posible instalar para este el mismo regulador que en el caso del equipo, puesto que este ofrece una tensión de salida insuficiente (9V). En este caso se ha seleccionado un regulador del mismo tipo pero que ofrece una tensión mayor, concretamente el regulador seleccionado es el regulador de tensión fija y positiva LM7812 con salida de 12 V. Se ha seleccionado este regulador puesto que es capaz de ofrecer tensión suficiente para un correcto funcionamiento del sensor a la vez que permite que el sistema de alimentación del sensor y el equipo sean parcialmente compartidos. Bien es cierto que se podrían haber seleccionado otros reguladores del mismo tipo que ofrecieran tensiones mayores dentro del rango de alimentación del sensor. Pero, estos no permiten que la alimentación del sensor y el equipo sean parcialmente compartidos y por ello privarían al sistema de las ventajas que ello ofrece. Esto es debido a que estos reguladores requieren tensiones muy superiores para su correcto funcionamiento, lo que implicaría la instalación de baterías que ofrecieran mayor voltaje y por ello se necesitaría un nuevo y diferente sistema de alimentación solo para el sensor.

Además, en la alimentación del sensor se ha instalado un convertidor de voltaje entre la batería y los componentes anteriormente descritos. Este se ha instalado con el fin de aumentar el voltaje de alimentación que ofrece la batería al regulador de tensión. Con ello se consigue que el regulador de tensión tenga una tensión de entrada suficientemente grande para ofrecer una tensión de salida para la alimentación del sensor estable de 12V. Es necesaria su instalación puesto que el regulador consume cierto voltaje y por ello necesita un voltaje de entrada superior al de salida, en este caso superior a 12V. La batería seleccionada ofrece una tensión de 12V, insuficiente para el regulador de tensión. Por ello se requiere de la instalación del convertidor de tensión el cual hace posible regular la tensión a 12 V con el regulador de tensión y batería seleccionados. Otra opción posible seria instalar una batería que ofreciera 24V, pero esta opción se ha descartado puesto que impediría que el sistema de alimentación del sensor y el equipo

sean parcialmente compartidos con los problemas que ello conllevaría, como se ha comentado anteriormente. El convertidor de tensión seleccionado es el modelo FS12D24S1 del fabricante Fulree, Figura 19. A continuación, se detallan sus principales características en la Tabla 16.

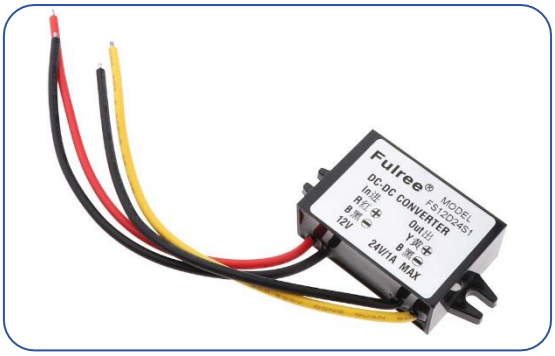


Figura 19. Convertidor de voltaje dc-dc, modelo FS12D24S1 del fabricante Fulree.

Tabla 16. Características del convertidor de voltaje dc-dc, modelo FS12D24S1 del fabricante Fulree.

Tensión de entrada	9-22Vd.c.
Tensión de salida	a 24Vd.c.
Corriente máxima de salida	1A
Potencia máxima de salida	24W
Eficiencia de conversión	> 95%
Protecciones	sobrecorriente, temperatura y cortocircuitos
Dimensiones(mm) (Largo x Ancho x Alto)	45x30x20

2.1.6. Acondicionamiento señal lvdt

Debido a la naturaleza del principio de funcionamiento del sensor LVDT la señal del mismo tiene tanto tensiones positivas como negativas, véase apartado 2.1.1.Principio funcionamiento. En este caso, para el sensor seleccionado y con la tensión de alimentación de 12V la señal de salida es de ±7,15 V en d.c. Estos hechos suponen un problema por dos motivos. El primero de ellos es que el microcontrolador no es capaz de leer tensiones negativas, además de que estas lo dañarían. El segundo problema es que la tensión máxima de entrada en el microcontrolador es de 5V, y superar dicho rango supondría nuevamente el problema de que no se leería la señal y esta dañaría el microcontrolador. Estos problemas hacen necesario que se realice un acondicionamiento de la señal para que esta pueda ser leída de la forma más adecuada, precisa y segura por el microcontrolador. Para ello se ha desarrollado un circuito de acondicionamiento de la señal. En dicho circuito se han empleado los siguientes elementos de hardware:

2.1.6.1. Resistencias

Las resistencias son empleadas en el circuito para crear divisores de tensión resistivos con los que se consigue disminuir la tensión de salida del sensor de 7,15V a 5V, solucionando así el problema de sobretensión. Para ello se han empleado resistencias de 39000 y 10000 Ω conectadas en serie (véase circuito eléctrico). Para su selección ha sido necesario calcular su capacidad de resistencia eléctrica aplicando la ley de ohm como se muestra a continuación:

Se tiene en cuenta que un divisor de tensión es una configuración de un circuito eléctrico que reparte la tensión de una fuente entre una o más impedancias conectadas en serie. En este caso, se instalan dos impedancias puramente resistivas (dos resistencias) conectadas en serie a la fuente del circuito de tensión $V_{cc,}$ que es igual a la tensión de salida del sensor de 7,15V. De ello resulta un divisor de tensión resistivo como el mostrado a continuación junto a su circuito equivalente en la Figura 20.

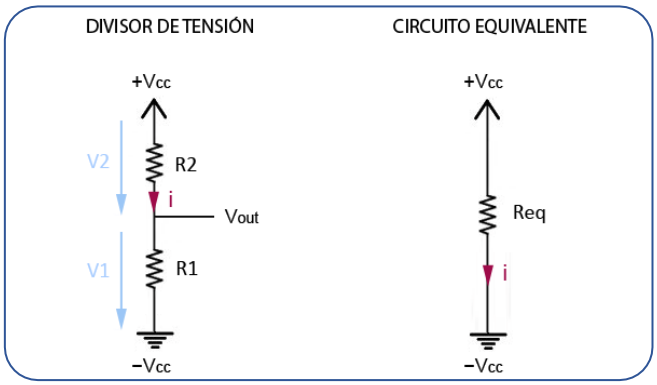


Figura 20. Esquema eléctrico divisor de tensión.

Comentar que la resistencia del circuito equivalente resulta ser igual al sumatorio de las resistencias del circuito del divisor de tensión.

$$R_{eq} = \sum R_n = R_1 + R_2$$

Para conocer el voltaje V_1 en la resistencia R_1, se utiliza la ley de Ohm:

$$V_1 = I \times R_1 \qquad\qquad I = \frac{V_{cc}}{R_{eq}}$$

Sustituyendo la segunda ecuación en la primera se obtiene que el voltaje V_1, en la resistencia R_1, será:

$$V_1 = \frac{R_1}{R_{eq}} \times V_{cc} = \frac{R_1}{R_1 + R_2} \times V_{cc}$$

Teniendo en cuenta que el voltaje V_1 es igual al voltaje de salida V_{out} y que la tensión de la fuente del circuito es igual a la tensión de salida del sensor de 7,15V se obtiene que la tensión de salida del circuito es la siguiente:

$$V_{out} = V_1 = \frac{R_1}{R_1 + R_2} \times V_{cc} = \frac{R_1}{R_1 + R_2} \times 7{,}15(V)$$

A partir de esta última ecuación se obtienen los valores de la capacidad de resistencia eléctrica de ambas resistencias. Para ello, se sustituirán R1 y R2 por valores comerciales existentes obteniendo así valores para Vout. Se probarán combinaciones de estos hasta obtener como resultado que V_{out} tiene una tensión inferior a 5 V, que es la máxima tensión de entrada en el microcontrolador, tratando de que esta se aproxime lo máximo posible a este valor para así aprovechar al máximo el rango de lectura del microcontrolador y con ello conseguir la máxima precisión de medida. Llevando a cabo dicho procedimiento se obtiene que R_1 y R_2 son iguales a 39000 Ω y 10000 Ω respectivamente obteniendo una tensión de salida de 5,69V, véase la ecuación resultante.

$$V_{out} = \frac{39000}{39000 + 10000} \times 7{,}15(V) = 5{,}69V$$

La tensión de salida del sensor resultante, V_{out}, sigue siendo superior al voltaje máximo de lectura del microcontrolador de 5V con el divisor de tensión calculado. A priori, dicha tensión de salida del sensor continuara siendo inadecuada y continuaran existiendo los problemas de sobretensión descritos anteriormente. Pero en realidad la tensión obtenida es adecuada debido a que hay que tener en cuenta que en el circuito de acondicionamiento de la señal del sensor también hay un diodo rectificador que consume 0,7V. Teniendo en cuenta todo lo comentado finalmente la tensión máxima de salida del circuito de acondicionamiento de la señal será igual a la de salida del divisor de tensión menos la tensión consumida por el diodo rectificador.

$V_{señal}= V_{div.\ tensión}-V_{diodo}=5,69-0.7=4,99V$

Por lo tanto, la tensión máxima de salida del circuito de acondicionamiento de la señal es de 4,99V, tensión adecuada y ajustada al valor máximo de lectura del microcontrolador, por lo que las resistencias elegidas son adecuadas. Añadir que dicho ajuste permite el uso completo del rango de lectura del microcontrolador con lo que se consigue una mayor precisión ya que con ello se aprovechan los 1024 bytes de lectura disponibles, hecho que no sería posible si la tensión de señal máxima fuese menor.

2.1.6.2. Diodo rectificador 1N407

El diodo rectificador 1N4007 es un rectificador de propósito general. Cuando el diodo es polarizado en directo, el ánodo a positivo y el cátodo a negativo, tiene una resistencia ideal igual a 0Ω. Por el contrario, cuando se polariza en inverso, tiene una corriente ideal infinita, no permitiendo el paso de corriente eléctrica. Por tanto, solo es capaz de transmitir corriente en una dirección. Debido a esta característica se ha empleado este componente para evitar que la corriente eléctrica tenga un voltaje negativo a la salida de la señal del sensor. Véase apartado 2.2.Esquemas específicos del sistema de acondicionamiento de señal del Anejo II para consultar la forma de instalación del mismo. Es importante destacar que este componente tiene una caída de tensión de aproximadamente 0,7V, hecho que se ha de tener en cuenta a la hora de calcular la reducción de la tensión de salida del sensor. A continuación, se muestra en la Tabla 17 las características principales del diodo seleccionado.

Tabla 17. Características del diodo rectificador 1N4007

Configuración de diodo	Simple
Corriente Continua Máxima Directa	1A
Tensión Repetitiva Inversa de Pico	500V
Tipo de Encapsulado	DO-41
Caída de tensión directa máxima	0,7V
Rango de temperatura	- 65 °C a +125 °C
Dimensiones(mm) (Largo x Ancho x Alto)	5.21 x 2.72 x 2.72mm
Corriente máxima de pico	30A
Tensión inversa de pico máximo	1KV
Tensión inversa máxima	50V

ANEJO II: ESQUEMAS DEL SISTEMA HARDWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ÍNDICE

ÍNDICE DE FIGURAS

En el presente anejo se muestran y explican los esquemas de montaje y la conexión de los componentes de los distintos sistemas de hardware desarrollados, así como de sus correspondientes esquemas eléctricos. Conjuntamente, se muestran los prototipos fabricados en base a estos esquemas con la identificación de cada una de sus partes.

Podemos diferenciar dos sistemas de hardware. Por una parte, el sistema hardware del dispositivo de monitorización diseñado en el que se centra el presente trabajo, el cual se encarga de la monitorización de sensores y el almacenamiento de los datos obtenidos de forma autónoma e independiente. Por otro, se encuentra el sistema hardware complementario al equipo principal de monitorización necesario para realizar el testeo e implementación del mismo. En él se integra el sensor empleado en el testeo e implementación junto a sus sistemas de alimentación y acondicionamiento de señal. Dado que se pueden diferenciar estos dos sistemas de hardware principales, sus esquemas se muestran por separado. De esta forma se muestran y explican los esquemas del equipo de monitorización principal por un lado y por otro los esquemas complementarios para el testeo e implementación.

1. ESQUEMAS EQUIPO MONITORIZACIÓN

En este apartado se muestran los esquemas correspondientes al equipo principal de monitorización desarrollado. En ellos destaca el microcontrolador encargado de la gestión del equipo y sus conexiones con los demás componentes, fundamentales para el correcto funcionamiento del equipo.. Destaca el módulo sim900 y el módulo SD cuyas conexiones son más complejas y/o delicadas. También destaca de la alimentación del mismo la conexión del regulador de carga, vital para conseguir una alimentación estable y de calidad, así como de los elementos de acondicionamiento de la misma, como son los condensadores y el regulador de tensión, los cuales se han de conectar del modo indicado para conseguir la mejor calidad de alimentación de gran importancia para el correcto funcionamiento del equipo.

En la Figura 1se muestra el esquema simplificado de montaje y conexiones de componentes. A continuación se muestra el esquema eléctrico del equipo, Figura 2. Por último, se puede ver el prototipo fabricado con la identificación de todos sus componentes, Figura 3.

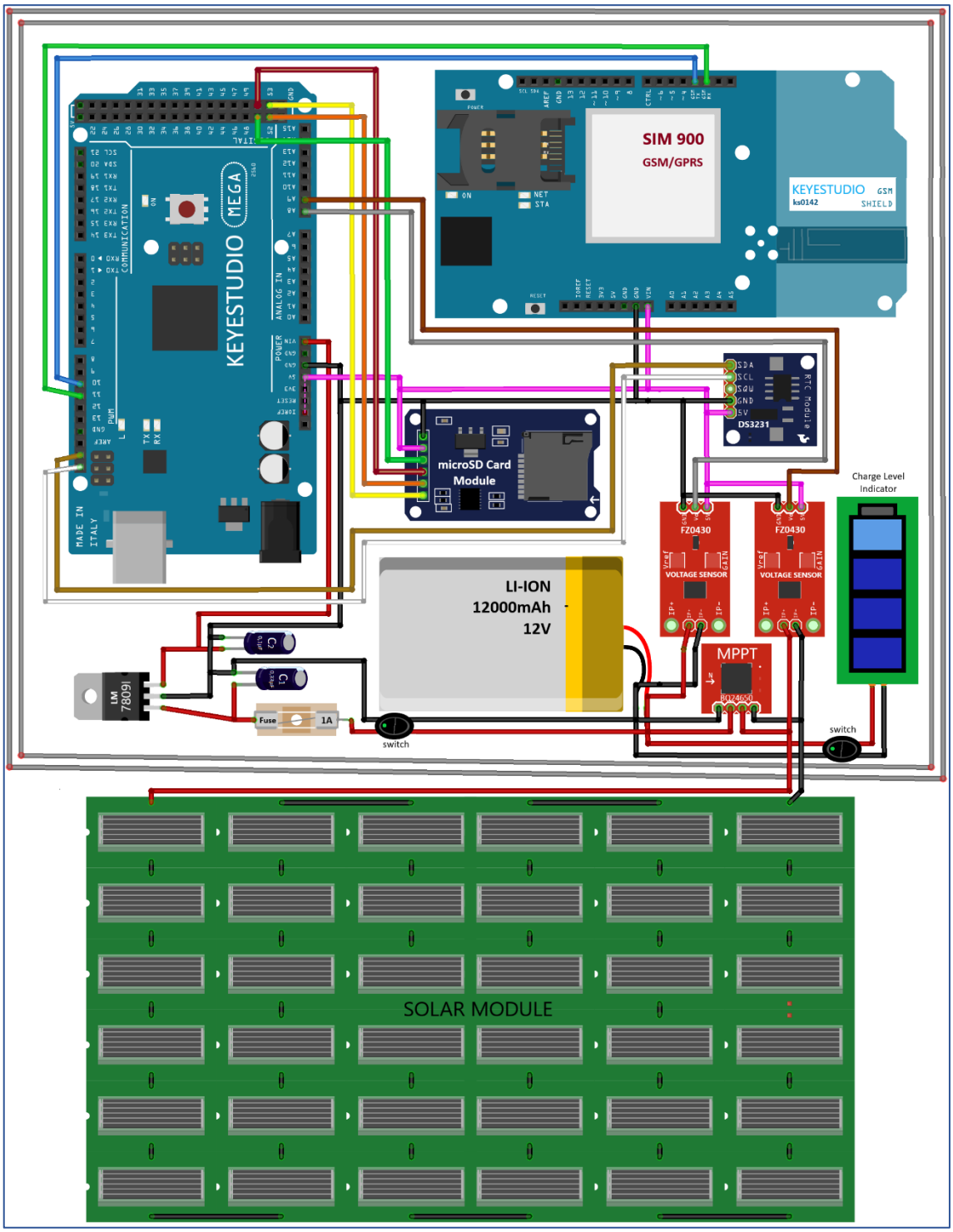


Figura 1. Esquema simplificado de montaje y conexiones de componentes del dispositivo de monitorización.

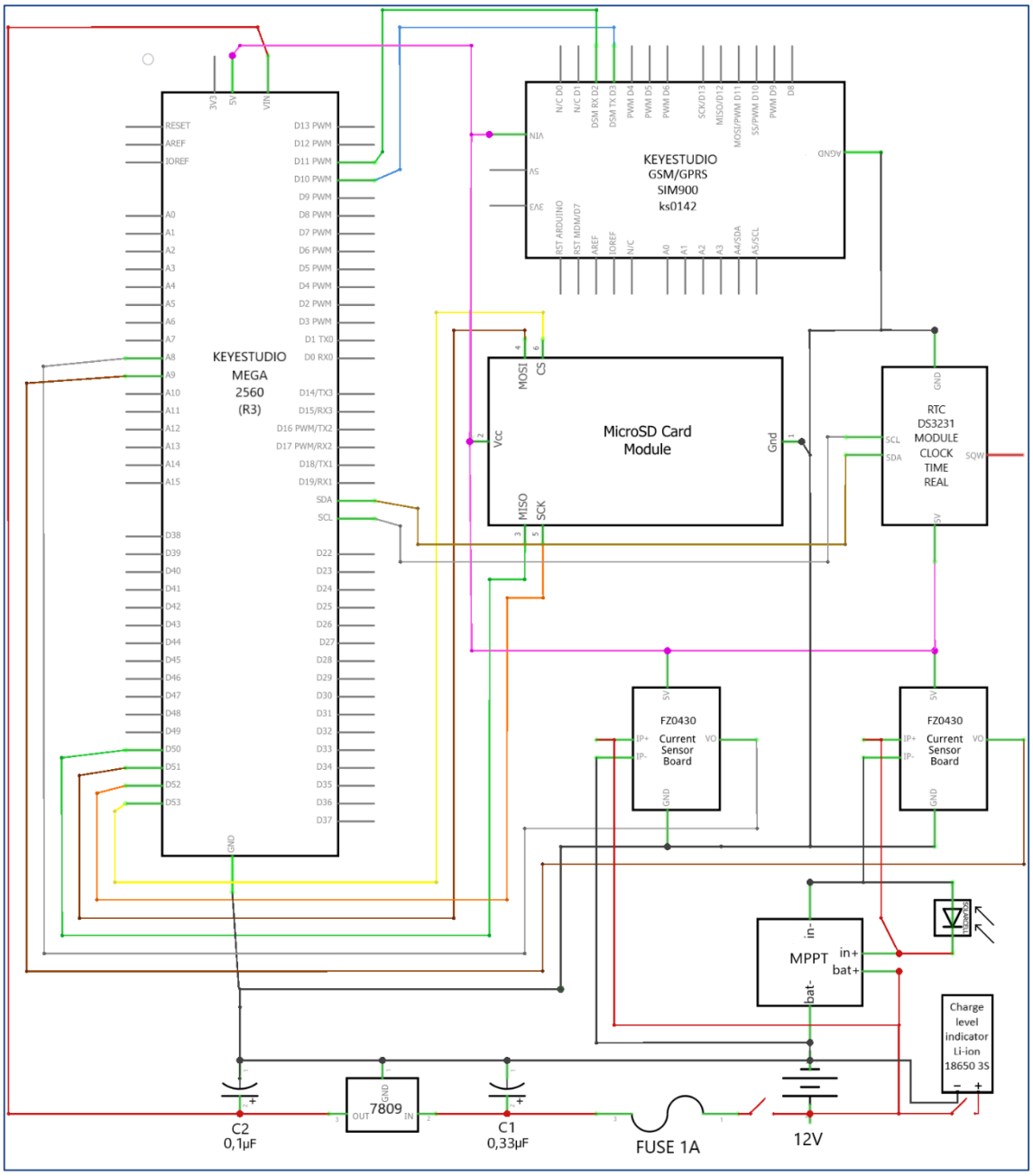


Figura 2. Esquema eléctrico del dispositivo de monitorización.

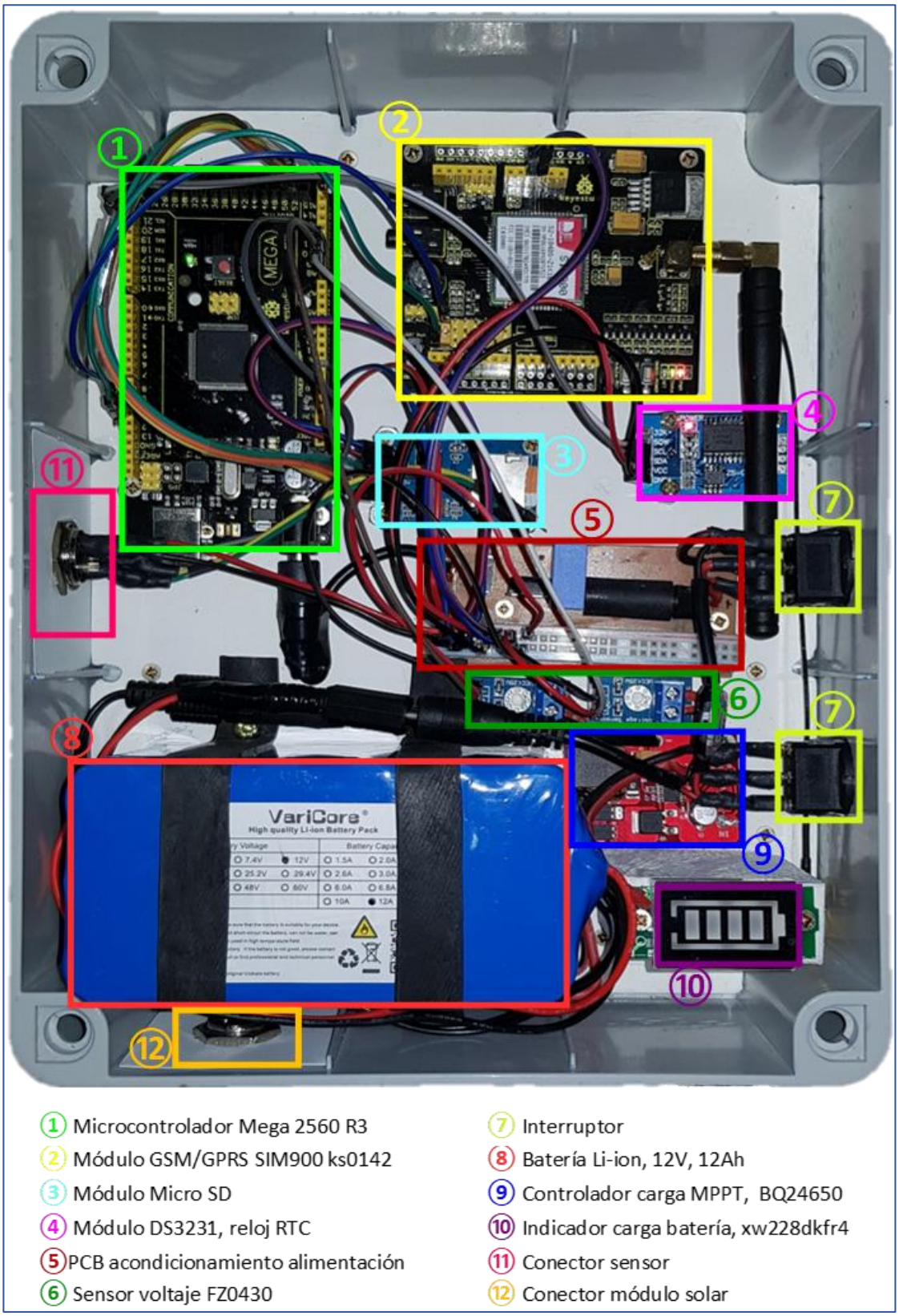


Figura 3. Prototipo del dispositivo de monitorización.

2. ESQUEMAS COMPLEMENTARIOS, TESTEO E IMPLEMENTACIÓN

En este apartado se muestran, en primer lugar, los esquemas generales del sistema hardware complementario al equipo principal de monitorización necesario para realizar el testeo e implementación del mismo. Dichos esquemas corresponden tanto al sistema de alimentación como al de acondicionamiento de la señal del sensor L.V.D.T. seleccionado. Además de estos esquemas, al igual que anteriormente para el equipo de monitorización, se presenta el prototipo del equipo del sensor con la identificación de sus componentes. Posteriormente se detalla el circuito desarrollado para el acondicionamiento de la señal del sensor y se explica su funcionamiento. Este sistema ha sido desarrollado específicamente para este caso. Por ello posteriormente se le dedica un apartado específico al acondicionamiento de la señal del sensor LVDT.

2.1. ESQUEMAS GENERALES DEL SISTEMA HARDWARE COMPLEMENTARIO

De los esquemas generales del sistema hardware complementario destacan en el sistema de alimentación del sensor el relé y el convertidor de tensión. Se destacan estos elementos por requerir especial atención para su correcta inclusión en el circuito al ser elementos nuevos no empleados en la alimentación del dispositivo de monitoreo, a diferencia del resto de componentes de la alimentación del sensor. Del sistema de acondicionamiento de señal destacar la placa pcb en la que se encuentran las resistencias y diodos encargados de acondicionar la señal, véase apartado 2.2. Esquemas específicos del sistema de acondicionamiento de señal.

En este caso se muestra primeramente el esquema simplificado de montaje y conexión de componentes del equipo del sensor, Figura 4 seguido por el esquema eléctrico del mismo, Figura 5. Después se presenta el prototipo del equipo del sensor, Figura 6 que se ha fabricado para poder realizar la comprobación del sistema de monitorización.

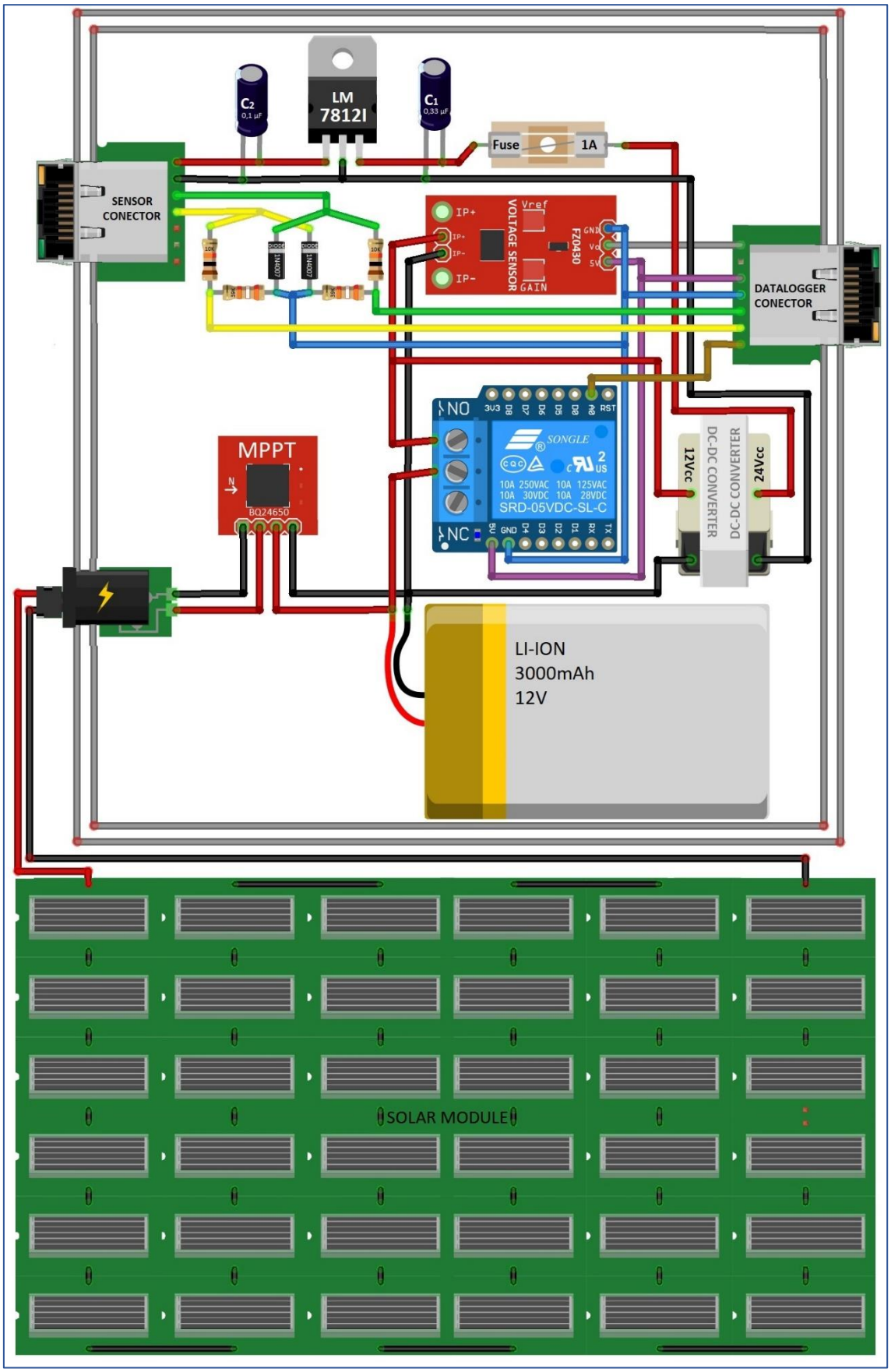


Figura 4. Esquema general simplificado de montaje y conexiones de componentes del equipo del sensor L.V.D.T.

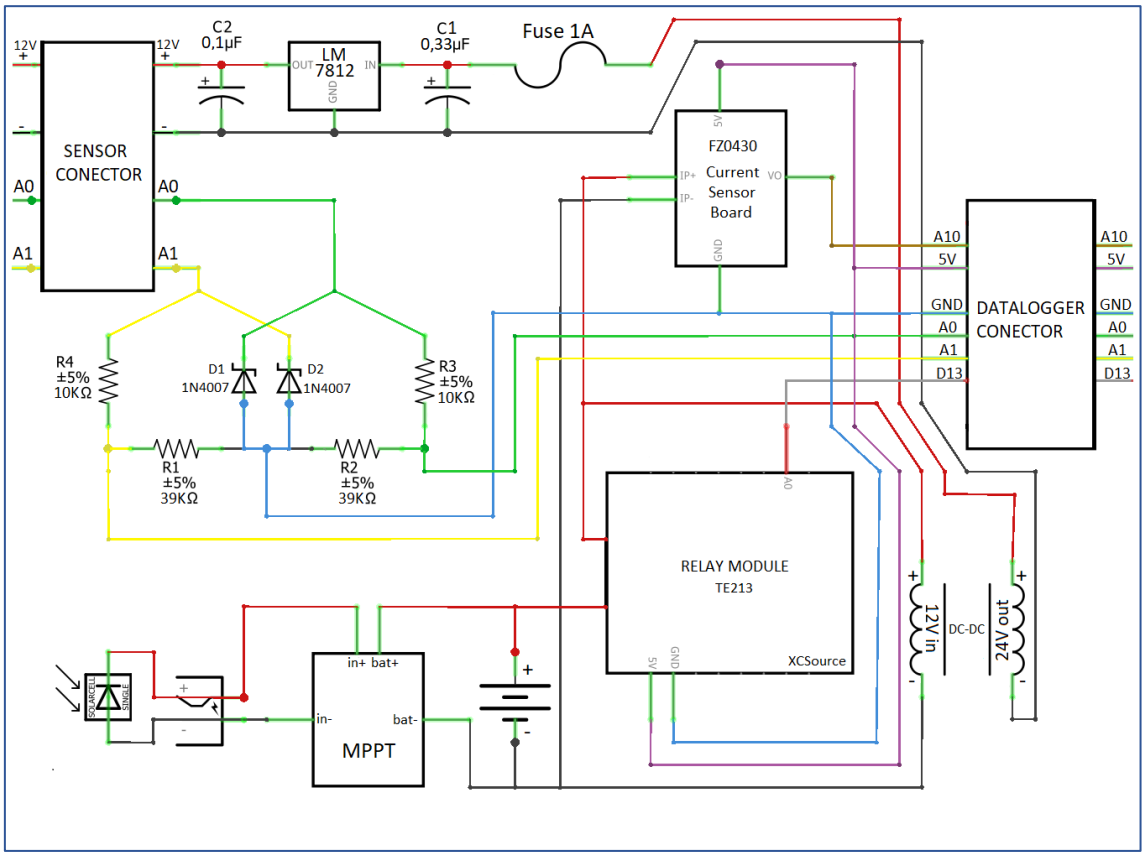


Figura 5. Esquema eléctrico general del equipo del sensor L.V.D.T.

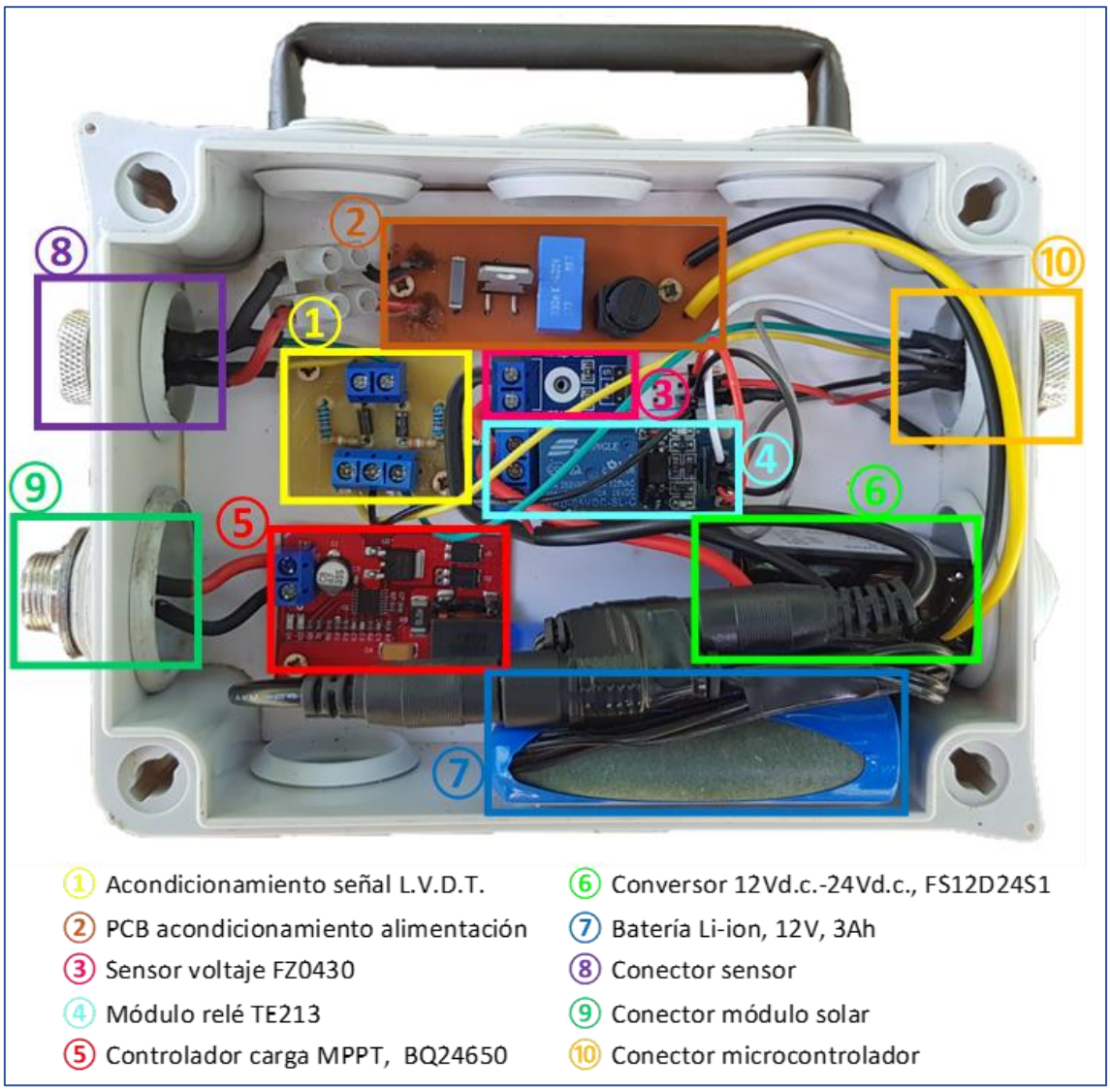


Figura 6. Prototipo del equipo del sensor L.V.D.T.

2.2. ESQUEMAS ESPECÍFICOS DEL SISTEMA DE ACONDICIONAMIENTO DE SEÑAL

A continuación, se detalla el circuito ideado para realizar el acondicionamiento de la señal del sensor que permite el monitoreo del mismo por parte del microcontrolador. Este tiene gran importancia, dado que sin él la señal emitida por el sensor L.V.D.T. es inadecuada para ser monitorizada por el microcontrolador, y en consecuencia no se podría realizar el testeo e implementación del mismo con este sensor.

Esto es debido a que el microcontrolador solo puede leer tensiones comprendidas entre 0 y +5V en corriente continua y el sensor proporciona un rango de tensiones de ±7,15V en corriente continua siendo alimentado a 12V, como es el caso. Lo que conlleva que el microcontrolador en estas condiciones no sea capaz de leer todos los datos emitidos por el sensor ya que no puede

leer las tensiones negativas ni las positivas superiores a 5V. Por tato, sin un acondicionamiento de la señal se producirían errores en las lecturas ya que habría datos fuera de rango de lectura que no podrían ser leídos, además de que las tensiones excesivas y negativas del L.V.D.T. dañarían el microcontrolador.

Teniendo en cuenta lo dicho, es necesario realizar un acondicionamiento de la señal del sensor que permita al microcontrolador leerla de la forma más adecuada, precisa y segura. Para hacer esto posible, en el acondicionamiento se reducirá el rango de tensiones emitido por el sensor ajustándolo lo máximo posible al rango de lectura de microcontrolador (de 0 a +5V). Con ello se evitarán los problemas anteriormente mencionados ya que el microcontrolador no recibirá tensiones negativas ni excesivas. Así se conseguirá una lectura adecuada y segura, que además al ajustar al máximo, será lo más precisa posible.

El hecho de conseguir la mayor precisión posible haciendo que la tensión de salida del sensor y la tensión de lectura del microcontrolador se ajusten lo máximo posible se explica porque de este modo se aprovechan los 1024 bytes de lectura disponibles en el microcontrolador, ya que estos están repartidos entre los 0 y +5V de lectura.

El acondicionamiento de la señal desarrollado se divide en dos partes. Una de ellas será la encargada de disminuir de forma proporcional la tensión emitida por el sensor de forma que la tensión máxima resultante pase de 7,15V a un valor inferior y lo más próximo posible a 5V. La otra, se encargará de transformar las tensiones negativas en tensiones positivas.

A continuación, se muestran el esquema simplificado de montaje y conexión de componentes del acondicionamiento de señal del sensor, Figura 7, seguido por el esquema eléctrico del mismo, Figura 8. En él se pueden ver los divisores de tensión formados por dos resistencias cada uno y empleados para reducir la tensión de la señal, así como los diodos rectificadores empleados para conseguir tensiones positivas. Posteriormente se profundiza en el funcionamiento de cada una de las partes y por ende del circuito de acondicionamiento resultante.

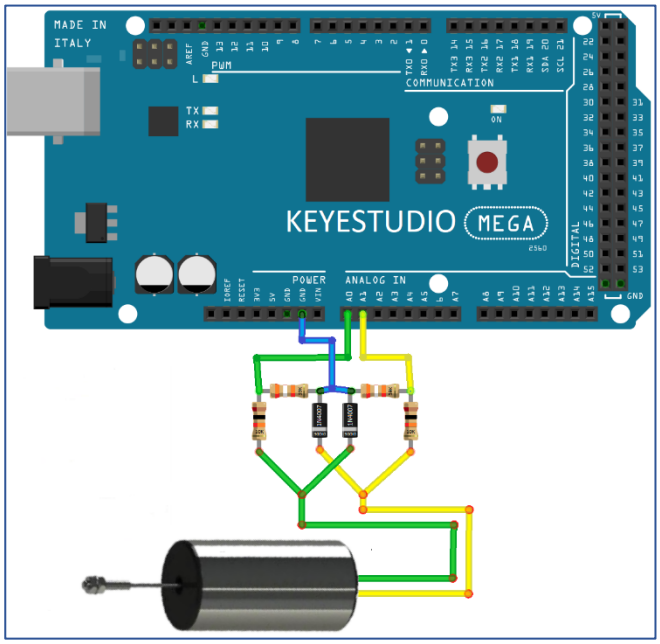


Figura 7. Esquema simplificado de montaje y conexiones de componentes del sistema de acondicionamiento de señal del sensor L.V.D.T.

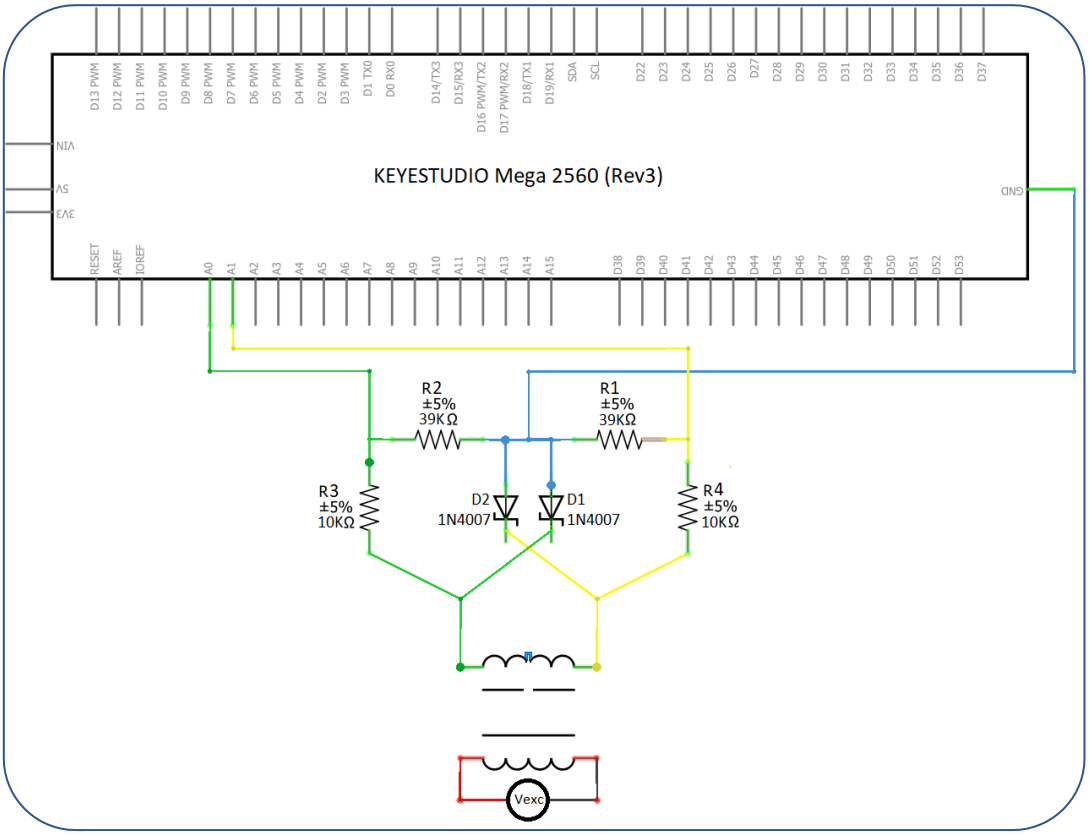


Figura 8. Esquema eléctrico del sistema de acondicionamiento de señal del sensor L.V.D.T.

2.2.1. Reducción de tensión

Para llevar a cabo la reducción de la tensión de la señal de forma proporcional se ha hecho uso de un divisor de tensión resistivo. Dicho divisor de tensión consiste en una configuración de un circuito eléctrico que reparte la tensión de una fuente entre una o más impedancias conectadas en serie, en este caso dos, véase Figura 9. Con dicha configuración se obtiene una tensión de salida V_{out} entre las dos resistencias inferior a la tensión de entrada V_{cc} y cuyo valor depende de la resistividad de las resistencias instaladas. En este caso, las resistencias seleccionadas son R_1 y R_2 con 39000 y 10000 ohm respectivamente, véase apartado 2.1.6.1.Resistencias del Anejo I para más información acerca de su elección. Con ellas el divisor de tensión resultante consigue reducir los 7,15V máximos iniciales a 5,69V máximos a su salida.

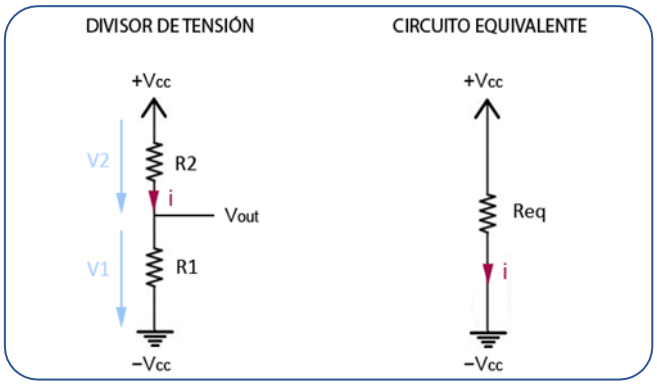


Figura 9. Esquema eléctrico divisor de tensión.

La tensión de salida máxima del divisor de tensión, V_{out} igual a 5,69V, sigue siendo superior al voltaje máximo de lectura del microcontrolador de 5V con el divisor de tensión calculado. A priori, dicha tensión de salida del sensor continuara siendo inadecuada y continuaran existiendo los problemas de sobretensión descritos anteriormente. Pero en realidad la tensión obtenida es adecuada debido a que hay que tener en cuenta que en el circuito de acondicionamiento de la señal del sensor también hay un diodo rectificador que consume 0,7V. Teniendo en cuenta todo lo comentado finalmente la tensión máxima de salida del circuito de acondicionamiento de la señal será igual a la de salida del divisor de tensión menos la tensión consumida por el diodo rectificador siendo por tanto la tensión máxima a la salida del acondicionamiento de 4,99V. Por ello con el circuito eléctrico de acondicionamiento explicado se consigue una tensión que resulta adecuada en valor a la que es capaz de leer el microcontrolador estando a la vez muy ajustada, por lo que se conseguirá la máxima precisión posible que el microcontrolador puede ofrecer a la vez que se solventan los problemas de sobretensión anteriormente comentados.

2.2.2. Conversión de tensiones negativas a positivas

Como se ha mencionado anteriormente para evitar que la señal del sensor LVDT tenga valores de tensión negativos y con ello evitar los problemas que este hecho acarrea es necesario llevar a cabo una conversión de las tensiones negativas a positivas respetando el valor de las mismas. Para ello primeramente se realizó una recopilación de información sobre los métodos existentes. De las soluciones recopiladas a priori tan solo dos de ellas resultaron viables para el presente caso.

La primera de ellas se basaba en la utilización de un componente electrónico conocido como ADS1115. Este componente es un conversor analógico digital (ADC) que usado en modo diferencial permite leer tensiones de ±2,5V. Al conectarlo de forma diferencial el microcontrolador tomara la tensión de 0V del sensor como la tensión media de lectura, en este caso 2,5V al tener un rango de lectura de 0 a 5 V. A partir de ahí, sumara a ese valor las tensiones positivas y le restara las negativas. Por tanto, para una tensión de salida del sensor de 2,5V con este componente el microcontrolador leerá una tensión de 5V y para una tensión de salida del sensor de -2,5V el microcontrolador leerá 0V. Esto hace que con el ADS115 el microcontrolador sea capaz de leer tensiones de hasta ±2,5V puesto que mayores tensiones superan su rango de lectura. En principio este método resulta muy interesante, pero tiene el inconveniente de resultar algo complejo. Ello es debido a que hay que reducir la tensión de señal de sensor y hay

que añadir un componente nuevo al circuito que además necesita de programación adicional. Debido a estos inconvenientes se ha descartado esta opción.

La segunda de ellas se basa en la utilización de un puente rectificador. Con él se evitarían las tensiones negativas, convirtiéndolas directamente a tensiones positivas del mismo valor. Esta solución es más sencilla ya que tan solo habría que instalar cuatro diodos rectificadores en una placa junto al divisor de tensión y no requiere de programación adicional. El problema de este método es que se falsearían los datos obtenidos. Para comprender mejor el motivo por el que esto sucede véase el apartado 2.1.1.Principio funcionamiento del Anejo I. Debido al principio de funcionamiento del sensor, la señal del mismo aumenta en valor conforme el núcleo se aleja del centro, y tiene signo positivo o negativo según en qué mitad del sensor se encuentre el núcleo, véase Figura 10. Por tanto, según el valor y signo de la señal se puede conocer la posición del núcleo en el sensor.

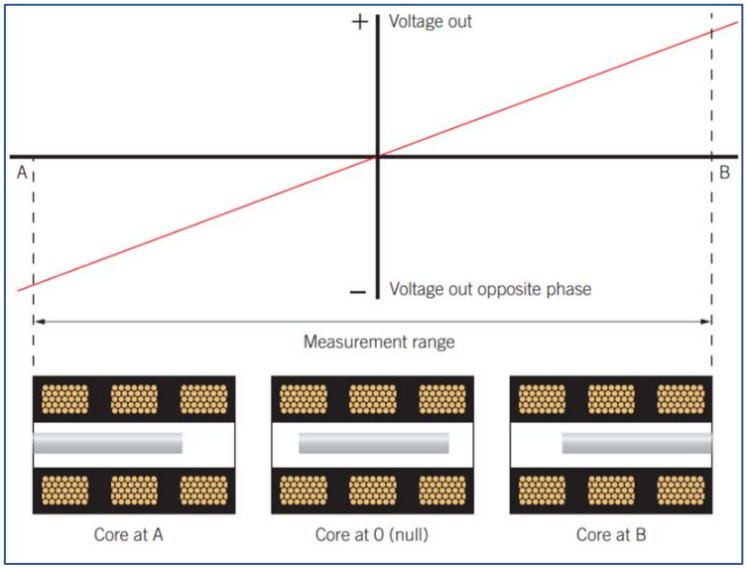


Figura 10. Funcionamiento de la señal del sensor L.V.D.T.

En cambio, con la solución propuesta se cambia el signo de los valores negativos a positivos, con lo que para cada valor de la señal habrá dos posiciones posibles del núcleo en el sensor equidistantes al centro del mismo puesto que el valor de la señal de ambos puntos es igual pero normalmente se diferencian por el signo del mismo, hecho que de este modo no sucede, véase Figura 11. Ello supone un problema porque hace que no sea posible saber con certeza la posición del núcleo, por lo que se descarta esta opción.

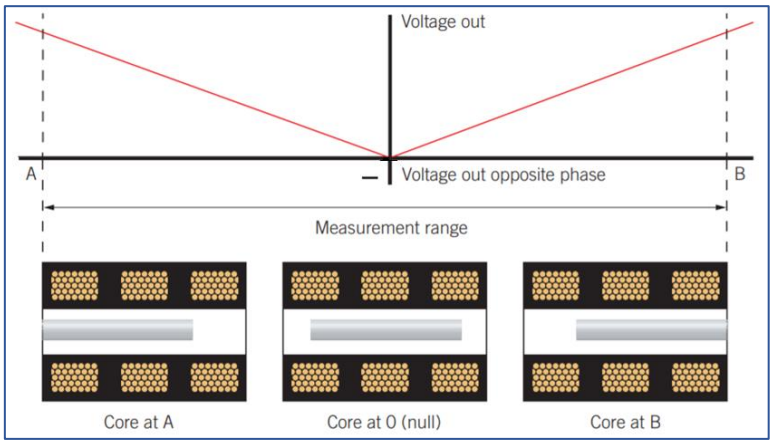


Figura 11. Funcionamiento de la señal del sensor L.V.D.T. con puente rectificador.

Finalmente, a partir de la solución del puente rectificador se idea una nueva solución con la que se aprovechan las bondades del mismo y se subsanan sus inconvenientes obteniendo con ello una solución sencilla de llevar a cabo, fiable y sin los problemas de las soluciones anteriormente comentadas. El circuito desarrollado como solución consiste básicamente en duplicar el circuito de la señal del sensor, haciendo coincidir las lecturas de una mitad del sensor con una de las dos salidas del acondicionamiento y la señal de la otra mitad (con tensión negativa) cambiarla de signo de tensión (hacer la tensión positiva) y leerla con la otra. De este modo el microcontrolador leerá a través de dos entradas analógicas el recorrido completo del núcleo a través del sensor, diferenciando una mitad de la otra según de la entrada analógica de la que provenga y subsanando así los inconvenientes del puente rectificador.

A continuación, se procede a explicar más detalladamente el principio de funcionamiento del sistema de acondicionamiento desarrollado para evitar las tensiones negativas en la señal del sensor LVDT. En primer lugar, se emplean dos diodos rectificadores los cuales solo permiten el paso de corriente continua de forma directa, es decir cuando la tensión es positiva. Estos son los encargados de evitar el paso de la señal negativa del sensor, evitando los problemas que ello acarrea. Además, para facilitar la explicación se considera que el sensor emite su señal a través de dos cables denominados cable 1 y cable 2 y que el núcleo del sensor puede estar posicionado en una mitad del sensor, denominada zona A, o en la otra, llamada zona B, véase Figura 12.

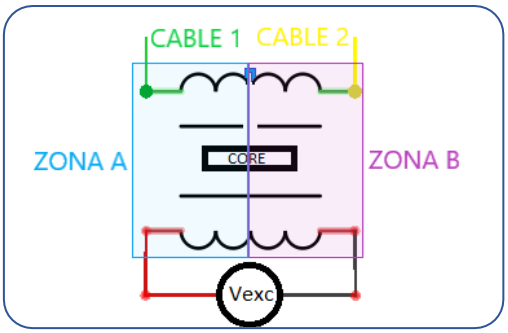


Figura 12. Esquema eléctrico simplificado sensor LVDT.

El simple uso de un diodo rectificador no es suficiente para lograr el objetivo deseado, ya que tan solo evita el paso de corrientes con tensión negativa. Para transformar las tensiones negativas en positivas se aprovecha el hecho de que, en corriente continua, si se conectan los polos de forma inversa resulta una tensión de igual valor, pero signo contrario. Por tanto, aplicándolo a este caso, conectando de forma inversa los cables de señal cuando el núcleo se encuentre en la zona que da una señal de tensión negativa, la tensión leída tendrá el mismo valor, pero será positiva. De igual modo cuando el núcleo pase a la otra mitad del sensor, donde a priori la tensión de señal es positiva, la señal leída tendrá una tensión de igual valor, pero signo cambiado, es decir será negativa.

Teniendo en cuenta todo lo expuesto anteriormente, si se unen en un mismo circuito la duplicación del circuito de señal, el concepto explicado en el párrafo anterior y el uso de diodos rectificadores que eviten el paso de corrientes de señal con tensión negativa se puede obtener un circuito de acondicionamiento de señal con dos salidas de señal con tensiones positivas, cada una correspondiente a la señal de una mitad del circuito. Para ello, en primer lugar, se duplica cada cable de la señal, tanto el cable 1 como el 2, obteniendo dos cables del cable uno, llamados cable1.1 y cable1.2. Y otros dos del cable 2 denominados cable2.1 y 2.2. Estos se conectan de tal forma que se forman dos circuitos. El primero formado por los cables 1.1 y el 2.1 (par de cables 1) y el segundo formado por los cables 1.2 y 2.2. (par de cables 2), véase Figura 13.

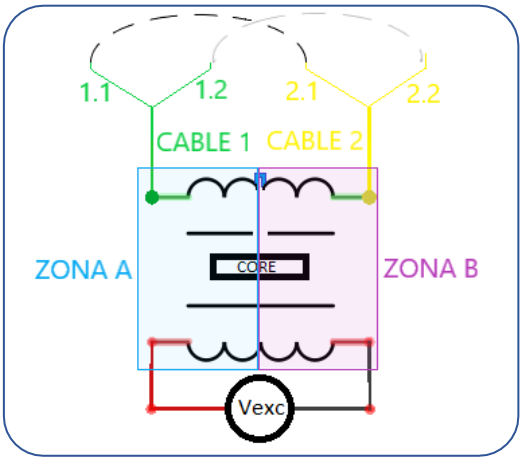


Figura 13.Esquema eléctrico simplificado del sensor L.V.D.T. con duplicado de salida de señal.

Llegados a este punto se va a completar el circuito mediante la instalación en cada subcircuito de dos diodos rectificadores. Estos se instalan de tal forma que en un subcircuito estén en sentido contrario al del otro, véase Figura 14. Con ello se consigue que en cada uno de los subcircuitos formados circule la electricidad en un sentido y que cuando esto suceda que la misma tenga una tensión positiva puesto que los diodos están polarizados de forma directa y no inversa, dejando circular la corriente eléctrica tan solo si su tensión es positiva respecto a ellos. Al circular la corriente eléctrica por cada subcircuito en un sentido, ésta tan solo circulará por uno de ellos en función de la zona en que se encuentre el núcleo ya que para cada zona la corriente tendrá un sentido. Además, ambos subcircuitos se conectan a tierra en puntos intermedios a los dos diodos de cada uno de ellos para hacer posible la medición de la tensión de los mismos por parte del microcontrolador. Es necesario conectar a tierra entre ambos diodos de cada circuito porque de conectarse en un punto previo o posterior a ellos se produciría un puente de los mismos con lo que la función de estos quedaría anulada y por ende el acondicionamiento quedaría anulado. Con todo ello, sabiendo por cuál de los dos subcircuitos

circula la electricidad se conocerá en que zona se encuentra el núcleo y con el valor medido de tensión su distancia al centro, quedando localizada su posición dentro del sensor.

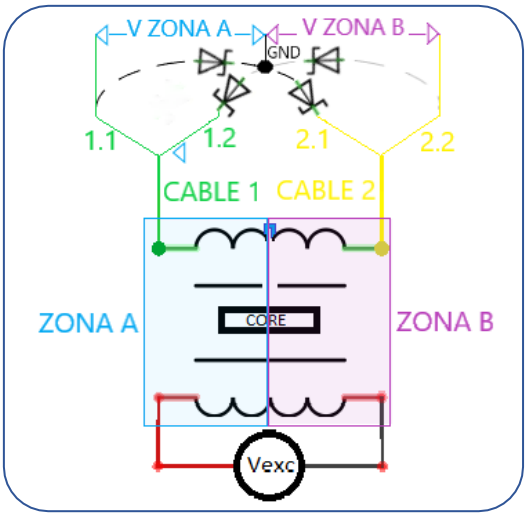


Figura 14. Esquema eléctrico simplificado sensor L.V.D.T. con circuitos de acondicionamiento de señal completos.

Veamos esta explicación aplicada al esquema de la Figura 14Figura 15. Si el núcleo se desplaza a la zona A la corriente tratara de circular del cable 1 al cable 2 debido al principio de funcionamiento del sensor. En este caso, en el circuito de la figura, los diodos instalados entre el cable 1.1 y el 2.2 permitirá el paso de la corriente por él y, en cambio, los diodos instalados entre los cables 1.2 y 2.2, lo impedirán puesto que la corriente trata de circular de forma inversa respecto a ellos. Por tanto, al medir la tensión entre el cable 1.1 y 2.1 o tierra, se obtendrá una medida de tensión positiva. En cambio, si medimos la tensión entre el cable 2.2 y el 1.2 o tierra, no se medirá tensión eléctrica puesto que los diodos rectificadores impiden la circulación eléctrica. Por tanto, con este circuito de acondicionamiento se obtiene en un subcircuito una señal en forma de tensión positiva perteneciente a una de las mitades del sensor. Y en el otro, otra señal en forma también de tensión positiva cuando el núcleo se encuentra en la otra mitad del sensor.

Por tanto, finalmente tendremos dos salidas del acondicionamiento que siempre serán positivas, en las que la señal del primer par de cables(par1) emitirá solo tensión positiva cuando el núcleo se encuentre en la zona A y no emitirá nada en cuando el núcleo esté en la zona B. El segundo par de cables (par2) emitirá solo tensión positiva cuando se encuentre en la zona B y no emitirá nada en la zona A. Teniendo así que el par 1 emitirá la señal de la zona A y el par 2 emitirá la señal de la zona B.

Una de las grandes ventajas a destacar y tener en cuenta de esta solución respecto a las anteriormente mencionadas es que se mejora la precisión de lectura del sensor respecto de las otras soluciones. A la vez que es una solución fácil de implementar. Esta mejora se debe a que la lectura del sensor se divide en dos mitades y cada una de ellas es leida con una entrada distinta del microcontrolador. Esto hace que se dispongan de 2048bytes para leer la señal del sensor en lugar de 1024 que se dispondrían de usar tan solo una entrada analógica para leer todo el sensor. Por tanto, debido a su sencillez, buen funcionamiento y al aumento tan importante de precisión que ofrece el circuito desarrollado, este es el elegido para el acondicionamiento de la señal de los LVDT.

Una vez seleccionadas las soluciones del acondicionamiento de señal se decide unir ambas montando todos los elementos en un circuito impreso en una placa pcb. Para unirlas se ha decidido montar para cada subcircuito de conversión de tensión negativa a positiva un divisor de tensión igual al explicado anteriormente, obteniendo con ello señales acondicionadas con tensiones dentro del rango de lectura, siempre positivas y de gran precisión. Hay que tener en cuenta que en la unión de estas soluciones se ha llevado a cabo una simplificación del circuito consistente en la eliminación de los diodos dispuestos aguas arriba de las conexiones a tierra. Ambos diodos se han eliminado puesto que en su lugar se han instalado divisores de tensión, que, a efectos prácticos son capaces de realizar su misma función en este circuito. Esto se explica porque los divisores de tensión ejercen gran resistencia eléctrica, impidiendo el paso de corriente a través de ellos y haciendo que la corriente eléctrica circule principalmente por el otro subcircuito, que ejerce poca resistencia eléctrica. Teniendo por tanto un circuito equivalente al circuito con todos los diodos instalados.

El montaje del circuito de acondicionamiento se muestra en la Figura 7 y en la Figura 8. En ellas se pueden apreciar las modificaciones realizadas en lo que respecta al circuito desarrollado para la conversión de las tensiones negativas. A continuación, se explica el funcionamiento del mismo en los dos posibles casos.

En el primer caso el núcleo se encuentra desplazado hacia la parte izquierda según el esquema (Figura 15). En este caso, el núcleo inducirá mayor corriente eléctrica en la bobina secundaria izquierda del LVDT, por lo que la corriente eléctrica ira del cable verde al amarillo. Para este caso el diodo izquierdo que une el cable verde con el amarillo permite la circulación de corriente, en cambio el otro diodo, instalado en sentido inverso, la impide por lo que la corriente eléctrica circulara en el subcircuito de la izquierda y no por el de la derecha. En él se encuentra el divisor de tensión instalado, por lo que el microcontrolador tomara la señal entre las dos resistencias del mismo y tierra. En cambio, si se toma la señal del subcircuito derecho la señal será cero puesto que no circula corriente eléctrica por él. Destacar que para la instalación del divisor de tensión se tiene que tener en cuenta el sentido de circulación de la corriente puesto que en función de este van colocadas las resistencias. Concretamente la corriente eléctrica ha de pasar primero por la resistencia de 39KΩ y después por la de 10KΩ hasta tierra.

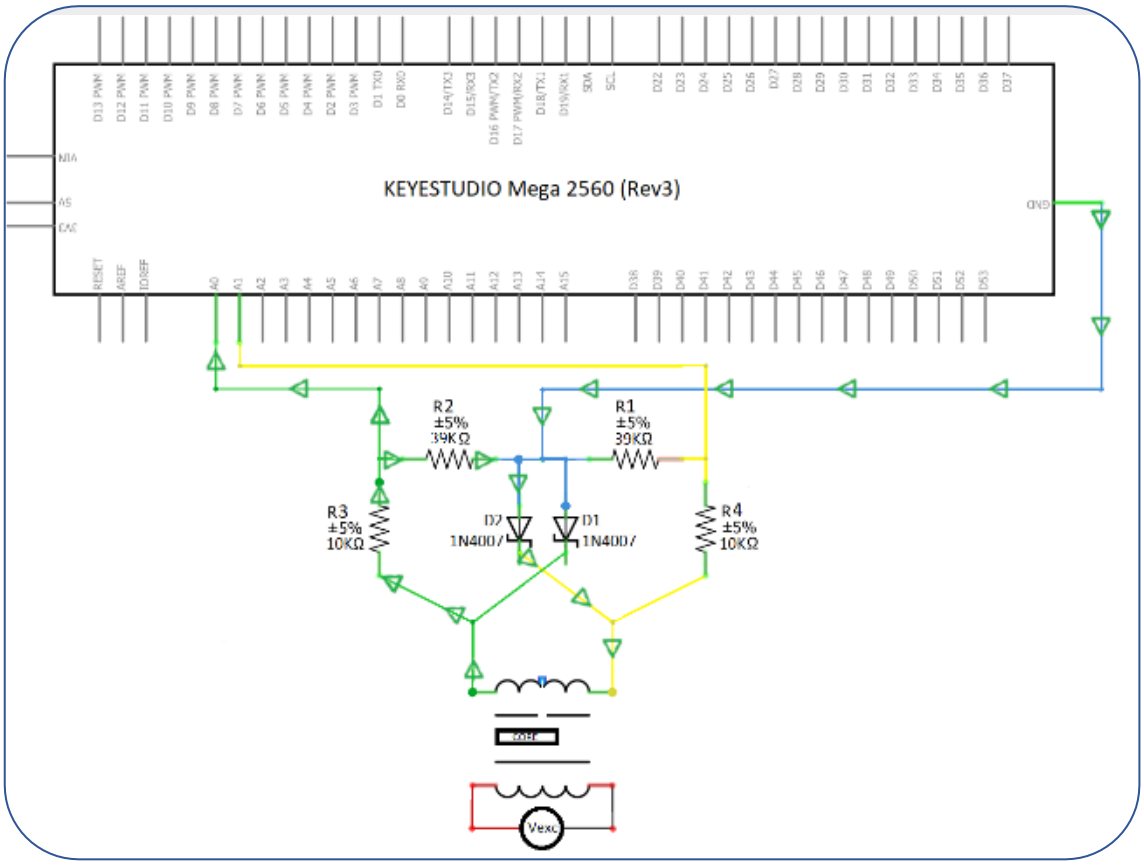


Figura 15. Esquema eléctrico del sistema de acondicionamiento de señal del sensor L.V.D.T. en funcionamiento con núcleo en zona izquierda.

El segundo caso en el que el núcleo se encuentra desplazado hacia la parte derecha del esquema (Figura 16), el núcleo inducirá mayor corriente eléctrica en la bobina secundaria derecha del LVDT, por lo que la corriente eléctrica ira del cable amarillo al verde. Para este caso el diodo derecho que une el cable amarillo con el verde permite la circulación de corriente, en cambio el otro diodo, instalado en sentido inverso, la impide. Por lo que la corriente eléctrica circulara por el subcircuito de la derecha y no por el de la izquierda. En él se encuentra el divisor de tensión instalado, por lo que el microcontrolador tomara la señal entre las dos resistencias del mismo y tierra. En cambio, si se toma la señal del subcircuito izquierdo la señal será cero puesto que no circula corriente eléctrica por él. Cabe destacar de nuevo, que para la instalación del divisor de tensión se tiene que tener en cuenta el sentido de circulación de la corriente puesto que en función de éste deben colocarse las resistencias. Concretamente la corriente eléctrica ha de pasar primero por la resistencia de 39KΩy después por la de 10KΩ hasta tierra.

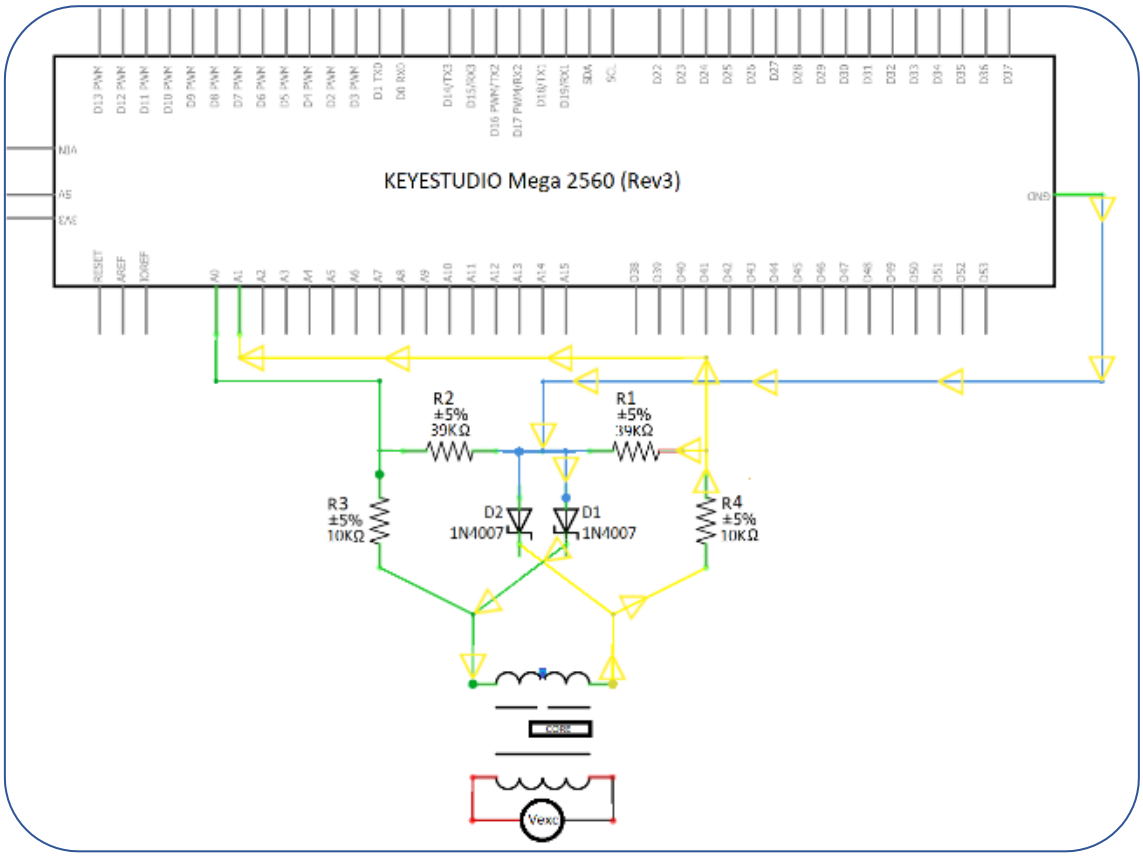


Figura 16. Esquema eléctrico del sistema de acondicionamiento de señal del sensor L.V.D.T. en funcionamiento con núcleo en zona derecha.

ANEJO III: SOFTWARE DEL SISTEMA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ÍNDICE

ÍNDICE DE FIGURAS

1. SOFTWARE DEL SISTEMA

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 e instalado en el equipo.

Para la creación y carga en el microcontrolador del software se ha empleado un ordenador y el programa de desarrollo de software libre Arduino IDE (Integrated Development Environment o Entorno de Desarrollo Integrado). El programa de desarrollo IDE se caracteriza por estar diseñado expresamente para la creación y carga de software en placas del tipo empleado en este trabajo, así como por tener un entorno muy sencillo de uso(ARDUINOCC, 2019). Por ello se ha decidido emplear este programa para el desarrollo y carga del software en el equipo. Para su uso, el programa es instalado y ejecutado en el ordenador. Una vez hecho esto, con él se desarrolla y escribe el software que se quiere que sea ejecutado por el microcontrolador. Una vez escrito, se conecta la placa del microcontrolador al ordenador a través de USB y se carga el software en el mismo. Cuando el software ha sido cargado en la placa, el microcontrolador comienza a ejecutar el software y, por tanto, a trabajar de forma autónoma.

En cuanto al software desarrollado, este consiste en un sketch (programa del microcontrolador) que está compuesto por una serie de códigos escritos en lenguaje similar a C++. Estos códigos, como se ha visto en el párrafo anterior, son desarrollados en un ordenador y se cargan en el microprocesador(ARDUINOCC, 2019). Una vez cargados son ejecutados por el microcontrolador, dándole las instrucciones necesarias para llevar a cabo las tareas automatizadas que se desean hacer, como son la lectura de sensores o el almacenamiento y envío de información.

Cabe mencionar que es posible incluir librerías de código libre en el sketch desarrollado. Dichas librerías son líneas de código escrito por terceros que facilitan la ejecución de una serie de funciones específicas. Estas se pueden emplear en cualquier proyecto con la finalidad de reutilizar código, con lo que se ahorra trabajo de programación. Además, para su uso, tan solo se requiere de la importación de la librería y definición de las variables necesarias para realizar la acción deseada, con lo que se consigue reducir el número de líneas de código obteniendo un sketch más sencillo tanto de hacer como de entender(CÁMARA, 2017). Debido a esta gran ventaja que presenta el uso de librerías, en este trabajo se ha hecho uso de algunas de ellas, como posteriormente se detallará.

Los programas empleados en este tipo de placas se llaman sketch y constan de, al menos, dos partes diferenciadas, las cuales son obligatorias: función setup() y función loop().

La primera, formada por la función setup, será la parte que se encarga de recoger la configuración del dispositivo con detalles como la velocidad de transmisión de la información (o baudios), la inicialización de los puertos serie que se vayan a emplear y la asignación de pines (entradas y salidas analógicas y digitales) y solamente se ejecutará una vez al inicio del programa. Una vez finalizado el setup se ejecuta constantemente el loop (MARTÍNEZ, 2015).

La segunda parte, formada por función loop, se corresponde con el bucle que contiene el programa principal, donde se incluyen todas las sentencias necesarias para obtener la funcionalidad deseada (CÁMARA, 2017). En él se programan las instrucciones que queremos realizar periódicamente, como la lectura o escritura de datos sobre los pines digitales y analógicos deseados. Puesto que es un bucle, se debe controlar el periodo de ejecución del mismo con alguna instrucción. Esta es delay (milis) siendo milis el tiempo en milisegundos que

el programa esperará hasta pasar a la siguiente línea (o volver a repetir el bucle si esa instrucción era la última línea del void loop() (MARTÍNEZ, 2015). Esta segunda parte puede darse el caso de que sea demasiado extensa o compleja, lo que puede conllevar problemas de funcionamiento. Como solución a este problema se pueden emplear funciones complementarias a la función principal loop. Estas funciones complementarias son utilizadas por la función principal loop, y en ellas se recogen partes del código principal, haciendo que éste se simplifique y con ello se eviten problemas de mal funcionamiento.

Antes de la función setup(), se pueden añadir comentarios de cómo es el funcionamiento del sketch. Además, se incluyen todas las librerías necesarias, 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 pudiendo ser empleadas por varias funciones (MARTÍNEZ, 2018; ALONSO, 2018).

A continuación, se procede a realizar una explicación detallada del código de programación desarrollado e instalado en el equipo. En este caso el código desarrollado en este sketch consta de las tres partes diferenciadas anteriormente. Por ello y dada su extensión se realiza la explicación de cada una de estas tres partes por separado.

1.1. CÓDIGO INICIAL, inclusión de librerías y definición de variables.

En esta parte del programa, se deben incluir todas las librerías necesarias para que puedan ejecutarse las distintas funciones necesarias para poder controlar cada uno de los componentes del sistema, como son el módulo SD, el reloj RTC, La función Sleep y el módulo GSM/GPRS. Además, se definen los pines empleados, y se declaran las variables y constantes globales, las cuales se usarán a lo largo de todo el programa, pudiendo ser utilizadas por varias funciones tanto de los módulos y la función Sleep, como de los sensores LVDT y de control de voltajes del equipo. El código se muestra en La Figura 1.

```
//SRAM LIBRE
#include <MemoryFree.h>

//MILISEGUNDOS DE FUNCIONAMIENTO
unsigned long time;

//WDT/RESET
#include <avr/wdt.h> // Incluir la librería que contiene el watchdog (wdt.h)

//LVDT
const int sensorPin1 = A0;
const int sensorPin2 = A1;
int sensor1=0;
int sensor2=0;
int rele = 13;

//SD:
#include <SD.h>
File logFile;
```

```
//RELOJ:
#include <Wire.h>
#include "Sodaq_DS3231.h"

char weekDay[][4] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

//year, month, date, hour, min, sec and week-day(starts from 0 and goes to 6)
//writing any non-existent time-data may interfere with normal operation of the RTC.
//Take care of week-day also.
DateTime dt(2011, 11, 10, 15, 18, 0, 5);

//SLEEP:
#include <SnoozeLib.h>  //Include the library
int state = HIGH;//Led State

//GSM
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial Sim900Serial(10, 11);//Configurarión de los pines serial por software

//BATERIA
int SensorBat = A8;
int SensorPanSol = A9;
int SensorLVDT = A10;
```

Figura 1. Código inicial correspondiente a la inclusión de librerías y declaración de variables.

Las librerías incluidas son:

MemoryFree.h, se utiliza para obtener la memoria RAM disponible;

avr/wdt.h, librería que contiene el watchdog (wdt.h), para controlar el tiempo que lleva en funcionamiento el microcontrolador y resetear su RAM cuando esta se llene;

SD.h, para el funcionamiento del módulo SD;

Wire.h y Sodaq_DS3231.h para obtener la fecha, hora y temperatura con el módulo RTC;

SnoozeLib.h para la función Sleep que pone en modo bajo consumo al microcontrolador;

SoftwareSerial.h y String.h para el envío de datos y avisos por SMS con módulo GSM/GPRS.

En esta parte del programa, también se declaran las constantes y variables globales:

Variable ”time”: para posteriormente conocer el tiempo que lleva en funcionamiento el microcontrolador.

Constante “sensorPin1” y “sensorPin2”: para leer la señal del LVDT a través de los pines analógicos 1 y 2, respectivamente.

Variables “sensor1” y “sensor2”: se igualan a 0 para obtener el valor de cada una de las dos señales del dendrómetro como resultantes de la media de 10 lecturas realizadas.

Variable “relé”: se iguala a la lectura del pin analógico 13 para activar y desactivar el relé encargado de controlar la alimentación del sensor.

Variable “SensorBat”: se iguala a la lectura del pin analógico 8 para leer la entrada de la señal correspondiente al voltaje de la batería del equipo

Variable “SensorPanSol”: se iguala a la lectura del pin analógico 9 para leer la entrada de la señal correspondiente al voltaje del panel solar

Variable “SensorLVDT”: se iguala a la lectura del pin analógico 10 para leer la entrada de la señal correspondiente al voltaje de la batería del equipo del sensor.

Tras la inclusión de la librería SD.h y para que con ello se pueda controlar la tarjeta donde se guardan los datos se utiliza la función logFile. Para ello se emplea el comando “File logFile“.

Por otro lado, en aplicaciones de este tipo, es importante establecer exactamente la fecha y la hora en que se toma cada una de las lecturas. Para ello, primero se define una variable “weekDay” de tipo char para dar el nombre a los días de la semana. Después, con el comando “DateTime dt” se pone en fecha y hora el reloj del módulo DS3231. Es importante que este comando se encuentre desactivado puesto que si no cada vez que encendiera el microcontrolador se reestablecerían la fecha y hora a la indicada con el comando, pudiendo quedar mal configuradas si no se actualiza en el código. Por tanto, este permanece desactivado y tan solo se activará cuando se configure el reloj, volviéndolo a desactivar posteriormente.

A continuación, se incluye la librería SnoozeLib.h que permite poner al microcontrolador en estado de bajo consumo o totalmente activo. En la línea siguiente se pone en estado activo mediante la definición de la variable “state” a la que se le da el valor HIGH.

Después, y tras la inclusión de las librerías SoftwareSerial.h y String.h necesarias para el control del módulo GSM/GPRS se configuran los pines seriales 10 y 11 como Tx y Rx con el comando “SoftwareSerial Sim900Serial (10, 11)”. A través de ellos se realiza la comunicación entre el microcontrolador y el shield GSM/GPRS.

1.2. CÓDIGO FUNCIÓN SETUP, configuración del dispositivo.

Tras el primer bloque de código anterior dedicado a la inclusión de librerías y definición de variables, el sketch continúa con el segundo gran apartado del que consta, en este caso destinado a la configuración del dispositivo. Este consiste en una función denominada función setup() que solamente se ejecutará una vez al inicio del programa. En ella se encuentra el código que recoge, como ya se ha comentado, la configuración del dispositivo con detalles como la velocidad de transmisión de la información (o baudios), la inicialización de los puertos serie que se vayan a emplear y la asignación de pines (entradas y salidas analógicas y digitales).

En la Figura 2 se muestra la función setup con el código desarrollado en ella, el cual se explica a continuación.

La primera línea de este apartado de configuración se dedica a establecer la función de configuración setup y a indicar el comienzo del código de configuración que en ella se encuentra. Para ello se hace uso del comando “void setup() {”. Como se puede ver, primero se establece la función y finalmente con la llave de apertura se da pie al comienzo del código de configuración de la misma, el cual se desarrolla en líneas posteriores. Todo el código se ejecutara al ejecutar la función.

```
void setup() {

  Serial.begin(9600);

  //LIBRERIA WDT
  wdt_disable(); // Desactivar el watchdog mientras se configura

  //GSM
  Sim900Serial.begin(19200);//Arduino se comunica con el SIM900 a una velocidad de 19200bps
      //Encendido del módulo por sofware
    //digitalWrite(9, HIGH);
    //delay(1000);
    //digitalWrite(9, LOW);
    //delay(20000);//Tiempo prudencial para el escudo inicie sesión de red con tu operador

  //LVDT
  pinMode(0,INPUT);
  pinMode(1,INPUT);
  pinMode(rele, OUTPUT); //modo salida

  //SD:
  Serial.print("Iniciando microSD ...");
  if (!SD.begin(53))
    {
    Serial.println("Error al iniciar");
    return;
    }
  Serial.println("Iniciado correctamente");

  //RELOJ:
  Wire.begin();
  rtc.begin();
 // rtc.setDateTime(dt); //Adjust date-time as defined 'dt' above

  //SLEEP:
  pinMode(6,OUTPUT);
  attachInterrupt(0,wakeup,RISING); //Attaching the 0 Interrupt for RISING.

}
```

Figura 2. Código Setup, configuración del dispositivo

La primera línea del código de configuración ubicado dentro de la función setup configura la velocidad de transmisión de la información, en este caso de 9600 baudios mediante el comando "Serial.begin(9600)", Figura 3.

Tras esta primera línea se desactiva el watchdog para que este no interfiera en la configuración y esta se lleve a cabo correctamente. Ello se consigue mediante el comando "wdt_disable()" de la librería avr/wdt.h. Véase Figura 3.

```
void setup() {

  Serial.begin(9600);

  //LIBRERIA WDT
  wdt_disable(); // Desactivar el watchdog mientras se configura
```

Figura 3. Velocidad de transmisión y desactivación del watchdog.

A continuación, se configura la velocidad de transmisión de información entre el microcontrolador y el módulo GSM/GPRS sim900. Esta se establece en 19200 baudios para un correcto funcionamiento con el comando "Sim900Serial.begin(19200)". Además, para los módulos GSM/GPRS que lo permitan, se dedican unas líneas, en este caso desactivadas, al encendido del mismo por software. Este se lleva a cabo mediante el pin digital número 9. Para ello primero se cambia el estado del pin digital 9 a activo mediante el comando "digitalWrite(9, HIGH)". Después se da un segundo de espera con el comando "delay(1000)" (1000 milisegundos de espera). Tras esta breve espera se desactiva el pin digital 9 con el comando "digitalWrite(9, LOW)" y de nuevo se da un tiempo prudencial de espera necesario para que inicie el módulo, en este caso de dos segundos, véase Figura 4.

```
//GSM
Sim900Serial.begin(19200);//Arduino se comunica con el SIM900 a una velocidad de 19200bps
    //Encendido del módulo por sofware
  //digitalWrite(9, HIGH);
  //delay(1000);
  //digitalWrite(9, LOW);
  //delay(20000);//Tiempo prudencial para el escudo inicie sesión de red con tu operador
```

Figura 4. Velocidad de transmisión shield GSM/GPRS y encendido automático del mismo.

Una vez configurado el módulo GSM/GPRS se pasa a la configuración relacionada con el L.V.D.T. En ella tan solo se establecen los pines analógicos 1 y 2 como entradas, aunque este paso no es estrictamente necesario, puesto que los pines funcionan por defecto como entradas. Para ello se emplean los comandos "pinMode(0,INPUT)" y "pinMode(1,INPUT)". En la siguiente línea se establece el pin que activara y desactivara el relé de control de la alimentación del sensor como pin de salida mediante el comando "pinMode(rele, OUTPUT)". Véase Figura 5.

```
//LVDT
pinMode(0,INPUT);
pinMode(1,INPUT);
pinMode(rele, OUTPUT); //modo salida
```

Figura 5. Establecimiento de los pines del dendrómetro y relé.

Después se comprueba la conexión del microcontrolador con el módulo microSD. Para ello primero se manda un mensaje al monitor serie indicando que se está iniciando el módulo microSD con el comando "Serial.print("Iniciando microSD ...")". Después se lleva a cabo la comprobación. Para ello se comprueba si hay señal en el pin 53 de comunicación con el shield mediante el uso de una sentencia condicional if con el comando "if (!SD.begin(53))". Este comando hace que en caso de que no haya comunicación a través del pin 53 se lleve a cabo el código posteriormente indicado entre llaves. Este código entre llaves consta de un mensaje de error al iniciar que se manda al monitor serial mediante el comando "Serial.println("Error al iniciar")" y también del comando "return" para que vuelva a ejecutarse el código desde el principio. En cambio, si no se cumple dicha condición, y si hay comunicación, se saltara el código entre llaves y se mandara un mensaje de inicio correcto mediante el comando "Serial.println("Iniciado correctamente")" y se continuara la ejecución del sketch. Véase Figura 6.

```
//SD:
Serial.print("Iniciando microSD ...");
if (!SD.begin(53))
  {
  Serial.println("Error al iniciar");
  return;
  }
Serial.println("Iniciado correctamente");
```

Figura 6. Comprobación del estado de la conexión del módulo microSD.

El siguiente paso en el código de este bloque se dedica a la configuración del módulo DS3231. Tan solo se inician las librerías que se emplean para el control de este módulo. Ello se lleva a cabo mediante los comandos "Wire.begin()" y "rtc.begin()". Además, se encuentra el comando "rtc.setDateTime(dt)" que esta desactivado ya que se emplea para hacer que se ajuste la fecha y hora del reloj a la indicada anteriormente en la función "dt" y esto solo se realiza cuando es necesario. Figura 7.

```
//RELOJ:
Wire.begin();
rtc.begin();
// rtc.setDateTime(dt); //Adjust date-time as defined 'dt' above
```

Figura 7. Inicio de librerías de control del módulo DS3231.

Por último, se ajusta la configuración del modo bajo consumo del microcontrolador. En ella se establece el pin 6 como salida mediante el comando "pinMode(6,OUTPUT)" necesario para controlar el modo de actividad en que se encuentra el microcontrolador con la librería SnoozeLib.h. Seguidamente se hace una interrupción con el comando "attachInterrupt(0,wakeup,RISING)". Esta interrupción se identifica como la interrupción número 0, su ISR (Interruption Service Routines) es la función wakeup y responde ante un evento de tipo RISING(la interrupción ocurre en el flanco de subida de LOW a HIGH del pin elegido). Esto quiere decir que cuando el pin6 anteriormente establecido pase de estar en estado low a high en cualquier punto del código se detendrá la ejecución del mismo y se ejecutara el ISR de la interrupción, en este caso la función wakeup. Dicha función, hace que el microcontrolador pase del estado de bajo consumo a totalmente activo, como posteriormente se verá. Por tanto, con esta interrupción se consigue cambiando el estado del pin 6 de low a high. Una vez finalizada la interrupción, en este caso cuando el microcontrolador haya sido activado, la ejecución del sketch continuará en el mismo punto en que se había detenido. Véase Figura 8.

```
//SLEEP:
pinMode(6,OUTPUT);
attachInterrupt(0,wakeup,RISING); //Attaching the 0 Interrupt(on Arduino UNO pin 2) for RISING.
```

Figura 8. Ajuste configuración modo bajo consumo.

Tras esta última línea de código de configuración, se hace uso de la llave de cierre de la función setup. Con ello se finaliza el código de configuración de la misma.

1.3. CÓDIGO FUNCIÓN LOOP, programa principal y funciones complementarias.

Una vez realizada la configuración del dispositivo en el segundo bloque con la función setup, el sketch continúa con la tercera y última parte diferenciada del mismo.. Esta consta de una función principal, la función loop, que se corresponde con el bucle que contiene el programa principal. A ella se suman otras funciones secundarias, las cuales son empleadas por la principal y que sirven para simplificar el programa de la función loop y con ello conseguir un mejor funcionamiento y comprensión del mismo. La función loop junto a sus funciones complementarias contienen las instrucciones que queremos realizar periódicamente. Entre estas instrucciones se encuentran la lectura del sensor LVDT, así como la del resto de sensores de control de estado del equipo; el almacenamiento de datos en la plataforma IoT y en la tarjeta microSD, además, del cambio de estado de actividad del microcontrolador entre otros.

Dada la gran extensión de todo el código empleado en este apartado y puesto que se utiliza una función principal y varias complementarias se procede a explicar y mostrar el código de cada una de estas por separado.

1.3.1. Código función principal loop.

En primer lugar, se va a proceder a mostrar y explicar el código empleado en la función principal, llamada función loop. Este código se encarga de gestionar y ejecutar todas las instrucciones a realizar, bien directamente, o bien indirectamente mediante la ejecución de las funciones complementarias. A continuación, en la Figura 10 se muestra el código empleado en la función principal loop. Además, posteriormente se lleva a cabo la explicación detallada del mismo.

Al igual que sucedía en el bloque anterior, este apartado comienza con el nombre de la función. En este caso la función de este apartado es la función principal loop. Dicha función loop se establece mediante el comando “void loop() {“. Mediante el uso de este comando primero se establece la función, y después, con la llave de apertura, se da pie al comienzo del código del programa principal del software que se contiene en ella. Este código se desarrolla en las líneas posteriores finalizando antes de la llave de cierre de la función y siendo ejecutado en bucle cada vez que se ejecute la función. Véase Figura 9.

```
void loop() {
  //WDT/RESET
  if(time>6500000) {
    wdt_enable(WDTO_15MS); // Configurado a 15 milisegundos
    delay(20);
  }
```

Figura 9. Establecimiento de función loop y reseteo del microcontrolador

```
void loop() {
  //WDT/RESET
  if(time>6500000) {
    wdt_enable(WDTO_15MS); // Configurado a 15 milisegundos
    delay(20);
  }

  //LVDT
  digitalWrite(rele, LOW);
  delay(500);

  //GSM
  comandosAT();//Llama a la función comandosAT
  if(Sim900Serial.available())//Verificamos si hay datos disponibles desde el SIM900
  Serial.write(Sim900Serial.read());//Escribir datos

  //SD
  comandosSD();//Llama a la función comandosSD

  //BATERIA
  float voltajeBat =  (float)25*analogRead(A8)/1023;
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;
  int temp = rtc.getTemperature();
  Serial.print("Voltaje Bateria =  ");
  Serial.println(voltajeBat);
  Serial.print("Voltaje LVDT =  ");
  Serial.println(voltajeLVDT);
  Serial.print("Temperatura interna =  ");
  Serial.println(temp);

  //SMS BATERIA BAJA Y ALTA TEMPERATURA INTERNA
  if(voltajeBat<11.05){
    comandosSMSbateria();//Llama a la función comandosSMSbateria
  }

  //if(voltajeLVDT<11.05){
   // comandosSMSlvdt();//Llama a la función comandosSMSlvdt
    //}

  if(temp>40){
    comandosSMStemp();//Llama a la función comandosSMSlvdt
    }

  //LVDT
  digitalWrite(rele, HIGH);
  delay(500);

  //SLEEP
  Serial.println("Durmiendo...");
  snoozeLib.snooze(550000);//Snooze the CPU(1000=1segundo); 10 min=550000
  Serial.println("Despierto!!");
  digitalWrite(6,state);

  delay(1000);
}
```

Figura 10. Código función principal loop.

Una vez que ya se ha establecido función y se ha dado lugar al comienzo del código del programa principal, este comienza con tres líneas empleadas para resetear el microcontrolador pasados 6500000 milisegundos de funcionamiento del mismo. Este reseteo se realiza pasado ese tiempo porque en la fase de implementación se comprobó que cuando el microcontrolador llevaba más de 7000000 milisegundos funcionando se bloqueaba. Este bloqueo se debe a conforme pasa el tiempo va llenándose la memoria RAM del mismo, y pasado ese tiempo no quedaba suficiente memoria RAM libre para que este funcionara adecuadamente, véase apartado 2.4.1. del Anejo V para mayor información. Para solucionar este problema se decidió resetear el microcontrolador cada cierto tiempo antes de que vuelva a surgir dicho problema. Con todo ello, la primera de las tres líneas incluye el comando "if(time>6500000) {". Si el tiempo que lleva funcionando el microcontrolador es superior a 6500000 milisegundos (tiempo prudencial que evita que se bloquee el procesador) se ejecutará el código que se incluye a continuación entre llaves, el cual se encargará de resetear el microcontrolador solucionando el problema mencionado. En este caso, por tanto, se ejecutará el código de la sentencia condicional el cual consta en primer lugar el comando "wdt_enable(WDTO_15MS)" y después el comando "delay(20)". El primero de estos dos comandos configura y activa que en 15 milisegundos se resetee el microcontrolador. El segundo da un tiempo prudencial de 20 milisegundos para que no se desactive el comando anterior y se lleve a cabo el reseteo con éxito. En caso de que el tiempo de funcionamiento sea menor no se ejecutará el código de la sentencia condicional y se continuará ejecutando el sketch sin resetear el microcontrolador. Véase Figura 9

Después se pasa a activar la alimentación del dendrómetro mediante el accionamiento del relé que tiene instalado conectado al pin trece. Para ello se emplea el comando "digitalWrite(rele, LOW)" de este modo se pone la constante relé correspondiente al pin 13 en LOW con lo que el relé cierra el circuito de alimentación permitiendo que el sensor funcione. Después, se dejan 500 milisegundos de espera usando un delay. Véase Figura 11.

```
//LVDT
digitalWrite(rele, LOW);
delay(500);
```

Figura 11. Activación de la alimentación del dendrómetro.

Una vez encendido el sensor se procede a la toma y al almacenamiento de los datos que de él se toman en la nube. Concretamente en Internet of Things (IoT) a través de la plataforma Thingspeak. Esta acción conlleva el uso de un código complejo y extenso, a la vez que sensible, lo que facilita la aparición de fallos en su ejecución. Si ello se suma al código principal se obtiene un programa aún más extenso y complejo, lo que dificulta su seguimiento y claridad. Por ello, el código necesario para el almacenamiento de los datos se ejecuta aparte, en una función complementaria. De esta forma, se obtienen dos códigos más sencillos y menos extensos que presentarán un menor riesgo de fallo, obteniéndose códigos más fiables. Por ello, y con el fin de aumentar la fiabilidad del código desarrollado, se opta por realizar esta segunda opción.

Para ello se hace uso en la función principal de una función complementaria llamada comandosAT, la cual se encarga del almacenamiento de datos en IoT. Dicha función recibe este nombre debido a que su código se compone principalmente de comandos de tipo AT. Para llevar a cabo este propósito sencillamente se llama en la función principal a la secundaria mediante el comando "comandosAT()" el cual hace que se ejecute dicha función. Tras este comando, y continuando en la función principal, se verifica si se dispone de datos en el puerto serie desde

el SIM900 con el uso del comando “if(Sim900Serial.available())”. Además, con el comando “Serial.write(Sim900Serial.read())” estos datos se escriben en el puerto serie. Comentar que la sentencia “sim900Serial.” hace que se active la comunicación por puerto serie con el módulo GSM/GPRS y que en esta se realice la acción indicada después del punto. Véase Figura 12.

```
//GSM
comandosAT();//Llama a la función comandosAT
if(Sim900Serial.available())//Verificamos si hay datos disponibles desde el SIM900
Serial.write(Sim900Serial.read());//Escribir datos

//SD
comandosSD();//Llama a la función comandosSD
```

Figura 12. Llamada a la función "comandosAT", verificación de conexión con modulo GSM/GPRS y llamada a la función”comandosSD.

Después de realizar el almacenamiento en IoT se procede a realizar el almacenamiento de los datos en la tarjeta microSD. De nuevo surge el mismo problema que con el código de almacenamiento en IoT, el código necesario para realizar el almacenamiento en la tarjeta microSD es complejo y extenso. Para solucionar este problema de nuevo se opta por hacer uso de una función complementaria en la función principal a fin de simplificar los códigos y aumentar la fiabilidad de funcionamiento de los mismos. La función complementaria empleada se llama comandosSD. Para hacer uso de la misma, sencillamente se emplea el comando ”comandosSD()” en la función principal. Dicho comando llama a la función complementaria comandosSD haciendo que esta se ejecute y como posteriormente se explica, guarde los datos en la tarjeta microSD. Véase Figura 12.

Seguidamente se establecen diversas variables que posteriormente serán utilizadas en el monitoreo del estado de los equipos y enviar alertas en caso necesario. Concretamente se establecen dos variables de tipo float para almacenar el voltaje de las baterías de los equipos, y un tipo int, para la temperatura interna del equipo de monitoreo. Véase Figura 13.

```
//BATERIA
float voltajeBat =  (float)25*analogRead(A8)/1023;
float voltajeLVDT =  (float)25*analogRead(A10)/1023;
int temp = rtc.getTemperature();
Serial.print("Voltaje Bateria =  ");
Serial.println(voltajeBat);
Serial.print("Voltaje LVDT =  ");
Serial.println(voltajeLVDT);
Serial.print("Temperatura interna =  ");
Serial.println(temp);
```

Figura 13. Establecimiento e impresión por monitor serie de las variables de voltaje de baterías y temperatura.

La primera variable se establece con el comando “float voltajeBat = (float)25*analogRead(A8)/1023” y almacena el valor del voltaje de la batería del equipo de monitoreo. Como se puede ver, a este se le llama voltajeBat. El valor del mismo se obtiene a partir de la lectura del pinA8, en el cual está conectado el sensor de voltaje, tras su acondicionamiento. Para ello se multiplica el valor leído por 25 y se divide por 1023. Esto es debido a que, por defecto, el microcontrolador muestra la señal leída en bytes. Por lo que la señal mostrada tendrá un valor entre 0 y 1023, correspondiendo un valor de 0 para un voltaje

de 0V y 1023 para 24V. Al multiplicar este valor por 25 y dividirlo por 1023 se transforma, la señal adquirida en bytes a la misma en voltios. El valor de 25 se obtuvo al calibrar el sistema por comparación de la señal obtenida con la lectura de un multímetro.

Para la segunda variable se procede de forma similar, aunque en este caso, la variable se llama voltajeLVDT por corresponder al voltaje de la batería del LVDT. En este caso el pin a leer será el A10 en el cual se encuentra conectado el sensor de voltaje de la batería del LVDT. Con todo ello el comando resultante es "float voltajeLVDT = (float)25*analogRead(A10)/1023".

El tercer parámetro medido corresponde a la temperatura interna del equipo de monitoreo. Para almacenar su valor, se crea una variable de tipo int a la que se le ha llamado temp. Con la sentencia "int temp = rtc.getTemperature()"se obtiene la temperatura por medio dela función "rtc.getTemperature()" de una de las librerías de control del módulo DS3231.

Estas variables son impresas en el monitor serie (en caso de que el microcontrolador esté conectado al pc) mediante los siguientes comandos Serial.print() y Serial.println() e incluyendo en el interior del paréntesis el mensaje a imprimir en el mismo.

En este punto, se describe la parte del código principal dedicada al envío de avisos a través de mensajes SMS. Estos mensajes pueden programarse en base a múltiples motivos, en este caso, a modo de ejemplo, se ha previsto enviar mensajes debido a un estado de batería baja y/o por exceso de temperatura interna del equipo.

Para ello, nuevamente se hace uso de funciones complementarias, a fin de simplificar el código principal y mejorar la fiabilidad del mismo. Concretamente se hace uso de tres funciones complementarias. La primera de ellas se encargará de enviar un SMS de aviso por batería baja en el equipo de monitoreo y se denomina comandosSMSbateria. La segunda también se encargará de dar aviso a través de SMS, esta vez por batería baja en el sistema de alimentación del sensor LVDT, la función es llamada comandosSMSlvdt. Por último, se emplea la función complementaria comandosSMStemp que se encargara de enviar un aviso a través de SMS en caso de que la temperatura interna del equipo de monitoreo sea elevada, concretamente mayor a 40°C.

El código empleado en la función principal para hacer uso de estas tres funciones complementarias es similar en los tres casos. En todos ellos, se hace uso de una sentencia condicional if, en la que, de cumplirse su condición, se activa la función complementaria que envía el aviso deseado para ese caso. Véase Figura 14.

```
//SMS BATERIA BAJA Y ALTA TEMPERATURA INTERNA
if(voltajeBat<11.05){
  comandosSMSbateria();//Llama a la función comandosSMSbateria
}

//if(voltajeLVDT<11.05){
 // comandosSMSlvdt();//Llama a la función comandosSMSlvdt
  //}

if(temp>40){
  comandosSMStemp();//Llama a la función comandosSMSlvdt
  }
```

Figura 14. Sentencias condicionales para activación de funciones de aviso.

En el caso de que la batería del equipo de monitoreo tenga un voltaje bajo, la sentencia condicional se corresponde con el comando "if(voltajeBat<11.05){". En él la condición establecida es que el voltaje de la batería del equipo de monitoreo (valor almacenado en la variable voltajeBat) tenga un voltaje menor a 11,05V. Este valor de voltaje se ha establecido de tal forma que sea un voltaje bastante bajo pero suficiente para poder actuar para solucionar el problema con tiempo suficiente. Si se cumple esta condición se pasará a ejecutar el comando "comandosSMSbateria()" de la siguiente línea el cual llama y ejecuta la función comandosSMSbateria, que enviará el mensaje de alerta pertinente. De no cumplirse la condición no se ejecutará este comando y se pasará al siguiente, ya fuera de la sentencia condicional.

A continuación, se encuentra el segundo caso, batería del sensor LVDT baja, para el cual se emplea la sentencia condicional if "if(voltajeLVDT<11.05){" . Este comando es igual al del caso anterior con la diferencia de que en este caso se trata del voltaje del sensor (anteriormente establecido como voltajeLVDT) y no el del equipo de monitoreo. Con este comando se comprueba si el voltaje de la batería del sensor LVDT es inferior a 11,05V y en ese caso se ejecutará el comando "comandosSMSlvdt()" de la siguiente línea de código. El comando ejecuta la función comandosSMSlvdt que enviará el mensaje de alerta pertinente. De no cumplirse la condición, no se ejecutará este comando y se pasará al siguiente, ya fuera de la sentencia condicional.

Después se encuentra el tercer caso, temperatura interna del equipo de monitoreo elevada, para el que se emplea la sentencia condicional if "if(temp>40){". Con este comando se establece la condición de que si la temperatura leída en el interior del equipo de monitoreo (anteriormente establecido como temp) es superior a 40°C se ejecute el comando "comandosSMStemp()" de la siguiente línea. Este comando llamara y ejecutara la función comandosSMStemp que enviara el mensaje de alerta pertinente. De no cumplirse la condición no se ejecutará el mismo y se pasará al siguiente ya fuera de la sentencia condicional.

Puesto que ya no se van a tomar más medidas del sensor LVDT hasta el siguiente ciclo, se pasa a desactivar su alimentación para ahorrar carga de la batería. Para ello se emplea el comando "digitalWrite(rele, HIGH)" de este modo se pone la constante relé, correspondiente al pin 13, en HIGH con lo que el relé abre el circuito de alimentación impidiendo que el sensor funcione y ahorrando con ello energía. Después, se dejan 500 milisegundos de espera usando un delay. Véase Figura 15.

```
//LVDT
digitalWrite(rele, HIGH);
delay(500);
```

Figura 15.Desactivación de la alimentación del dendrómetro.

Por último, se procede al cambio del modo de funcionamiento del microcontrolador de activo a bajo consumo con el fin de ahorrar energía. Para ello, primero se da aviso al monitor serie (en caso de estar conectado al pc) mediante el comando "Serial.println("Durmiendo...")" con el que se imprime en el monitor serie el mensaje "Durmiendo…". El siguiente comando, "snoozeLib.snooze(550000)" es una función de la librería SnoozeLib.h con la que el microcontrolador entra en modo bajo consumo durante 550000 milisegundos, aproximadamente nueve minutos. Con ello conseguiremos tomar los datos cada 10 minutos puesto que el microcontrolador está en modo bajo consumo 9 minutos y en realizar la toma de datos y el almacenamiento de los mismos se tarda un minuto. Después se procede a cambiar el

modo del microcontrolador a activo para posteriormente volver a comenzar a ejecutar el código principal desde el principio. Para ello se cambia el estado del pin 6 a state con lo que pasa del estado low a high. Esto conlleva que se active la interrupción anteriormente explicada en el apartado de setup, que hace que se ejecute la función wakeup que activa el microcontrolador como posteriormente se explicara. Después se deja de ejecutar dicha interrupción y se continúa ejecutando el programa principal. Ya de vuelta al código principal queda el ultimo comando, que es un delay de un segundo para dar tiempo suficiente para que se encienda correctamente el equipo de monitoreo. Véase Figura 16.

```
  //SLEEP
  Serial.println("Durmiendo...");
  snoozeLib.snooze(550000);//Snooze the CPU(1000=1segundo); 10 min=550000
  Serial.println("Despierto!!");
  digitalWrite(6,state);

  delay(1000);
}
```

Figura 16. Activación del modo bajo consumo del microcontrolador.

Una vez ejecutado este último comando se cierra la función principal con una llave de cierre. Hecho esto y al tratarse de la función loop que se ejecuta en bucle se volverá al principio de la misma para de nuevo ejecutar todo el programa.

1.3.2. Códigos funciones complementarias

En este apartado se encuentran recogidas todas las funciones complementarias a la función principal loop que bien directa o indirectamente han sido empleadas por esta.

1.3.2.1. Código función complementaria comandosAT

En este apartado se va a presentar y comentar el código empleado en la función complementaria comandosAT. Esta función es empleada por la función principal loop para realizar el almacenamiento en IoT de todos los datos deseados. En este caso, y a modo de ejemplo, se almacenarán la señal del sensor LVDT, la temperatura interna, voltaje de la batería del equipo de monitorización, voltaje de la batería del sensor LVDT, voltaje ofrecido por el módulo solar, tiempo de funcionamiento del microcontrolador desde su último inicio y memoria RAM libre. Evidentemente, de la misma forma es posible almacenar cuantas señales se desee. Esta función se encuentra codificada a continuación de la función principal y emplea en gran medida comandos AT o comandos Hayes. Esto se debe a que el módulo Shield Arduino SIM900 opera con estos comandos, permitiendo comunicar al microcontrolador con dicho módulo de la forma más sencilla posible. De no ser por estos comandos dicha comunicación resultaría ser mucho más compleja. Estos comandos son un lenguaje desarrollado por la compañía Hayes Communications y que en el pasado prácticamente se convirtieron en un estándar abierto de comandos para configurar y parametrizar módems hasta que cada fabricante desarrolló los suyos propios.

A continuación, en la Figura 17 se muestra el código empleado en la función complementaria comandosAT. Posteriormente se lleva a cabo la explicación detallada del mismo.

```
void comandosAT(){

  //lvdt
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor1 += analogRead(sensorPin1);
  delay(10);
  }
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor2 += analogRead(sensorPin2);
  delay(10);
  }
  sensor1 /= 10;
  sensor1 = map(sensor1,0,1023,8536,869);
  sensor2 /= 10;
  sensor2 = map(sensor2,0,1023,8544,16211);

  //Tª RTC
      int temp = rtc.getTemperature();

  //BATERIA
  float voltajeBat =   (float)25*analogRead(A8)/1023;
  float voltajePanSol =   (float)25*analogRead(A9)/1023;
  float voltajeLVDT =   (float)25*analogRead(A10)/1023;

  int sram = freeMemory();

  //GSM
  Sim900Serial.println("AT");//Comprueba estado del módulo
  delay(2000);
  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);
  Sim900Serial.println("AT+CREG?");// estado de conexión a la red
  delay(2000);
  Sim900Serial.println("CGATT?");//Si está conectado a internet o no
  delay(2000);
  Sim900Serial.println("CIPSHUT");//cerrará el contexto GPRS PDP
  delay(2000);
  Sim900Serial.println("AT+CIPSTATUS");//Consultar el estado actual de la conexión
  delay(2000);
  Sim900Serial.println("AT+CIPMUX=0");//comando configura el dispositivo para una conexión
                                      //IP única o múltiple 0=única
  delay(3000);
  mostrarDatosSeriales();
  //Comando configura el APN, nombre usuario y contraseña.
  Sim900Serial.println("AT+CSTT=\"internet\",\"\",\"\"");
  delay(1000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIICR");//REALIZAR UNA CONEXIÓN INALÁMBRICA CON GPRS O CSD
  delay(3000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIFSR");// Obtenemos nuestra IP local
  delay(2000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSPRT=0");//Establece un indicador '>' al enviar datos
  sensor1;
  sensor2;
  temp;
  voltajeBat;
  voltajePanSol;
  voltajeLVDT;
  time;
  sram;
  delay(3000);
  mostrarDatosSeriales();
```

```
    Sim900Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
    //Indicamos el tipo de conexión, url o dirección IP y puerto al que realizamos la conexión
    delay(6000);
    mostrarDatosSeriales();
    Sim900Serial.println("AT+CIPSEND");//ENVÍA DATOS A TRAVÉS DE una CONEXIÓN TCP O UDP
    delay(4000);
    mostrarDatosSeriales();

    if(sensor2>8544) {
      String datos="GET https://api.thingspeak.com/update?api_key=BPTUlJL7TURYPBY5&field1=0"
      + String(sensor2)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
      + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
      + "&field6=0" + String(time)+ "&field7=0" + String(sram);

      Sim900Serial.println(datos);//Envía datos al servidor remoto
      delay(4000);
      mostrarDatosSeriales();
      delay(1000);
        }
    else {
      String datos="GET https://api.thingspeak.com/update?api_key=BPTUlJL7TURYPBY5&field1=0"
      + String(sensor1)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
      + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
      + "&field6=0" + String(time)+ "&field7=0" + String(sram);

      Sim900Serial.println(datos);//Envía datos al servidor remoto
      delay(4000);
      mostrarDatosSeriales();
      delay(1000);
       }

  sensor1 =0;
  sensor2 =0;
  Sim900Serial.println((char)26);
  delay(5000);
  //Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
  //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

}
```

Figura 17.Código función complementaria comandosAT.

El código empleado para esta función complementaria comienza, al igual que en las funciones anteriores, estableciendo la función en sí misma, Véase Figura 18. En este caso se trata de la función complementaria comandosAT. Dicha función comandosAT se establece mediante el comando “void comandosAT(){“. La primera parte del comando la establece y después, con la llave de apertura, se da pie al comienzo del código de la misma. Con ello el código se desarrolla en las líneas posteriores finalizando antes de la llave de cierre de la función y siendo ejecutado una única vez cuando el programa principal del software así lo demanda (llama a la función).

Una vez se ha establecido la función y se ha dado lugar el comienzo de su código este empieza obteniendo el valor de la señal del sensor LVDT. Para ello no se realiza sencillamente su lectura,

si no que esta se filtra por medio de software con lo que se obtiene un valor de la señal del sensor LVDT más fiable y suavizada eliminando parte del ruido de la lectura de la misma. Para conseguirlo se realizan diez lecturas de forma consecutiva de tal forma que se van sumando conforme se van leyendo. La suma de estas diez lecturas se divide entre diez con lo que se obtendrá que el valor de la señal del sensor es igual a la media de las diez lecturas realizadas con el sensor. Para llevar esto a cabo con el código software se utiliza un bucle tipo for con el que se obtendrán diez valores de señal del sensor lvdt. Posteriormente se calcula la media del mismo directamente. Además de este filtrado con el que se obtiene un valor de señal de mejor calidad que un único valor, es necesario mapear dicho valor. 'Mapear' consiste en transformar el valor de la señal en un valor de posición. Para ello fue necesario realizar la calibración del sistema (apartado 1.3.2 del Anejo V). Con ello se convierte el valor obtenido en bytes a un valor que se corresponde con una posición del núcleo en el sensor. Estas, van desde 869, cuando el núcleo se encuentra en un extremo, hasta 16211, cuando el núcleo está en el otro, véase apartado 1.3.2. del Anejo V donde se obtiene la ecuación de calibración con la que se obtienen estos valores y Figura 18 con el código empleado para el filtrado.

Para mostrar cómo se aplica este filtrado, se procede a su explicación con el código de lectura de la primera mitad del sensor LVDT. Analizando detalladamente el bucle for empleado, tenemos en primer lugar el comando "for (int i = 0 ; i < 10 ; i++) { " . En este primero se define directamente la variable índice como int i = 0, esta variable índice es local y solo es accesible dentro del bucle for. Seguidamente, se define la condición que hace que se ejecute el código del bucle tantas veces como sea necesario mientras se cumpla dicha condición. En este caso, la condición es i < 10, por lo que el código se repetirá mientras que la variable índice sea menor a 10, como hemos empezado con el valor 0 para la variable índice esta condición solo se cumplirá 10 veces. Por último, se encuentra el incrementador de la variable índice. Cada vez que se evalúa la condición y se ejecuta el código se incrementa la variable índice, en este caso el incrementador se escribe como i++, que es la forma abreviada de i = i + 1. Tras este primer comando comienza el código interno del bucle que se ejecutara diez veces, como se ha establecido en el comando anterior. Dicho código se corresponde con el comando "sensor1 += analogRead(sensorPin1)" y después con un delay de 10 milisegundos. Con este comando, a la variable sensor1(declarada anteriormente) se le suma el valor de lectura del sensor. Al realizar esta acción diez veces seguidas se va incrementando cada vez el valor de la variable con el valor leído en el sensor, con lo que al finalizar el bucle se tiene que la variable sensor1 es igual a la suma de las diez lecturas realizadas.

```
void comandosAT(){

  //lvdt
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor1 += analogRead(sensorPin1);
  delay(10);
  }
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor2 += analogRead(sensorPin2);
  delay(10);
  }
```

Figura 18. Establecimiento de función loop y lectura de la señal del dendrómetro.

Puesto que ya se tiene la suma de las diez lecturas del sensor, el siguiente paso es dividir este valor entre diez para obtener el valor medio leído. Para ello se emplea el comando "sensor1 /= 10". Véase Figura 19.

Como se ha explicado anteriormente este valor obtenido es necesario mapaeralo para que corresponda con una posición de núcleo en el sensor. Para ello se emplea el comando "sensor1=map(sensor1,0,1023,8536,869)". Con este comando se convierte el rango de 0 a 1023 que se ha adquirido al rango de 8536 a 869. Con ello, por ejemplo, cuando la lectura del sensor sea 0 es porque la posición del núcleo está en 8536micras y cuando sea 1023 será 869 micras. Véase Figura 19.

Con todo ello se habrá obtenido un valor fiable de la posición del núcleo en la primera mitad del sensor L.V.D.T.

Para obtener un valor fiable de la posición del núcleo en la segunda mitad del sensor se emplea el mismo código, cambiando la variable sensor1 por la variable sensor2 y la lectura en el código del bucle se realiza en el pin correspondiente a la otra mitad, el pinA2, con el comando "sensor2 += analogRead(sensorPin2)". Además, el mapeo cambiara sus valores puesto que para este caso el núcleo se encuentra posicionado en la otra mitad quedando el comando de este del siguiente modo "sensor2=map(sensor2,0,1023,8544,16211)". Con ello, por ejemplo, cuando la lectura del sensor sea 0 se obtendrá que la posición del núcleo está en 8544micras y cuando sea 1023 será 16211 micras.

Con todo ello se obtiene con las dos variables la posición del núcleo que se encontrara en el rango de 869 micras a 16211 micras. Dividiéndose este en dos mitades, la primera identificada con la variable sensor1 y la segunda con la variable sensor2.

```
sensor1 /= 10;
sensor1 = map(sensor1,0,1023,8536,869);
sensor2 /= 10;
sensor2 = map(sensor2,0,1023,8544,16211);
```

Figura 19. Acondicionamiento de la señal del dendrómetro por software.

Seguidamente, y continuando con la obtención de datos para su posterior almacenamiento en IoT, se obtienen la temperatura interna del equipo de monitoreo, el voltaje de la batería del mismo, el voltaje de la batería del sensor y el voltaje ofrecido por la placa solar. Estos se obtienen con los mismos comandos explicados anteriormente en la función loop. Voy a destacar el comando empleado para obtener el voltaje de la placa solar puesto que este no se ha mostrado, el cual es "float voltajePanSol = (float)25*analogRead(A9)/1023" y cuyo funcionamiento es igual al de obtención del voltaje de las baterías. Destacar también el código empleado para obtener la memoria RAM disponible. Para ello se establece la variable de tipo int llamada sram. Esta variable se establece con el comando "int sram = freeMemory()". Como se puede ver esta es igual a la función "freeMemory()"de la librería MemoryFree.h con la que se obtiene el valor de la memoria RAM libre. Véase Figura 20.

```
//Tª RTC
    int temp = rtc.getTemperature();

//BATERIA
float voltajeBat =   (float)25*analogRead(A8)/1023;
float voltajePanSol =   (float)25*analogRead(A9)/1023;
float voltajeLVDT =   (float)25*analogRead(A10)/1023;

int sram = freeMemory();
```

Figura 20. Establecimiento de las variables temperatura, voltaje de baterías y memoria RAM libre.

En este punto la función ya dispone de todos los datos a enviar. Por tanto, se comienza el proceso de conexión del microcontrolador con el módulo sim900 y con ello la conexión con IoT y el almacenamiento de los datos en la plataforma Thinghspeak. Para ello se hace uso de comandos AT. Para la comunicación y envío de cada uno de estos comandos por parte del microcontrolador al módulo GSM/GPRS se hace uso del comando "Sim900Serial.println("")". Este comando hace que el microcontrolador envíe el mensaje o comando que se encuentre entre las comillas dentro de su paréntesis al módulo GSM/GPRS. Siempre tras el envío de un comando de este tipo se realizará una espera para dar tiempo al módulo sim900 a procesar la información y dar una respuesta al mismo.

La primera acción a llevar a cabo para conseguir el objetivo comentado en el anterior párrafo es comprobar que hay una buena conexión entre el microcontrolador y el módulo, así como configurarlo para su conexión a internet. Véase la Figura 21.

```
//GSM
Sim900Serial.println("AT");//Comprueba estado del módulo
delay(2000);
Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
delay(2000);
Sim900Serial.println("AT+CREG?");// estado de conexión a la red
delay(2000);
Sim900Serial.println("CGATT?");//Si está conectado a internet o no
delay(2000);
Sim900Serial.println("CIPSHUT");//cerrará el contexto GPRS PDP
delay(2000);
Sim900Serial.println("AT+CIPSTATUS");//Consultar el estado actual de la conexión
delay(2000);
Sim900Serial.println("AT+CIPMUX=0");//comando configura el dispositivo para una conexión
                                    //IP única o múltiple 0=única
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CSTT=\"internet\",\"\",\"\"");
//comando configura el APN, nombre de usuario y contraseña.
delay(1000);
mostrarDatosSeriales();
```

Figura 21.Conexión del microcontrolador con módulo GSM/GPRS y configuración del mismo.

Para ello, con la primera línea de código se comprueba la conexión y el estado del módulo. Para ello el microcontrolador envía el comando "AT" al módulo, del modo anteriormente indicado. Si todo está correcto el módulo responderá con un OK.

Tras dos segundos de espera, para que el módulo responda al anterior código, el microcontrolador introduce el pin de la tarjeta sim enviando el comando "AT+CPIN?" al shield GSM/GPRS. De nuevo se esperan dos segundos para que el módulo procese el comando.

El siguiente comando enviado al módulo sim900 es "AT+CREG?". Con él que se comprueba el estado de conexión a la red. De nuevo se esperan dos segundos a que el módulo responda con un OK. Además, se comprueba que se ha conectado el módulo adecuadamente a internet con el comando "CGATT?" y de nuevo se espera su respuesta dos segundos. Tras estas comprobaciones se cierra el contexto GPRS PDP con el envío del comando "CIPSHUT" al módulo GSM/GPRS y se espera dos segundos.

Seguidamente se comprueba el estado actual de la conexión con el envío del comando "AT+CIPSTATUS" y se espera dos segundos, y se configura el módulo shield para que obtenga una conexión IP única con el comando "AT+CIPMUX=0" tras lo cual se esperan en este caso tres segundos.

Para ver la respuesta del módulo a todos estos comandos se hace uso de una función complementaria a esta, llamada mostrarDatosSeriales , véase apartado 1.3.2.2.Código función complementaria mostrarDatosSeriales. Después se envía el comando "AT+CSTT=\"internet\",\"\",\"\"". En él, a partir del signo igual y separado por comas, se configura el APN, se indica el nombre de usuario y la contraseña. Tras este se espera un segundo y de nuevo se llama a la función mostrarDatosSeriales para mostrar la respuesta a dicho comando.

Con estos comandos quedaría realizada toda la configuración del módulo y se pasa a conectar el módulo GSM/GPRS a internet y a obtener una dirección IP local. Véase Figura 22.

```
Sim900Serial.println("AT+CIICR");//REALIZAR UNA CONEXIÓN INALÁMBRICA CON GPRS O CSD
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIFSR");// Obtenemos nuestra IP local
delay(2000);
mostrarDatosSeriales();
```

Figura 22. Conexión del módulo GSM/GPRS a internet y comprobación de esta conexión.

Para ello, primeramente se activa el perfil de datos inalámbrico con el comando "AT+CIICR", seguido de tres segundos de espera y de la llamada a la función mostrarDatosSeriales. Luego se obtiene la dirección IP local enviando al módulo el comando"AT+CIFSR" y nuevamente se realiza una espera de dos segundos y se llama a la función mostrarDatosSeriales con la que ver la respuesta del módulo.

En este punto ya se ha comprobado la conexión del microcontrolador con el módulo shield, se ha configurado el módulo para su conexión a internet, se ha conectado obteniendo una dirección IP y se ha comprobado dicha conexión a la red. El siguiente paso ya es el envío de datos, para ello primero se pasan los datos desde el microcontrolador al ódulo GSM/GPRS, después se conecta el módulo con la plataforma IP y estos se le envían. Finalmente se cerrará la conexión.

Por tanto, en el siguiente paso se envía al módulo el comando "AT+CIPSPRT=0" con el que se establece que a continuación se le van a enviar datos desde el microcontrolador, y por ello se le advierte de que va a recibir datos. Los datos que recibirá serán los datos que hemos obtenido al

principio de esta función y que queremos que se envíen a la plataforma IoT. Para ello en las siguientes líneas se nombran las variables de cada dato de una en una. Después se dan tres segundos de espera para que el módulo reciba los datos y se llama a la función mostrarDatosSeriales para ver si han sido subidos correctamente. Con ello ya se han enviado los datos desde el microcontrolador al módulo GSM/GPRS. Véase la Figura 23.

```
Sim900Serial.println("AT+CIPSPRT=0");//Establece un indicador '>' al enviar datos
sensor1;
sensor2;
temp;
voltajeBat;
voltajePanSol;
voltajeLVDT;
time;
sram;
delay(3000);
mostrarDatosSeriales();
```

Figura 23. Transferencia de datos del microcontrolador al módulo GSM/GPRS.

A continuación vamos a conectar el módulo con la plataforma Thingspeak. Para ello se indica el tipo de conexión a realizar, en este caso TCP; el url a la que se realiza la conexión, en este caso el url de la plataforma Thingspeak "api.thingspeak.com"; y el puerto al que se va a realizar dicha conexión, el 80. Esta acción se lleva a cabo con el comando "AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"".Después se esperan seis segundos para procesar toda la información y se llama a la función mostrarDatosSeriales para ver la respuesta a la misma. Véase Figura 24.

```
Sim900Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
//Indicamos el tipo de conexión, url o dirección IP y puerto al que realizamos la conexión
delay(6000);
mostrarDatosSeriales();
```

Figura 24. Conexión del módulo GSM/GPRS con la plataforma Thingspeak mediante internet.

Posteriormente se realiza el envío de datos. Este consta de tres pasos principales, que son el preestablecimiento de envío de datos, la preparación de los datos y el envío de los mismos.

Para el preestablecimiento de envío de datos se emplea primero el comando "AT+CIPSEND" con el que se establece que se van a enviar los datos a través de la conexión TCP establecida anteriormente, se espera 4 segundos y se llama a la función mostrarDatosSeriales que muestra la respuesta del módulo. Véase Figura 25.

```
Sim900Serial.println("AT+CIPSEND");//ENVÍA DATOS A TRAVÉS DE una CONEXIÓN TCP O UDP
delay(4000);
mostrarDatosSeriales();
```

Figura 25. Preestablecimiento de envío de datos a plataforma Thingspeak.

El siguiente paso es la preparación de los datos. En ella se va a indicar que datos se han de subir. Para ello estos se van a reunir en una cadena de caracteres tipo String la cual se llamara datos. Puesto que dependiendo de en qué lado del sensor se encuentre el núcleo, el dato de posición del LVDT a almacenar será el de la variable sensor 1 o sensor2 es necesario establecer una condición que haga que se almacene o una o la otra variable en función de sus valores. Para conseguirlo se hace uso de una sentencia condicional de tipo if else. Con esta se hace que, si el

valor del sensor2 es mayor al valor mínimo que esta puede adquirir, en concreto 8544micras, se almacenara el valor de esta variable como valor del sensor L.V.D.T. En caso contrario se almacenará el valor de la variable sensor1 como valor del sensor L.V.D.T. El comando empleado para ello es el comando "if(sensor2>8544) {" para el primer caso y para el segundo el comando es "else {". Con ello se habrá diferenciado cuando hay que subir como valor del sensor la variable sensor1 de la variable sensor2. A partir de aquí, cuando se tenga que enviar el dato de la variable sensor2 se ejecutara el código de envío que se encuentra entre las llaves del comando"if(sensor2>8544) {". Y cuando se envíe la variable sensor1 s ejecutara el código de envío que se encuentra entre las llaves del comando "else {". En ambos casos el código es el mismo, cambiando solamente la variable sensor2 por sensor1 o viceversa. Sea del modo que sea, en los dos siguientes párrafos se procede a explicar dicho código. Véase Figura 26.

```
if(sensor2>8544) {
  String datos="GET https://api.thingspeak.com/update?api_key=BPTU1JL7TURYPBY5&field1=0"
  + String(sensor2)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
  + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
  + "&field6=0" + String(time)+ "&field7=0" + String(sram);

  Sim900Serial.println(datos);//Envía datos al servidor remoto
  delay(4000);
  mostrarDatosSeriales();
  delay(1000);
    }
else {
  String datos="GET https://api.thingspeak.com/update?api_key=BPTU1JL7TURYPBY5&field1=0"
  + String(sensor1)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
  + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
  + "&field6=0" + String(time)+ "&field7=0" + String(sram);

  Sim900Serial.println(datos);//Envía datos al servidor remoto
  delay(4000);
  mostrarDatosSeriales();
  delay(1000);
   }
```

Figura 26. Preparación y envío de datos a la plataforma Thingspeak.

Una vez seleccionados los datos a enviar y dentro de la llave de la sentencia condicional correspondiente, se procede a encadenar todos los datos que se pretende enviar en un String (cadena de caracteres). Este String se declara llamándolo datos y se hace igual a la dirección exacta del servidor donde se guardan los datos, junto a la identificación del campo donde se guarda el primero de los datos; más el primer dato, que es el valor del sensor(según el caso sensor1 o sensor2) como valor tipo String; más la identificación del campo donde se guarda el segundo dato(temperatura interna);más el segundo dato a subir, que es el valor de la variable temp como valor String y así sucesivamente hasta poner todos los datos quedando el comando como el de la Figura 27.

```
String datos="GET https://api.thingspeak.com/update?api_key=BPTU1JL7TURYPBY5&field1=0" + String(sensor2)
+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)+ "&field4=0" + String(voltajePanSol)
+ "&field5=0" + String(voltajeLVDT)+ "&field6=0" + String(time)+ "&field7=0" + String(sram);
```

Figura 27. Detalle del encadenado de datos.

Con los datos encadenados en el String "datos" se procede a subirlo a la plataforma IoT Thingspeak. Para ello se utiliza el comando "Sim900Serial.println(datos)" que hace que el

módulo envíe el string datos a la dirección indicada en el mismo. A continuación, se espera 4 segundos y se llama a la función mostrarDatosSeriales que hará que el módulo muestre la respuesta del comando enviado. A continuación, se espera un segundo más y se cierra la llave de la sentencia condicional pertinente con lo que se finaliza el código de envío de datos.

Ya fuera de las sentencias condicionales anteriores, se resetean las variables sensor1 y sensor2, para que se les pueda volver a dar valor en la próxima toma de datos. Ello se consigue con los comandos "sensor1 =0" y "sensor2 =0". De este modo ambas variables vuelven a tener valor cero con lo que se podrá volver a realizar la medida con filtrado explicada anteriormente sin que se produzcan errores por que se añadan los valores obtenidos en una toma de muestras a la siguiente. Véase Figura 28.

```
  sensor1 =0;
  sensor2 =0;
  Sim900Serial.println((char)26);
  delay(5000);
  //Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
  //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

}
```

Figura 28. Reseteo de las variables del sensor, comprobación de envío correcto de los datos y cierre de conexión.

Puesto que ya se han subido los datos se emplea el comando "Sim900Serial.println((char)26)" con el que el microcontrolador pide al módulo sim900 que le de una respuesta de la red para comprobar que se han subido los datos con éxito. Después se esperan cinco segundos y de nuevo se llama a la función mostrarDatosSeriales para que el módulo muestre la respuesta obtenida. Véase Figura 28.

Por último, se cierra la conexión, enviando al módulo el comando "AT+CIPSHUT". Se esperan cinco segundos y de nuevo se llama a la función mostrarDatosSeriales para ver la respuesta del módulo. Véase Figura 28.

1.3.2.2. Código función complementaria mostrarDatosSeriales

La función mostrarDatosSeriales no es empleada directamente en el código del programa principal de la función loop, si no que es empleada por la función complementaria comandosAT. Por tanto, esta función es accesoria a la función complementaria comandosAT. La función mostrarDatosSeriales es llamada en la función comandosAT para mostrar las respuestas producidas por el módulo GSM/GPRS, al enviarle ordenes o comandos de tipo AT, o Hayes desde el microcontrolador. A continuación, se muestra el código de la función mostrarDatosSeriales en la Figura 29 y posteriormente se profundiza en él con una explicación.

```
void mostrarDatosSeriales()//Muestra los datos que va entregando el sim900
{
  while(Sim900Serial.available()!=0)
  Serial.write(Sim900Serial.read());
}
```

Figura 29. Código función complementaria mostrarDatosSeriales

Esta función es muy sencilla y necesaria puesto que su tarea es realizada gran cantidad de veces en la función comandosAT. Con su utilización se simplifica y acorta la programación, con las ventajas que ello conlleva. De este modo, al crear esta función, el código de la función Comandos AT se simplifica haciéndose más sencillo, fácil de ejecutar y con ello más fiable. Así se evitan los problemas que ocasionan los códigos extensos y complejos como anteriormente se ha comentado.

Al igual que el código del resto de funciones, el de esta comienza con el establecimiento de la misma. Para ello se emplea el comando "void mostrarDatosSeriales()". Seguidamente se encuentra una llave de apertura la cual da pie al comienzo del código de la función. Este se desarrolla en las líneas siguientes, finalizando al llegar a la llave de cierre de la función. Este código se ejecutará cada vez que la función complementaria comandos AT llame a la función mostrarDatosSeriales.

Continuando con el código de la función, en un primer momento se encuentra el comando "while(Sim900Serial.available()!=0)". Se trata de una sentencia condicional tipo while, lo que quiere decir que mientras se cumpla la condición del comando se ejecutara el código siguiente, en este caso el comando "Serial.write(Sim900Serial.read())". Por tanto, el comando "while(Sim900Serial.available()!=0)" hace que mientras se cumpla la condición de que los datos disponibles del módulo sim900 sean distintos a cero se ejecute el comando "Serial.write(Sim900Serial.read())" el cual escribe en el monitor serie los datos que se leen en ese momento del módulo sim900. Después, con la llave de cierre, se finaliza el código de la función mostrarDatosSeriales.

1.3.2.3. Código función complementaria comandosSD

Esta sección está dedicada a la función complementaria comandosSD. Esta es empleada por la función principal en su código para guardar en una tarjeta microSD los datos obtenidos haciendo uso del módulo microSD. Al igual que en la función complementaria comandosAT se almacenarán la señal del sensor LVDT, la temperatura interna, voltaje de la batería del equipo de monitorización, voltaje de la batería del sensor LVDT y el voltaje ofrecido por el módulo solar.

En esta ocasión se justifica la necesidad de emplear esta función como complementaria de la principal debido a la gran extensión del código necesario para guardar los datos en la tarjeta microSD. De este modo, se evitan los problemas que puede acarrear el hecho de usar dicho código en la función principal, simplificando su programación y mejorando la fiabilidad del programa.

Seguidamente se muestra en la Figura 30 el código de la función comandosSD y se procede a su explicación.

```
void comandosSD(){

  //RELOJ
   DateTime now = rtc.now(); //get the current date-time
    float tempFloat = rtc.getTemperature(); //read registers and display the temperature
    rtc.convertTemperature();            //convert current temperature into registers

  //SD
  logFile = SD.open("datalog.txt", FILE_WRITE); // Abrir archivo y escribir valor

  //LVDT
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor1 += analogRead(sensorPin1);
  delay(10);
  }
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor2 += analogRead(sensorPin2);
  delay(10);
  }
  sensor1 /= 10;
  sensor1 = map(sensor1,0,1023,8536,869);
  sensor2 /= 10;
  sensor2 = map(sensor2,0,1023,8544,16211);

    //BATERIA
  float voltajeBat =  (float)25*analogRead(A8)/1023;
  float voltajePanSol =  (float)25*analogRead(A9)/1023;
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;

  if (logFile) {
      if(sensor2>8544) {
        // Escribir en SD:

        {logFile = SD.open("datalog.txt", FILE_WRITE);  //Abrir fichero y escribir valor
        logFile.print(weekDay[now.dayOfWeek()]);    // Send Day-of-Week
        logFile.print(" ; ");
        logFile.print(now.year(), DEC);    // Send date
        logFile.print('/');
        logFile.print(now.month(), DEC);
        logFile.print('/');
        logFile.print(now.date(), DEC);
        logFile.print(' ');
        logFile.print(" ; ");
        logFile.print(now.hour(), DEC);    // Send time
        logFile.print(':');
        logFile.print(now.minute(), DEC);
        logFile.print(':');
        logFile.print(now.second(), DEC);
        logFile.print(" ; ");
        logFile.print("InternalTemperature; ");    //Send temperature
        logFile.print(rtc.getTemperature());
        logFile.print(" ; ");
        logFile.print(" °C ; ");
        logFile.print("Sensor Value2(µ);"); // Eviar dato sensor
        logFile.print(sensor2);
        logFile.print(" ; ");
        logFile.print("Voltaje Bateria =;  ");
        logFile.print(voltajeBat);
        logFile.print(" ; ");
```

```
    logFile.print("Voltaje Panel Solar =;  ");
    logFile.print(voltajePanSol);
    logFile.print(" ; ");
    logFile.print("Voltaje LVDT =;  ");
    logFile.println(voltajeLVDT);
    logFile.close();}   //Cerrar escritura SD

    //Escribir en monitor serie:
    Serial.print(weekDay[now.dayOfWeek()]);   // Send Day-of-Week
    Serial.print(", ");
    Serial.print(now.year(), DEC);      // Send date
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.date(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);      // Send time
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    Serial.print("InternalTemperature; ");      //Send temperature
    Serial.print(rtc.getTemperature());
    Serial.println(" °C");
    Serial.print("Sensor Value2(µ); "); //Enviar dato sensor
    Serial.print(sensor2);
    Serial.print(";");
    Serial.print("Voltaje Bateria =  ");
    Serial.print(voltajeBat);
    Serial.print(";");
    Serial.print("Voltaje Panel Solar =  ");
    Serial.print(voltajePanSol);
    Serial.print(";");
    Serial.print("Voltaje LVDT =  ");
    Serial.print(voltajeLVDT);
    Serial.println();

    //delay(1000);
    }

  else {
    // Escribir en SD:
    {logFile = SD.open("datalog.txt", FILE_WRITE);  //Abrir fichero y escribir valor
    logFile.print(weekDay[now.dayOfWeek()]);   // Send Day-of-Week
    logFile.print(" ; ");
    logFile.print(now.year(), DEC);  // Send date
    logFile.print('/');
    logFile.print(now.month(), DEC);
    logFile.print('/');
    logFile.print(now.date(), DEC);
    logFile.print(' ');
    logFile.print(" ; ");
    logFile.print(now.hour(), DEC);   // Send time
    logFile.print(':');
    logFile.print(now.minute(), DEC);
    logFile.print(':');
    logFile.print(now.second(), DEC);
    logFile.print(" ; ");
    logFile.print("InternalTemperature; ");   //Send temperature
    logFile.print(rtc.getTemperature());
    logFile.print(" ; ");
```

```
        logFile.print(" °C ; ");
        logFile.print("Sensor Value1(µ);"); // Eviar dato sensor
        logFile.print(sensor1);
        logFile.print(" ; ");
        logFile.print("Voltaje Bateria =;  ");
        logFile.print(voltajeBat);
        logFile.print(" ; ");
        logFile.print("Voltaje Panel Solar =;  ");
        logFile.print(voltajePanSol);
        logFile.print(" ; ");
        logFile.print("Voltaje LVDT =;  ");
        logFile.println(voltajeLVDT);
        logFile.close();}    //Cerrar escritura SD

        //Escribir en monitor serie
        Serial.print(weekDay[now.dayOfWeek()]);
        Serial.print(", ");
        Serial.print(now.year(), DEC);
        Serial.print('/');
        Serial.print(now.month(), DEC);
        Serial.print('/');
        Serial.print(now.date(), DEC);
        Serial.print(' ');
        Serial.print(now.hour(), DEC);
        Serial.print(':');
        Serial.print(now.minute(), DEC);
        Serial.print(':');
        Serial.print(now.second(), DEC);
        Serial.println();
        Serial.print("Temperature: ");       //Send temperature
        Serial.print(rtc.getTemperature());
        Serial.println(" °C");
        Serial.print("Sensor Value1(µ): "); //Enviar dato sensor
        Serial.print(sensor1);
        Serial.print(";");
        Serial.print("Voltaje Bateria =  ");
        Serial.print(voltajeBat);
        Serial.print(";");
        Serial.print("Voltaje Panel Solar =  ");
        Serial.print(voltajePanSol);
        Serial.print(";");
        Serial.print("Voltaje LVDT =  ");
        Serial.print(voltajeLVDT);
        Serial.println();

      }
    sensor1 =0;
    sensor2 =0;
  }

  else {
    Serial.println("Error al abrir el archivo");
  }
}
```

Figura 30.Código función complementaria comandosSD.

El código de la función complementaria comandosSD comienza con su establecimiento. Para ello se hace uso del comando "void comandosSD(){". Con él se establece la función y se da comienzo al código de la misma mediante el uso de una llave de apertura. Con ello el código se desarrolla en las líneas posteriores finalizando antes de la llave de cierre de la función y siendo ejecutado una única vez cuando el programa principal del software llama a esta función.

En primer lugar se obtienen del módulo DS3231 la fecha y hora, así como la temperatura interna del equipo de monitoreo. Para ello se hace uso del comando "DateTime now = rtc.now()" con el que se obtienen la fecha y la hora. Para la temperatura se hace uso del comando "float tempFloat = rtc.getTemperature" seguido por el comando "rtc.convertTemperature()". Con estos comandos se leen los registros anteriores y después se registra la temperatura actual. Véase Figura 31.

```
//RELOJ
 DateTime now = rtc.now(); //get the current date-time
  float tempFloat = rtc.getTemperature(); //read registers and display the temperature
  rtc.convertTemperature();             //convert current temperature into registers
```

Figura 31. Obtención de la fecha, hora y temperatura interna.

A continuación, se procede a abrir el archivo de texto en el cual se van a registrar todos los datos. Este se llamará datalog.txt y se creará con el comando "logFile = SD.open("datalog.txt", FILE_WRITE)" en caso de que no exista en la tarjeta microSD. En caso de ser un archivo existente este se abrirá. Además, con este comando se indica que los datos se escribirán en este mismo archivo abierto o creado. Véase Figura 32.

```
//SD
logFile = SD.open("datalog.txt", FILE_WRITE); // Abrir archivo y escribir valor
```

Figura 32. Apertura de archivo texto en tarjeta microSD.

El siguiente paso es la obtención de todos los datos que se quieren almacenar en la tarjeta microSD. Puesto que son los mismos que se almacenaran en IoT, a excepción de la memoria RAM libre y el tiempo de funcionamiento que no se almacenan en este caso, estos se obtienen del mismo modo. Por ello el código de obtención de estos datos es igual al de obtención de datos en la función comandosAT, por lo que para más información acerca de este, se sugiere al lector que consulte el apartado correspondiente. A continuación, se muestra el código empleado para la obtención de datos en la Figura 33.

El siguiente paso a realizar es comprobar la conexión con la tarjeta microSD. Esto se realiza mediante el uso de una sentencia condicional de tipo if else. Con ella se hace que en caso de que se puedan guardar los datos se proceda a ejecutar el código con el que estos se escriben en la tarjeta microSD. Y también que en caso de que no sea posible guardar datos en la tarjeta se envíe un mensaje de error al puerto serie.

A continuación, se muestra la primera parte de esta sentencia y su código interno. El comando de esta es "if (logFile) {" y como se ha comentado, esto hace que, en caso de que se puedan guardar los datos, se proceda a ejecutar su código interno, el cual se encargará de almacenar los datos en la tarjeta microSD.

```
//LVDT
for ( int i = 0 ; i < 10 ; i++ ) {
sensor1 += analogRead(sensorPin1);
delay(10);
}
for ( int i = 0 ; i < 10 ; i++ ) {
sensor2 += analogRead(sensorPin2);
delay(10);
}
sensor1 /= 10;
sensor1 = map(sensor1,0,1023,8536,869);
sensor2 /= 10;
sensor2 = map(sensor2,0,1023,8544,16211);

  //BATERIA
float voltajeBat =  (float)25*analogRead(A8)/1023;
float voltajePanSol =  (float)25*analogRead(A9)/1023;
float voltajeLVDT =  (float)25*analogRead(A10)/1023;
```

Figura 33. Obtención de los datos de las distintas variables a almacenar.

El código de la sentencia “if (logFile) {” cuenta a su vez con otra sentencia condicional de tipo if else igual a la empleada en la función comandosAT. Con ella, como se explica en el apartado 1.3.2.1., se hace que se guarde el dato del sensor correspondiente a una u otra mitad del sensor. Es decir, a guardar como dato del sensor la variable sensor1 o la variable sensor2 en función de sus valores, o lo que es lo mismo en función de la posición del núcleo. Los comandos de esta sentencia condicional son “if(sensor2>8544) {“ para guardar como dato del sensor la variable sensor2 y “else {“ para guardar como dato del sensor la variable1. En ambos casos se guardan los mismos datos, a excepción del dato del sensor que será diferente.

El código empleado para guardar los datos es el mismo en todos los casos, y usa en primer lugar, el comando “logFile = SD.open("datalog.txt", FILE_WRITE)” para abrir el archivo y escribir en él. Después comandos de tipo “logFile.print()” para escribir los datos en la tarjeta. Con este tipo de comandos se escribe en el archivo de la tarjeta el dato o caracteres entrecomillados que se encuentran en el paréntesis. Y por último se hace uso de comandos de tipo “Serial.print()” para enviar esos mismos datos al monitor serie, para que estos puedan ser leídos cuando el microcontrolador esté conectado a un ordenador.

Además de dicha sentencia condicional con la que se almacenan los datos, dentro del código perteneciente a la sentencia “if (logFile) {” se resetean los valores del sensor1 y sensor2 a cero para realizar correctamente la próxima lectura del sensor, con los comandos “sensor1 =0” y “sensor2 =0”.

En la Figura 34 se pueden ver el comando “if (logFile) {” ,que ejecuta su código si hay acceso a la tarjeta microSD; y parte de su código interno, con el comando “if(sensor2>8544) {“, que hará que se guarde como valor del sensor el de la variable sensor2; y a su vez el código de esta segunda sentencia condicional, con el comando “logFile = SD.open("datalog.txt", FILE_WRITE)” para abrir el archivo y escribir en él y un comando de tipo “logFile.print()”, que en este caso guarda el día de la semana.

```
if (logFile) {
    if(sensor2>8544) {
      // Escribir en SD:

      {logFile = SD.open("datalog.txt", FILE_WRITE);  //Abrir fichero y escribir valor
      logFile.print(weekDay[now.dayOfWeek()]);    // Send Day-of-Week
      logFile.print(" ; ");
```

Figura 34. Detalle de condición que almacena los datos si existe acceso a la tarjeta microSD y envío de datos.

En cambio, en caso de que no se puedan guardar los datos no se ejecutara el comando "if (logFile) {" su código explicado en los párrafos anteriores no se ejecutara, y se ejecutara el código de la segunda parte de esa misma sentencia condicional. Es decir, no se ejecutará el comando "if (logFile) {" y se ejecutará el comando "else {" encargado de enviar con su código un mensaje de error al puerto serie. Para ello su código consta del comando "Serial.println("Error al abrir el archivo")" encargado de realizar dicha tarea.

1.3.2.4. Código función complementaria comandosSMSbateria

La función complementaria comandosSMSbateria es empleada por la función principal cuando el nivel de carga del equipo de monitoreo es bajo. Cuando es llamada por la función principal esta se ejecuta y envía un mensaje de texto al número de teléfono deseado para alertar del estado de bajo nivel de carga de la batería del equipo de monitorización. En el presente apartado se procede a mostrar y explicar el código empleado en esta función.

En la Figura 35 se muestra el código empleado en la función complementaria comandosSMSbateria. Además, posteriormente se lleva a cabo la explicación detallada del mismo.

```
void comandosSMSbateria(){
  float voltajeBat =  (float)25*analogRead(A8)/1023;

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);

  //llamar:
  //Sim900Serial.println("atd645934270;");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  //delay(2000);
  //delay(25000);//Duración del llamado antes de cortar
  //Sim900Serial.println("ath645934270;"); // comando AT cortar llamada

  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA,BATERIA CENTRALITA BAJA, VOLTAJE(V):");// Reemplzar por el texto a enviar
  Sim900Serial.println(voltajeBat);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red y
              //este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();
  }
```

Figura 35.Código función complementaria comandosSMSbateria.

Como en el resto de funciones, el código de la función complementaria comandosSMSbateria comienza estableciendo la función y dando pie al código de la misma. Ello se lleva a cabo con el comando void “comandosSMSbateria(){“. En las líneas siguientes se desarrolla su código con el que se envía el mensaje de alerta por batería baja en el equipo de monitoreo. Este finaliza con la llave de cierre y es ejecutado cuando la función loop principal llama a esta función.

Una vez que ya se ha establecido la función y se ha dado lugar al comienzo de su código este comienza obteniendo el valor del voltaje de la batería. Para ello se hace uso del mismo comando empleado en la función principal para tal efecto, “float voltajeBat=(float)25*analogRead(A8)/1023”, véase apartado correspondiente con la explicación del mismo.

A continuación, se hace uso de comandos AT con los que se comunica el microcontrolador con el shield GSM/GPRS, a fin de configurarlo y de que este envíe el mensaje deseado. Para establecer dicha comunicación y para el envío de cada uno de estos comandos por parte del microcontrolador al shield GSM/GPRS se utiliza el comando “Sim900Serial.println("")”. Este comando hace que el microcontrolador envíe el mensaje o comando que se encuentre dentro de las comillas de su paréntesis al módulo GSM/GPRS.

El primer comando utilizado es el comando "AT+CPIN?" con el que se envía el pin de desbloqueo de la tarjeta SIM en caso de ser necesario, en este caso la tarjeta no lo requiere. Después se espera dos segundos para que el módulo sim900 procese el comando.

El siguiente comando configura el sim900 en modo texto. Esto se lleva a cabo con el comando "AT+CMGF=1\r" tras el cual se realiza una espera de un segundo. El siguiente paso es indicar a que número de teléfono se desea enviar el mensaje de alerta. Esto se lleva a cabo con el comando "AT + CMGS = \"XXXXXXXXX\"". En él se han de sustituir las X por el número de teléfono deseado. Después se realiza una espera de un segundo.

En este punto se le ha de indicar al módulo sim900 qué mensaje se va a enviar. Esto se hace sin emplear ningún comando AT, simplemente se le envía al módulo con el comando “Sim900Serial.println("")” poniendo dentro de las comillas el mensaje deseado. Para este caso el mensaje es enviado con dos comandos, en el primero se envía el mensaje "ALERTA,BATERIA CENTRALITA BAJA, VOLTAJE(V):" y en el segundo el voltaje de la batería anteriormente obtenido como la variable voltajeBat. Ambos se incluirán en el mismo SMS. Después se realiza nuevamente una espera de un segundo.

Puesto que ya se ha enviado el SMS de alerta se emplea el comando “Sim900Serial.println((char)26)” con el que el microcontrolador pide al módulo sim900 que le pida una respuesta a la red para comprobar que se ha enviado el mensaje con éxito. Después se esperan cinco segundos y se llama a la función mostrarDatosSeriales para que el módulo muestre la respuesta obtenida.

Por último, se cierra la conexión, enviando al módulo el comando "AT+CIPSHUT". Se esperan cinco segundos y de nuevo se llama a la función mostrarDatosSeriales con la que comporbar la respuesta del módulo.

1.3.2.5. Código función complementaria comandosSMSlvdt

Al igual que la función anterior, esta función es empleada por la función principal para mandar un mensaje de alerta por SMS. En concreto la función complementaria comandosSMSlvdt es empleada cuando el nivel de carga de la batería del sensor LVDT es bajo. Cuando esta es llamada por la función principal se ejecuta y envía un mensaje de texto al número de teléfono deseado para alertar del estado de bajo nivel de carga de la batería del sensor. A continuación, se muestra el código empleado por la misma, Figura 36.

```
void comandosSMSlvdt(){
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);
  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA,BATERIA LVDT BAJA, VOLTAJE(V):");// Reemplzar por el texto a enviar
  Sim900Serial.println(voltajeLVDT);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
              //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

  }
```

Figura 36.Código función complementaria comandosSMSlvdt.

El código empleado en esta función es idéntico al empleado en la función complementaria comandosSMSbateria. Tan solo cambian el comando de obtención de voltaje, que es igual, pero leyendo el voltaje de la batería del sensor, y el mensaje enviado. Para este caso el mensaje es enviado con dos comandos, en el primero se envía el mensaje "ALERTA,BATERIA LVDT BAJA, VOLTAJE(V):" y en el segundo el voltaje de la batería anteriormente obtenido como la variable voltajeLVDT. A continuación, se muestran en detalle los comandos que cambian en la Figura 37.

```
void comandosSMSlvdt(){
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;
  Sim900Serial.println("ALERTA,BATERIA LVDT BAJA, VOLTAJE(V):");// Reemplzar por el texto a enviar
  Sim900Serial.println(voltajeLVDT);// Reemplzar por el texto a enviar
```

Figura 37. Detalle código obtención voltaje y código mensaje de aviso empleado.

1.3.2.6. Código función complementaria comandosSMStemp

La función complementaria comandosSMStemp, al igual que las dos funciones anteriores, es llamada por la función principal loop, en este caso para enviar un mensaje de alerta a través de SMS por exceso de temperatura en el interior del equipo de monitoreo. A continuación, se muestra el código empleado por la misma, Figura 38.

```
void comandosSMStemp(){
  int temp = rtc.getTemperature();

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);

  //llamar:
  //Sim900Serial.println("atd645934270;");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  //delay(2000);
  //delay(25000);//Duración del llamado antes de cortar
  //Sim900Serial.println("ath645934270;"); // comando AT cortar llamada

  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA, ALTA TEMPERATURA INTERNA, TEMPERATURA(°C):");// Texto a enviar
  Sim900Serial.println(temp);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
              //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

  }
```

Figura 38.Código función complementaria comandosSMStemp.

El código empleado en esta función es idéntico al empleado en la función complementaria comandosSMSbateria y al de la función comandosSMSlvdt. Tan solo cambian el comando de dato a enviar y el mensaje enviado.

Para el la obtención del dato a enviar se emplea el comando “int temp = rtc.getTemperature()”. Con él se obtiene la temperatura interna del equipo de monitoreo a través del módulo DS3231. En este caso el mensaje es enviado de nuevo con dos comandos, en el primero se envía el mensaje "ALERTA, ALTA TEMPERATURA INTERNA, TEMPERATURA(°C):" y en el segundo la temperatura del equipo de monitoreo anteriormente obtenido como la variable temp. A continuación, se muestran en detalle los comandos que cambian en la Figura 39.

```
void comandosSMStemp(){
  int temp = rtc.getTemperature();

  Sim900Serial.println("ALERTA, ALTA TEMPERATURA INTERNA, TEMPERATURA(°C):");
  Sim900Serial.println(temp);// Reemplzar por el texto a enviar
```

Figura 39. Detalle código obtención voltaje y código mensaje de aviso empleado

1.3.2.7. Código función complementaria wakeup

En el presente apartado se presenta y comenta el código empleado en la función complementaria wakeup. Esta función a diferencia del resto de funciones complementarias no es llamada directamente por la función principal loop. En esta ocasión a esta función la llama la interrupción que se encuentra al final del código de la función setup, "attachInterrupt(0,wakeup,RISING)". La misión de la función wakeup es cambiar al microcontrolador de estado inactivo o de bajo consumo a estado activo cuando es llamado por la interrupción.

A continuación, en la Figura 40 se muestra el código empleado en la función complementaria wakeup. Además, posteriormente se lleva a cabo la explicación detallada del mismo.

```
void wakeup(){ //Interrupt Handle Function
snoozeLib.wakeup(); //WakeUp The CPU
 state=!state;//Change the State
}
```

Figura 40.Código función complementaria comandosSMStemp.

El código empleado para esta función complementaria comienza, al igual que en las funciones anteriores, estableciendo la función en sí misma. En este caso se trata de la función complementaria wakeup. Dicha función se establece mediante el comando "void wakeup(){". La primera parte del comando la establece y después, con la llave de apertura, se da pie al comienzo del código de esta. Con ello el código se desarrolla en las líneas posteriores finalizando antes de la llave de cierre de la función y siendo ejecutado una única vez cuando la interrupción así lo demanda.

Seguidamente se continúa con el código de la propia función. Este es sencillo y corto ya que consta únicamente de dos líneas. A pesar de ello es necesario hacer una función para este código puesto que la interrupción así lo requiere. La primera línea tiene el comando "snoozeLib.wakeup()", Este comando llama a la función snoozeLib.wakeup de la librería SnoozeLib.h que hace que se active el microcontrolador. La segunda línea cambia el estado del pin6 de HIGH a LOW con el comando "state=!state" con lo que el pin queda preparado para activar la interrupción de activación del microcontrolador en un futuro.

ANEJO IV: CÓDIGO SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

```
//SRAM LIBRE
#include <MemoryFree.h>

//MILISEGUNDOS DE FUNCIONAMIENTO
unsigned long time;

//WDT/RESET
#include <avr/wdt.h> // Incluir la librería que contiene el watchdog (wdt.h)

//LVDT
const int sensorPin1 = A0;
const int sensorPin2 = A1;
int sensor1=0;
int sensor2=0;
int rele = 13;

//SD:
#include <SD.h>
File logFile;

//RELOJ:
#include <Wire.h>
#include "Sodaq_DS3231.h"

char weekDay[][4] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

//year, month, date, hour, min, sec and week-day(starts from 0 and goes to 6)
//writing any non-existent time-data may interfere with normal operation of the RTC.
//Take care of week-day also.
DateTime dt(2011, 11, 10, 15, 18, 0, 5);

//SLEEP:
#include <SnoozeLib.h>  //Include the library
int state = HIGH;//Led State

//GSM
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial Sim900Serial(10, 11);//Configurarión de los pines serial por software

//BATERIA
int SensorBat = A8;
int SensorPanSol = A9;
int SensorLVDT = A10;

void setup() {

  Serial.begin(9600);

  //LIBRERIA WDT
  wdt_disable(); // Desactivar el watchdog mientras se configura

  //GSM
  Sim900Serial.begin(19200);//Arduino se comunica con el SIM900 a una velocidad de 19200bps
      //Encendido del módulo por sofware
    //digitalWrite(9, HIGH);
    //delay(1000);
    //digitalWrite(9, LOW);
    //delay(20000);//Tiempo prudencial para el escudo inicie sesión de red con tu operador
```

```
  //LVDT
  pinMode(0,INPUT);
  pinMode(1,INPUT);
  pinMode(rele, OUTPUT); //modo salida

  //SD:
  Serial.print("Iniciando microSD ...");
  if (!SD.begin(53))
    {
    Serial.println("Error al iniciar");
    return;
    }
  Serial.println("Iniciado correctamente");

  //RELOJ:
  Wire.begin();
  rtc.begin();
 // rtc.setDateTime(dt); //Adjust date-time as defined 'dt' above

  //SLEEP:
  pinMode(6,OUTPUT);
  attachInterrupt(0,wakeup,RISING); //Attaching the 0 Interrupt for RISING.

}

void loop() {
  //WDT/RESET
  if(time>6500000) {
    wdt_enable(WDTO_15MS); // Configurado a 15 milisegundos
    delay(20);
  }

  //LVDT
  digitalWrite(rele, LOW);
  delay(500);

  //GSM
  comandosAT();//Llama a la función comandosAT
  if(Sim900Serial.available())//Verificamos si hay datos disponibles desde el SIM900
  Serial.write(Sim900Serial.read());//Escribir datos

  //SD
  comandosSD();//Llama a la función comandosSD

  //BATERIA
  float voltajeBat =  (float)25*analogRead(A8)/1023;
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;
  int temp = rtc.getTemperature();
  Serial.print("Voltaje Bateria =  ");
  Serial.println(voltajeBat);
  Serial.print("Voltaje LVDT =  ");
  Serial.println(voltajeLVDT);
  Serial.print("Temperatura interna =  ");
  Serial.println(temp);

  //SMS BATERIA BAJA Y ALTA TEMPERATURA INTERNA
  if(voltajeBat<11.05){
    comandosSMSbateria();//Llama a la función comandosSMSbateria
  }
```

```
  //if(voltajeLVDT<11.05){
   // comandosSMSlvdt();//Llama a la función comandosSMSlvdt
    //}

  if(temp>40){
    comandosSMStemp();//Llama a la función comandosSMSlvdt
    }

  //LVDT
  digitalWrite(rele, HIGH);
  delay(500);

  //SLEEP
  Serial.println("Durmiendo...");
  snoozeLib.snooze(550000);//Snooze the CPU(1000=1segundo); 10 min=550000
  Serial.println("Despierto!!");
  digitalWrite(6,state);

  delay(1000);
}

void comandosAT(){

  //lvdt
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor1 += analogRead(sensorPin1);
  delay(10);
  }
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor2 += analogRead(sensorPin2);
  delay(10);
  }
  sensor1 /= 10;
  sensor1 = map(sensor1,0,1023,8536,869);
  sensor2 /= 10;
  sensor2 = map(sensor2,0,1023,8544,16211);

  //Tª RTC
      int temp = rtc.getTemperature();

  //BATERIA
  float voltajeBat =  (float)25*analogRead(A8)/1023;
  float voltajePanSol =  (float)25*analogRead(A9)/1023;
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;

  int sram = freeMemory();

  //GSM
  Sim900Serial.println("AT");//Comprueba estado del módulo
  delay(2000);
  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);
  Sim900Serial.println("AT+CREG?");// estado de conexión a la red
  delay(2000);
  Sim900Serial.println("CGATT?");//Si está conectado a internet o no
  delay(2000);
  Sim900Serial.println("CIPSHUT");//cerrará el contexto GPRS PDP
  delay(2000);
  Sim900Serial.println("AT+CIPSTATUS");//Consultar el estado actual de la conexión
  delay(2000);
  Sim900Serial.println("AT+CIPMUX=0");//comando configura el dispositivo para una conexión
                                      //IP única o múltiple 0=única
```

```
  delay(3000);
  mostrarDatosSeriales();
  //Comando configura el APN, nombre usuario y contraseña.
  Sim900Serial.println("AT+CSTT=\"internet\",\"\",\"\"");
  delay(1000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIICR");//REALIZAR UNA CONEXIÓN INALÁMBRICA CON GPRS O CSD
  delay(3000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIFSR");// Obtenemos nuestra IP local
  delay(2000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSPRT=0");//Establece un indicador '>' al enviar datos
  sensor1;
  sensor2;
  temp;
  voltajeBat;
  voltajePanSol;
  voltajeLVDT;
  time;
  sram;
  delay(3000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
  //Indicamos el tipo de conexión, url o dirección IP y puerto al que realizamos la conexión
  delay(6000);
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSEND");//ENVÍA DATOS A TRAVÉS DE una CONEXIÓN TCP O UDP
  delay(4000);
  mostrarDatosSeriales();

  if(sensor2>8544) {
    String datos="GET https://api.thingspeak.com/update?api_key=BPTU1JL7TURYPBY5&field1=0"
    + String(sensor2)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
    + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
    + "&field6=0" + String(time)+ "&field7=0" + String(sram);

    Sim900Serial.println(datos);//Envía datos al servidor remoto
    delay(4000);
    mostrarDatosSeriales();
    delay(1000);
      }
  else {
    String datos="GET https://api.thingspeak.com/update?api_key=BPTU1JL7TURYPBY5&field1=0"
    + String(sensor1)+ "&field2=0" + String(temp)+ "&field3=0" + String(voltajeBat)
    + "&field4=0" + String(voltajePanSol)+ "&field5=0" + String(voltajeLVDT)
    + "&field6=0" + String(time)+ "&field7=0" + String(sram);

    Sim900Serial.println(datos);//Envía datos al servidor remoto
    delay(4000);
    mostrarDatosSeriales();
    delay(1000);
     }
sensor1 =0;
sensor2 =0;
Sim900Serial.println((char)26);
delay(5000);
//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
//y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
Sim900Serial.println();
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
delay(5000);
```

```
 Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
 delay(5000);
 mostrarDatosSeriales();

void mostrarDatosSeriales()//Muestra los datos que va entregando el sim900
{
  while(Sim900Serial.available()!=0)
  Serial.write(Sim900Serial.read());
}

void comandosSD(){

  //RELOJ
   DateTime now = rtc.now(); //get the current date-time
    float tempFloat = rtc.getTemperature(); //read registers and display the temperature
    rtc.convertTemperature();            //convert current temperature into registers

  //SD
  logFile = SD.open("datalog.txt", FILE_WRITE); // Abrir archivo y escribir valor

  //LVDT
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor1 += analogRead(sensorPin1);
  delay(10);
  }
  for ( int i = 0 ; i < 10 ; i++ ) {
  sensor2 += analogRead(sensorPin2);
  delay(10);
  }
  sensor1 /= 10;
  sensor1 = map(sensor1,0,1023,8536,869);
  sensor2 /= 10;
  sensor2 = map(sensor2,0,1023,8544,16211);

    //BATERIA
  float voltajeBat =  (float)25*analogRead(A8)/1023;
  float voltajePanSol =  (float)25*analogRead(A9)/1023;
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;

  if (logFile) {
      if(sensor2>8544) {
        // Escribir en SD:

        {logFile = SD.open("datalog.txt", FILE_WRITE);  //Abrir fichero y escribir valor
        logFile.print(weekDay[now.dayOfWeek()]);    // Send Day-of-Week
        logFile.print(" ; ");
        logFile.print(now.year(), DEC);    // Send date
        logFile.print('/');
        logFile.print(now.month(), DEC);
        logFile.print('/');
        logFile.print(now.date(), DEC);
        logFile.print(' ');
        logFile.print(" ; ");
        logFile.print(now.hour(), DEC);    // Send time
        logFile.print(':');
        logFile.print(now.minute(), DEC);
        logFile.print(':');
        logFile.print(now.second(), DEC);
        logFile.print(" ; ");
        logFile.print("InternalTemperature; ");    //Send temperature
        logFile.print(rtc.getTemperature());
```

```
    logFile.print(" ; ");
    logFile.print(" °C ; ");
    logFile.print("Sensor Value2(µ);"); // Eviar dato sensor
    logFile.print(sensor2);
    logFile.print(" ; ");
    logFile.print("Voltaje Bateria =;  ");
    logFile.print(voltajeBat);
    logFile.print(" ; ");
    logFile.print("Voltaje Panel Solar =;  ");
    logFile.print(voltajePanSol);
    logFile.print(" ; ");
    logFile.print("Voltaje LVDT =;  ");
    logFile.println(voltajeLVDT);
    logFile.close();}   //Cerrar escritura SD

    //Escribir en monitor serie:
    Serial.print(weekDay[now.dayOfWeek()]);   // Send Day-of-Week
    Serial.print(", ");
    Serial.print(now.year(), DEC);      // Send date
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.date(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);      // Send time
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    Serial.print("InternalTemperature; ");      //Send temperature
    Serial.print(rtc.getTemperature());
    Serial.println(" °C");
    Serial.print("Sensor Value2(µ); "); //Enviar dato sensor
    Serial.print(sensor2);
    Serial.print(";");
    Serial.print("Voltaje Bateria =  ");
    Serial.print(voltajeBat);
    Serial.print(";");
    Serial.print("Voltaje Panel Solar =  ");
    Serial.print(voltajePanSol);
    Serial.print(";");
    Serial.print("Voltaje LVDT =  ");
    Serial.print(voltajeLVDT);
    Serial.println();

    //delay(1000);
    }

  else {
    // Escribir en SD:
    {logFile = SD.open("datalog.txt", FILE_WRITE);  //Abrir fichero y escribir valor
    logFile.print(weekDay[now.dayOfWeek()]);   // Send Day-of-Week
    logFile.print(" ; ");
    logFile.print(now.year(), DEC);  // Send date
    logFile.print('/');
    logFile.print(now.month(), DEC);
    logFile.print('/');
    logFile.print(now.date(), DEC);
    logFile.print(' ');
    logFile.print(" ; ");
```

```
      logFile.print(now.hour(), DEC);    // Send time
      logFile.print(':');
      logFile.print(now.minute(), DEC);
      logFile.print(':');
      logFile.print(now.second(), DEC);
      logFile.print(" ; ");
      logFile.print("InternalTemperature; ");    //Send temperature
      logFile.print(rtc.getTemperature());
      logFile.print(" ; ");
      logFile.print(" °C ; ");
      logFile.print("Sensor Valuel(µ);"); // Eviar dato sensor
      logFile.print(sensor1);
      logFile.print(" ; ");
      logFile.print("Voltaje Bateria =;   ");
      logFile.print(voltajeBat);
      logFile.print(" ; ");
      logFile.print("Voltaje Panel Solar =;   ");
      logFile.print(voltajePanSol);
      logFile.print(" ; ");
      logFile.print("Voltaje LVDT =;   ");
      logFile.println(voltajeLVDT);
      logFile.close();}     //Cerrar escritura SD

      //Escribir en monitor serie
      Serial.print(weekDay[now.dayOfWeek()]);
      Serial.print(", ");
      Serial.print(now.year(), DEC);
      Serial.print('/');
      Serial.print(now.month(), DEC);
      Serial.print('/');
      Serial.print(now.date(), DEC);
      Serial.print(' ');
      Serial.print(now.hour(), DEC);
      Serial.print(':');
      Serial.print(now.minute(), DEC);
      Serial.print(':');
      Serial.print(now.second(), DEC);
      Serial.println();
      Serial.print("Temperature: ");       //Send temperature
      Serial.print(rtc.getTemperature());
      Serial.println(" °C");
      Serial.print("Sensor Valuel(µ): "); //Enviar dato sensor
      Serial.print(sensor1);
      Serial.print(";");
      Serial.print("Voltaje Bateria =   ");
      Serial.print(voltajeBat);
      Serial.print(";");
      Serial.print("Voltaje Panel Solar =   ");
      Serial.print(voltajePanSol);
      Serial.print(";");
      Serial.print("Voltaje LVDT =   ");
      Serial.print(voltajeLVDT);
      Serial.println();

    }
  sensor1 =0;
  sensor2 =0;
}
```

```
  else {
    Serial.println("Error al abrir el archivo");
  }
}

void comandosSMSbateria(){
  float voltajeBat =  (float)25*analogRead(A8)/1023;

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);

  //llamar:
  //Sim900Serial.println("atd645934270;");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  //delay(2000);
  //delay(25000);//Duración del llamado antes de cortar
  //Sim900Serial.println("ath645934270;"); // comando AT cortar llamada

  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA,BATERIA CENTRALITA BAJA, VOLTAJE(V):");// Reemplzar por el texto a enviar
  Sim900Serial.println(voltajeBat);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red y
              //este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();
  }
void comandosSMSlvdt(){
  float voltajeLVDT =  (float)25*analogRead(A10)/1023;

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);
  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA,BATERIA LVDT BAJA, VOLTAJE(V):");// Reemplzar por el texto a enviar
  Sim900Serial.println(voltajeLVDT);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
              //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

  }
```

```
void comandosSMStemp(){
  int temp = rtc.getTemperature();

  Sim900Serial.println("AT+CPIN?");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  delay(2000);

  //llamar:
  //Sim900Serial.println("atd645934270;");//Introducir el PIN de la SIM. Cambiar XXXX por el PIN.
  //delay(2000);
  //delay(25000);//Duración del llamado antes de cortar
  //Sim900Serial.println("ath645934270;"); // comando AT cortar llamada

  //Enviar sms
  Serial.println("Enviando SMS...");
  Sim900Serial.print("AT+CMGF=1\r"); // comando AT para configurar el SIM900 en modo texto
  delay(1000);
  Sim900Serial.println("AT + CMGS = \"XXXXXXXXX\"");//reemplzar por el número a enviar el mensaje
  delay(1000);
  Sim900Serial.println("ALERTA, ALTA TEMPERATURA INTERNA, TEMPERATURA(°C):");// Texto a enviar
  Sim900Serial.println(temp);// Reemplzar por el texto a enviar
  delay(1000);

  Sim900Serial.println((char)26);
  delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red
              //y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
  Sim900Serial.println();
  mostrarDatosSeriales();
  Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
  delay(5000);
  mostrarDatosSeriales();

  }
void wakeup(){ //Interrupt Handle Function
snoozeLib.wakeup(); //WakeUp The CPU
 state=!state;//Change the State
}
```

ANEJO V: IMPLEMENTACIÓN Y PUESTA A PRUEBA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ÍNDICE

ÍNDICE DE FIGURAS

En el presente anejo se describe la implementación de la solución propuesta. Esta consiste en la puesta en funcionamiento del prototipo construido, empleando los componentes, esquemas y software seleccionados y diseñados para llevarlo a cabo. El prototipo se muestra en el presente documento junto a algunas imágenes de su proceso de construcción. También se muestran y aclaran algunas de las técnicas usadas para llevar a cabo la construcción. Además, se presentan y explican algunos componentes accesorios diseñados, construidos e instalados en este proceso de implementación que serán empleados en el posterior testeo para su instalación en campo.

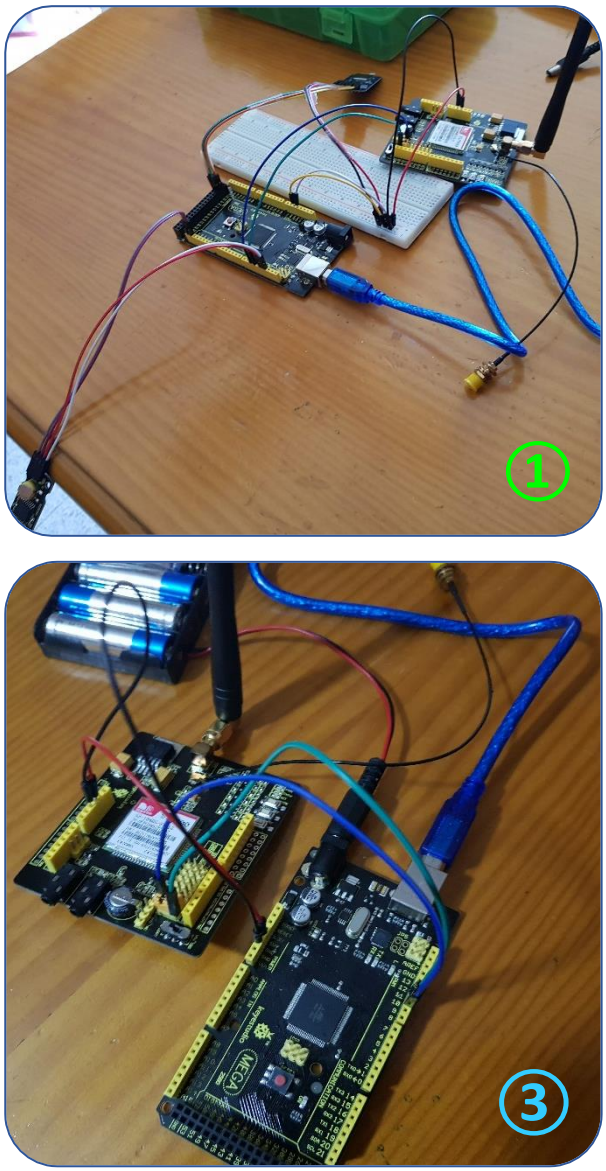
Por otra parte, también se lleva a cabo un testeo del mismo, con el que se comprueba que el prototipo funciona correctamente, que ningún componente es problemático y que la solución diseñada es realmente válida y funcional. Además, en la comprobación de su funcionamiento se detectan posibles mejoras a realizar en el futuro. Para llevar a cabo el testeo se han empleado los componentes indicados en el Anejo I, los esquemas y conexiones del Anejo II, y el software de los Anejos III y IV. Concretamente el testeo se basa en la monitorización de un dendrómetro basado en un sensor de desplazamiento L.V.D.T e instalado en un cerezo de una parcela de cerezos situada en Biar.

1. IMPLEMENTACIÓN

En el presente apartado se explica cómo se ha llevado a cabo el proceso de fabricación y puesta a punto del prototipo y los elementos empleados para su testeo, como por ejemplo el equipo del sensor L.V.D.T. Para ello se explica cómo han sido montados junto con los elementos accesorios que han sido necesarios emplear en este proceso, la descripción del mismo se acompaña de imágenes que facilitan el entendimiento. Además, se detallan determinados procesos singulares tanto de fabricación/instalación de algunos de sus componentes como de puesta a punto del prototipo con el equipo de sensor L.V.D.T. incluida la calibración del mismo.

1.1. PRUEBAS INICIALES

Previo al montaje de todos los componentes se realizaron pruebas con el microcontrolador y los distintos módulos que se iban a instalar. El objetivo de estas pruebas fue comprobar su buen funcionamiento. Con ello se ha verificado, en primer lugar, que las conexiones entre el microcontrolador y los distintos módulos son correctas y funcionan adecuadamente. Posteriormente, se comprueba que el software desarrollado funciona correctamente. En este proceso, se ha optimizado el programa de funcionamiento. Para la optimización del software, este se ha ido modificando y mejorando a medida que se comprobaban cada módulo por separado y en conjunto. De este modo se ha obtenido un sketch que además de funcionar correctamente, es más sencillo que el inicial y mucho más fiable, dado que ahora se ha comprobado que funciona con seguridad. En este proceso surgieron distintos problemas, tanto de conexiones de los módulos como fallos de software. Cada uno de los fallos detectados sirvió para ir introduciendo mejoras de programación y también de configuración y conexionado. La Figura 1 muestra varias imágenes pertenecientes a este proceso. En ellas se realiza la comprobación del módulo microSD y módulos GSM/GPRS sim900 por separados y también conjuntamente con el módulo DS3231.



①Microcontrolador y módulos microSD, GSM/GPRS sim900 y DS3231.

② Microcontrolador y módulo microSD.

③Microcontrolador y módulo, GSM/GPRS sim900.

Figura 1. Comprobación de funcionamiento de módulos microSD, GSM/GPRS sim900 y DS3231.

1.2. ELEMENTOS ACCESORIOS DE MONTAJE DEL PROTOTIPO

Una vez comprobado el buen funcionamiento de los componentes principales del equipo diseñado se procede a su montaje, para el cual han sido necesarios diversos accesorios detallados a continuación.

1.2.1. Caja eléctrica

Todos los componentes del equipo principal han sido montados en una caja eléctrica estanca IP56 de 25x19,5x10cm. De este modo quedan todos confinados en un lugar seguro, puesto que dicha caja los protege de las inclemencias ambientales. A la caja se le realizaron perforaciones en los laterales para instalar en ellos los interruptores para el encendido del equipo y del indicador de nivel de batería, así como para los conectores de la placa solar y el sensor de testeo. Para la instalación de los conectores se usaron arandelas de goma, con las que se garantiza la estanqueidad de la caja y también se fabricaron dos soportes en chapa con los que se refuerza la pared de la caja para evitar su rotura al apretar las tuercas de sujeción de los conectores.

Además, en la tapa frontal se realizó una ventana a la que se le instaló un cristal templado y a través de la cual se observa el indicador de nivel de carga sin necesidad de abrir la caja. Véase Figura 2.

De igual modo, los componentes del sensor LVDT son montados en una caja eléctrica estanca IP55 de 16x12x7cm. Esta dispone de terminaciones integradas en sus laterales. En ellas se instalan los conectores de la placa solar y el conector del sensor del equipo de monitoreo, así como también un conector para comunicar el equipo del sensor L.V.D.T con el equipo de monitoreo. En la Figura 2 se puede ver una imagen de la caja del sensor empleada.

① Vista frontal de la caja del equipo de monitoreo en la que se aprecia la ventana con el indicador de nivel de batería.

② Vista frontal de la caja del sensor L.V.D.T.

③ Vista lateral de la caja del equipo de monitoreo en la que se ven los interruptores en el lateral derecho y el conector del panel solar en el izquierdo.

④ Vista lateral de la caja del equipo de monitoreo en la que se ve el conector del sensor L.V.D.T. en el lateral izquierdo y el conector del panel solar en el derecho.

⑤ Detalle de los soportes de refuerzo de los conectores.

Figura 2. Cajas eléctricas del equipo de monitoreo y sensor con detalle de algunos accesorios.

1.2.2. Soportes cajas eléctricas

Además, para ambas cajas se ha diseñado, fabricado e instalado un soporte con asa con el cual se transporta e instala con facilidad la caja en campo. Este consta de un marco metálico que es atornillado a la caja con juntas de goma para garantizar la estanqueidad de la misma. A este marco se fija mediante tornillos y presillas un tubo en forma de u invertida. De este modo dicho tubo hace de asa para llevar la caja y a la vez permite la instalación en campo de las cajas. En la Figura 3 se pueden observar algunas imágenes del proceso de fabricación y montaje del soporte del equipo de monitoreo y de su instalación en campo.

① Fabricación de piezas del soporte.

② Lijado de piezas del soporte.

③ Pintado de piezas del soporte.

④ Montaje del soporte en caja eléctrica de monitoreo.

⑤ Instalación de caja de monitoreo con soporte en campo.

Figura 3. Soporte equipo de monitoreo.

Para su instalación en campo, sencillamente, se clavan dos piquetas con un diámetro inferior al diámetro interior del tubo de soporte. Una vez clavadas se coloca la caja de tal forma que las piquetas queden en el interior del tubo. Con ello las cajas se instarán fácilmente en campo. En la Figura 3 se pueden observar algunas imágenes del proceso de fabricación y montaje del soporte del equipo de monitoreo y de su instalación en campo. De igual modo, en la Figura 4 también pueden verse imágenes del proceso de fabricación del equipo para el sensor LVDT y de su montaje e instalación en campo.

1 Fabricación de piezas del soporte.

2 Presentación de piezas del soporte.

3 Soporte pintado y montado en caja del equipo L.V.D.T.

4 Instalación de caja del sensor L.V.D.T. con soporte en campo.

Figura 4. Soporte equipo del sensor L.V.D.T.

1.2.3. Soporte modulo solar

Al igual que para la instalación en campo de las cajas eléctricas donde se montan los equipos, para la instalación de la placa solar en campo también es necesario un soporte. La estructura de

soporte del módulo solar ha de ser elevada, con el fin de evitar la sombra producida por los árboles. Además, sería convenientemente que fuera regulable tanto en altura, como en inclinación del módulo. Los soportes comerciales de estas características tienen un alto coste, teniendo en cuenta que son relativamente simples. Por ello se decidió diseñar y fabricar un soporte propio. Este se ha construido a partir de tres perfiles en ángulo de 90 grados, dos bisagras, un tubo rectangular galvanizado y una piqueta. El soporte diseñado es regulable en altura y ángulo de inclinación del módulo solar.

Básicamente la estructura consta de un tubo galvanizado rectangular con orificios dispuestos de verticalmente. A este se le une en la parte superior el módulo solar con una bisagra. La parte inferior del módulo se sujeta a la estructura utilizando dos perfiles en L y otra bisagra que permite modificar el ángulo de inclinación del panel. El segundo perfil en L se une a la piqueta perpendicularmente mediante el uso de un pasador y los orificios de que disponen los dos elementos conforme se observa en la Figura 5. En ella se observa como el perfil en L se une perpendicularmente al módulo solar por el extremo opuesto y haciendo uso de una bisagra. Cabe destacar que tanto la piqueta como el tubo rectangular y el perfil en L disponen de múltiples orificios. Gracias a ello y a que las uniones son móviles, gracias al uso de bisagras en ellas, se puede regular la altura y ángulo del panel solar. Para llevar a cabo dicha regulación, sencillamente se ha de jugar con las distintas combinaciones de orificios empleados para unir estos tres elementos con el pasador. En la Figura 5 se puede observar el módulo solar instalado en campo con el soporte diseñado.

Figura 5. Modulo solar y el soporte diseñado instalados en campo.

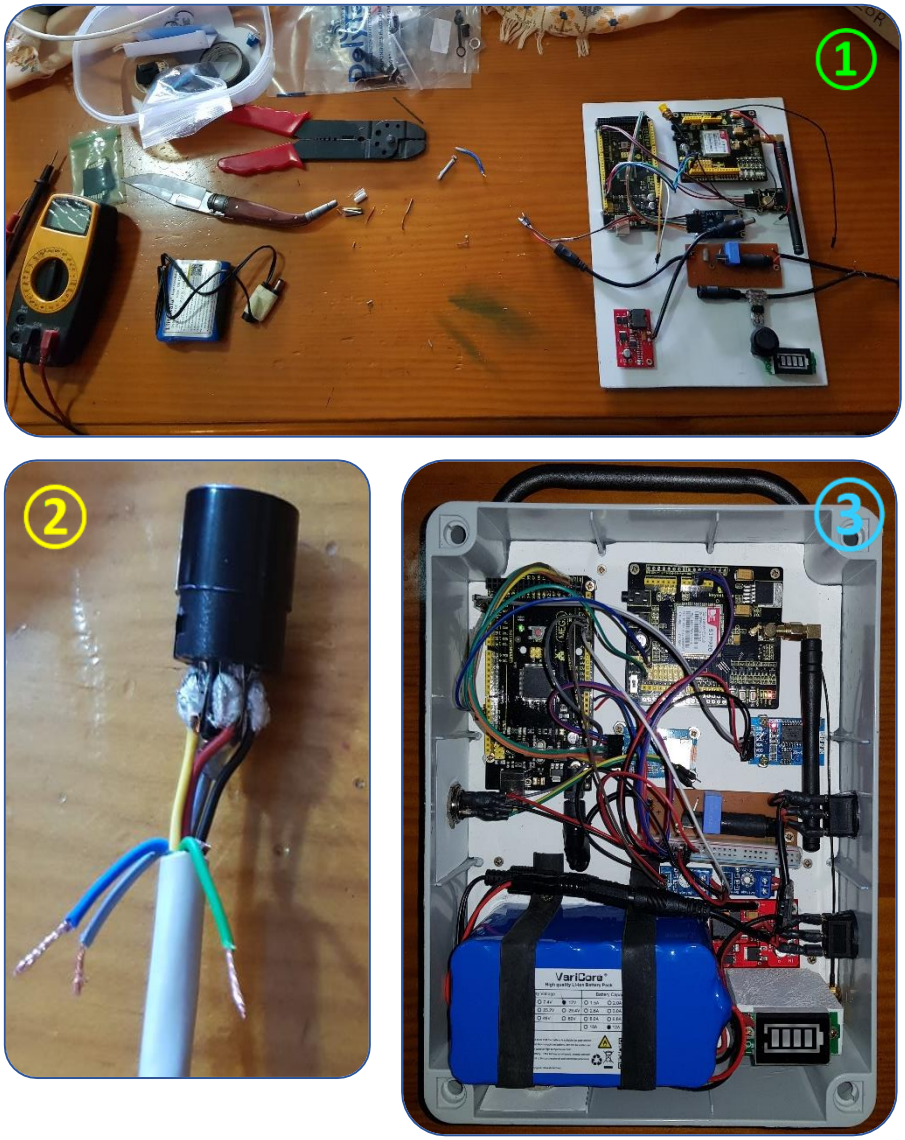
1.3. MONTAJE DEL PROTOTIPO

Como se ha comentado anteriormente, el prototipo se va a fabricar montando los componentes seleccionados para la solución propuesta, en cajas eléctricas y junto a los accesorios que estas necesiten, también comentados en apartados previos. Para su montaje se han seguido los esquemas eléctricos y de conexiones diseñados como solución, tanto para el equipo de monitoreo, como para el sensor y descritos en el anejo correspondiente.

Para facilitar su instalación en las cajas eléctricas, los componentes se montan sobre una lámina plástica que hace de soporte para el equipo de monitoreo, y en otra para el equipo del sensor.

Estos montajes con todos los componentes ya instalados se fijan al interior de sus respectivas cajas con tornillos. Para llevar a cabo este montaje, primero se han presentado los componentes sobre las láminas plásticas, con el fin de ubicarlos de tal forma que las conexiones se pudieran realizar sencillamente, de forma segura y permitiendo a la vez una buena visualización de todos los componentes. Una vez ubicados de forma definitiva todos los elementos, se procede a la fijación de cada componente sobre las mismas. Para ello se emplean tornillos y separadores, con los que se fija cada elemento en su lugar a través de los orificios de que disponen los componentes para tal fin.

En la Figura 6 se muestran imágenes del proceso de fijación de los componentes del equipo de monitoreo principal a su plancha plástica y también, del interior de la caja eléctrica de este mismo equipo con todos los componentes y accesorios ya montados.



① Proceso de presentación y fijación de componentes a plancha plástica.

② Detalle conexión de cables a conector.

③ Equipo final con todos los componentes instalados.

Figura 6. Montaje de componentes y detalle de conexión de conector.

El indicador de nivel de carga de la batería se además se ha emplazado sobre un taco de madera que lo eleva, dejándolo cerca de la tapa, con el fin de mejorar su visualización a través de la ventana de la misma. La batería se ha fijado por medio de dos tiras elásticas que se han atornillado en la plancha. El hecho de que la batería se haya fijado con tiras elásticas hace que esta esté bien fijada a la vez que se puede sacar y reemplazar en caso necesario, sin necesidad de desatornillar nada.

En cuanto a la conexión del módulo solar con el equipo de monitoreo y con el equipo del sensor se han empleado en ambos casos cable de alimentación de dos hilos de 0,75mm^2 y un conector macho junto a otro hembra. Para conectar el equipo de monitoreo con el equipo del sensor se han empleado cuatro conectores, dos macho y dos hembra, además de un cable de 7 hilos. En todos los casos los cables se han unido a los conectores mediante soldadura y se han aislado las uniones con tubo termoretráctil. Véase Figura 6.

En general, los componentes del equipo son tarjetas o módulos, preparados para funcionar, sin necesidad de realizar el montaje de sus piezas en la placa. Estos tan solo han de ser fijados en las cajas y conectados unos con otros. En cambio, para llevar a cabo el acondicionamiento de las alimentaciones y de la señal del sensor se emplean elementos sueltos, como el fusible o el regulador de tensión que requieren de un soporte donde conectarlos. En el siguiente apartado se explica cómo se montan estos elementos en una placa con circuito impreso, que les da soporte y conexión.

1.3.1. Soporte y sujeción de elementos sueltos

La mayoría de los componentes del equipo son tarjetas o módulos preparados para funcionar, con elementos electrónicos que los componen ya conectados y soportados en placas, que además cuentan con orificios para su fijación. En cambio el acondicionamiento de la alimentación del equipo de monitoreo y del sensor LVDT, requiere de elementos, como son el fusible, el regulador de tensión y los condensadores en el primer caso, y los elementos de acondicionamiento de la señal del sensor, como son las resistencias y diodos en el caso del LVDT. Por tanto, para estos componentes, es necesario realizar una solución que los soporte y conecte según los circuitos diseñados para el correcto funcionamiento del equipo. Para ello, se ha decidido crear una placa de circuito impreso PCB (de sus siglas en ingles) en la que se instalen estos componentes. Con ello se consigue dar un soporte a estos elementos a la vez que se conectan unos con otros, a través del circuito impreso en la misma. En cada caso, los elementos disponen en la placa de una posición y un circuito específicos, que corresponden a los diseñados. Además, a estas placas se les realizan orificios que permiten su instalación en la caja con tornillos y separadores, al igual que el resto de componentes.

Al tratarse de un prototipo las placas no han sido compradas bajo demanda, ya que para ello sería necesario un pedido mínimo, si no que han sido de diseño y fabricación propia. A continuación, se procede a explicar el proceso de fabricación de las placas de circuito impreso y el montaje de los componentes que se conectan y dan funcionalidad a estas placas. En todos los casos se sigue el mismo procedimiento, que se describe a continuación.

En la Figura 7 se muestran imágenes del proceso de fabricación de las placas con los componentes de acondicionamiento.

① Placa virgen cortada a medida

② Placa virgen con circuito trazado

③ Elementos utilizados en ataque a placa

④ Atacado a placa

⑤ Placa atacada y aclarada

⑥ Placa con disolvente tras el atacado

⑦ Limpieza con disolvente de placa atacada

⑧ Placa atacada y limpia con circuito impreso

⑨ Placa con circuito impreso y perforaciones de soporte.

⑩ ⑪ Montaje y soldadura de elementos

⑫ ⑬ Placas montadas y listas para instalar

Figura 7. Proceso de fabricación y montaje de placas de circuito impreso (PCB).

Para llevar a cabo la fabricación de las placas de circuito impreso se parte de una placa virgen, que tiene una de sus caras recubierta por una lámina de cobre. Esta placa será la base sobre la cual se trazará el circuito deseado y en la que se instalarán los elementos. El procedimiento seguido se describe a continuación.

Primero, se pule la placa ligeramente y se limpia con un paño y disolvente la superficie de la misma en la que se encuentra la lámina de cobre. De este modo se elimina la suciedad superficial, quedando el cobre al descubierto.

El siguiente paso, consiste en trazar el circuito deseado con un rotulador permanente sobre el cobre. Con ello se protege el cobre situado bajo la tinta, así cuando posteriormente se ataque a la placa con ácido el cobre bajo la tinta no se verá afectado, eliminándose todo el cobre excepto el protegido y quedando con ello solo el circuito en cobre en la placa.

Una vez trazado el circuito se procede a atacar la placa. Para ello se emplea ácido clorhídrico (HCl) comercial con una concentración del 20%. Además, también se usa peróxido de hidrogeno (H_2O_2) 30% p./v. Estos dos elementos se mezclan y diluyen con agua en una proporción del 25% ácido clorhídrico, 25% peróxido de hidrógeno y 50% de agua. De este modo en la mezcla primero entran en reacción el ácido clorhídrico con el peróxido de hidrogeno, liberando dicloro (Cl_2) y agua según la siguiente reacción química.

$$2HCL + H_2O_2 \rightarrow Cl_2 + 2H_2O$$

Seguidamente, se sumerge la placa con el circuito trazado en esta mezcla. Al hacerlo, el dicloro reacciona con el cobre de la placa que se encuentra al descubierto formándose cloruro de cobre, véase la reacción química mostrada a continuación. El cloruro de cobre se disuelve en la mezcla, por lo que, de este modo, se elimina el cobre que esta al descubierto en la placa. Así se consigue eliminar el cobre sobrante de la placa quedando tan solo el cobre cubierto por el rotulador permanente que representa el trazado del circuito.

$$Cl_2 + 2Cu \rightarrow 2CuCl$$

Una vez realizado el ataque a la placa, esta se aclara con agua para retirar los restos de la disolución y se limpia con disolvente. De este modo se elimina la tinta del permanente y los restos de suciedad que sobre la placa puedan haber.

Posteriormente se perfora la placa en los puntos donde se van a instalar los componentes. Los componentes se colocan en su posición y se sueldan al circuito con estaño, quedando la placa lista para su instalación.

Finalmente, estas placas se fijan en las cajas de igual modo que se fijan el resto de componentes, como se ha explicado anteriormente.

1.3.2. Calibrado del sensor con el prototipo

El proceso de calibrado es necesario para poder obtener con el prototipo y el dendrómetro lecturas de las variaciones de distancia del núcleo del sensor en micras respecto de una posición de referencia. Con la calibración de estos dos elementos se consigue relacionar la medida en bytes obtenida con el prototipo con la medida en micras de la posición del núcleo en el sensor respecto del punto de referencia. De este modo, se consigue que el equipo pueda obtener la variación de la posición del núcleo del sensor en micras y con ello poder realizar la comprobación del funcionamiento del prototipo.

El calibrado del sensor con el prototipo se ha llevado a cabo en el laboratorio. Para ello, se ha hecho uso de un soporte universal de laboratorio, unas galgas, pie de rey, piezas planas de grosor conocido y un ordenador, además del prototipo, el sensor y su equipo.

En primer lugar, se monta el sensor en el soporte universal de laboratorio a una altura fija. Esta altura fija permite que el núcleo se mueva de un extremo al otro del sensor. Además, el sensor está montado de forma tal que la parte más alta corresponde a la zona donde se encuentran los cables de alimentación y señal del sensor. Vamos a llamar a esta zona interna, en contraposición a la parte opuesta del sensor, situada más baja, que llamaremos externa o menos interna. Por ella es por donde se introduce el núcleo del sensor. Véase Figura 8.

Figura 8.Instalación del dendrómetro en el soporte.

Posteriormente se conecta el sensor a su equipo de acondicionamiento y este al microcontrolador. A su vez el microcontrolador es conectado a un ordenador donde se muestran, a través del monitor serie, los valores obtenidos del sensor, véase Figura 9. El software del microcontrolador tiene, para este proceso, desactivadas las funciones de entrada en bajo consumo y el almacenamiento de datos. De este modo se agiliza la toma de datos del sensor, tomándose lectura del sensor cada segundo. Las lecturas de la posición del núcleo en el sensor efectuadas por el microcontrolador son mostradas en bytes, adquiriendo valores desde 0 a 2047, lo que hace un total de 2048 valores, puesto que se toma lectura del sensor con dos entradas analógicas del microcontrolador (1024 + 1024 = 2048). Siendo de 0 a 1023 los valores para las posiciones del núcleo de la parte interna hasta la zona media del sensor, respectivamente medidos con el pin A2. Y siendo de los valores 1024 a 2047 la posición del núcleo desde la parte media del sensor a la externa o menos interna, medidos con el pin A2.

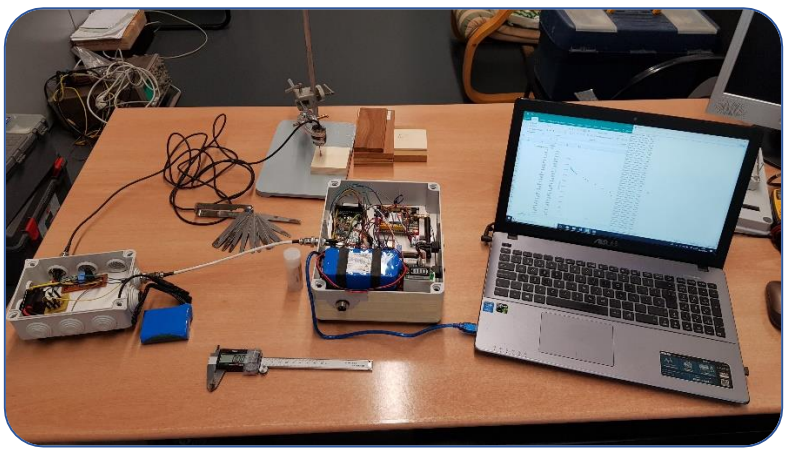


Figura 9. Puesto de calibrado con todos sus elementos preparados.

Llegados a este punto, tenemos todo preparado para llevar a cabo la calibración del LVDT con el sistema desarrollado. Este proceso se lleva a cabo del siguiente modo.

Primero se coloca una pieza plana debajo del sensor. En ella se apoya el extremo del núcleo que sirve de soporte del mismo. A partir de ahí se ajusta la altura del sensor con el soporte universal, haciendo que el núcleo se posicione en la parte que hemos llamado externa o menos interna del sensor, de forma tal, que el valor leído por el microcontrolador sea el valor límite que ofrece el sensor en ese extremo. Esta posición será considerada el origen y se le asigna el valor 0 micras. Véase Figura 10.

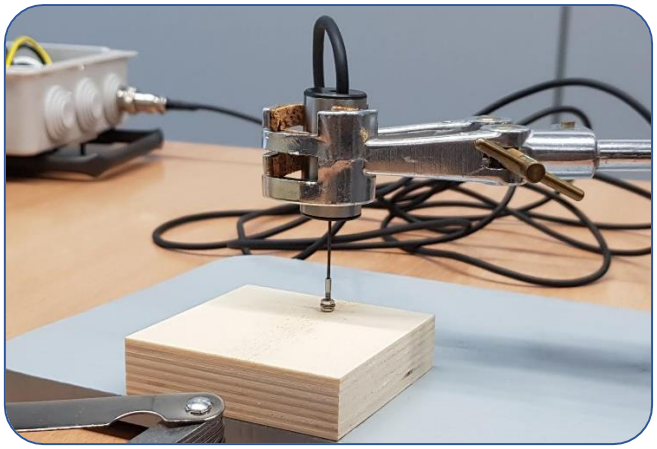


Figura 10. Detalle del dendrómetro (LVDT) con el núcleo ubicado en el origen.

Una vez obtenido el valor cero, se mueve el núcleo hacia arriba, hacia el interior del sensor, una distancia conocida en micras. Este proceso se realiza introduciendo galgas de distinto grosor conocido en micras entre el núcleo del sensor y la placa que marca la posición 0. De este modo se conoce cuanto se ha desplazado el núcleo hacia arriba respecto al origen o posición 0 en valores de distancia medidos en micras. Véase Figura 11.

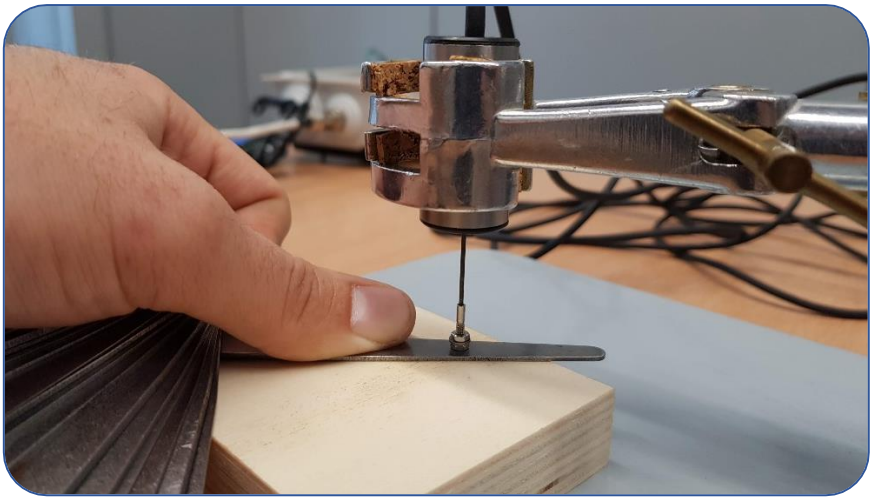


Figura 11. Utilización de las galgas para el calibrado del dendrómetro.

Cuando las galgas resultan ser insuficientes, además se hace uso de piezas de grosor conocido, ya que el grosor de las mismas se ha medido con el pie de rey. Así combinando las piezas y las galgas se toman medidas de múltiples posiciones del núcleo a lo largo del sensor.

Para cada posición del núcleo del sensor es conocida la distancia en micras respecto al origen o posición 0, que se anota en una hoja de cálculo. Junto a cada uno de estos valores se anota la lectura realizada por el microcontrolador y el sensor que está en bytes. De este modo para cada posición hay un valor de lectura en bytes con el microcontrolador del equipo. Véase Figura 12.

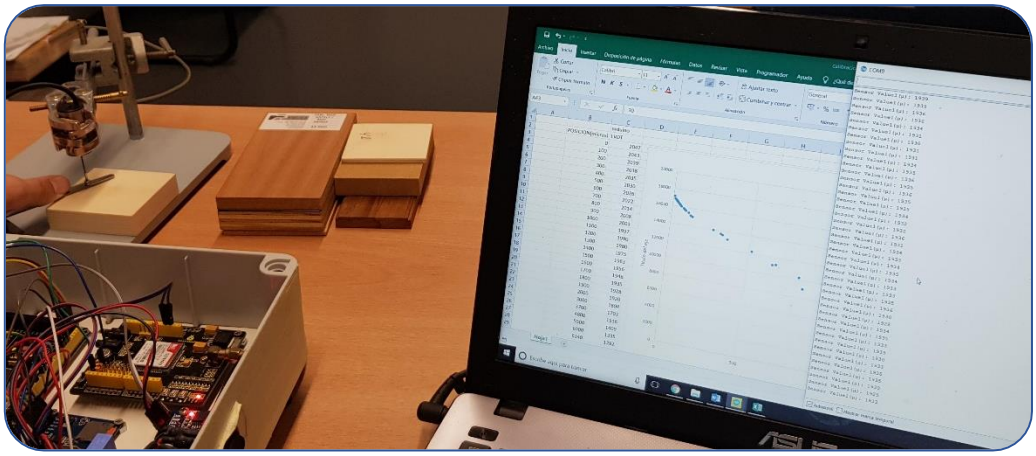


Figura 12. Proceso de calibrado con galgas y anotación de resultados.

Realizando una representación de todas estas medidas se obtiene el resultado mostrado en la Figura 13.

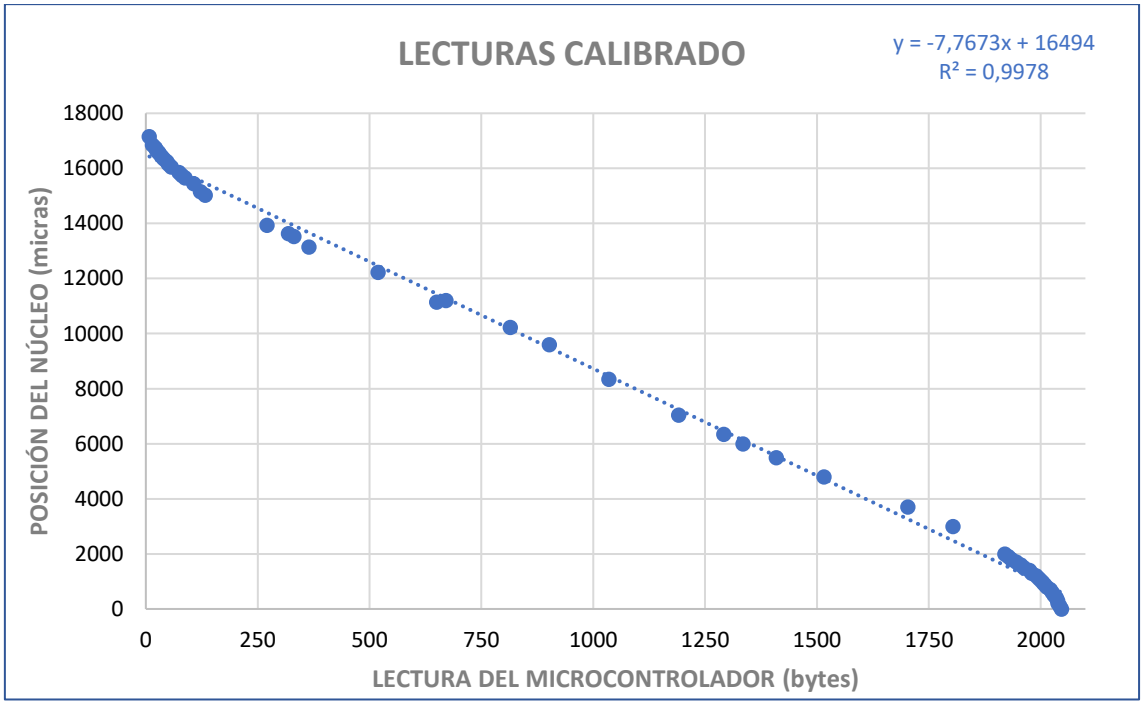


Figura 13. Representación de los resultados obtenidos de la calibración del dendrómetro.

Como se puede observar, de forma general la distribución de los puntos obtenidos en la representación corresponde a una línea recta de ecuación *y=-7,7673x+16494* con un $R^2=0,9978$ donde "*y*" es la posición en micras del núcleo en el interior del sensor y "*x*" es el valor en bytes leído por el microcontrolador.

Examinando esta representación más detalladamente, se ve como, en los extremos de la recta, se produce una desviación respecto de la misma. En cambio, el resto de puntos de la zona central

se ajusta perfectamente a una recta. Este aspecto es característico de este tipo de sensores, dando valores en sus extremos poco fiables.

Debido a ello, y con el fin de evitar lecturas erróneas, se decide ir eliminando progresivamente los valores de los extremos hasta obtener una recta de calibrado lo más ajustada posible tratando de eliminar la menor cantidad posible de valores. Decir que el descarte de valores se ha realizado por igual en ambos extremos y de forma progresiva hasta llegar al punto en el que se estabiliza y no mejora el ajuste de la recta de calibrado obtenida. En la Figura 14 se muestra la representación de los valores de calibración tanto antes (en azul) como después (en verde) de realizar los descartes, así como las rectas de calibrado de ambos casos.

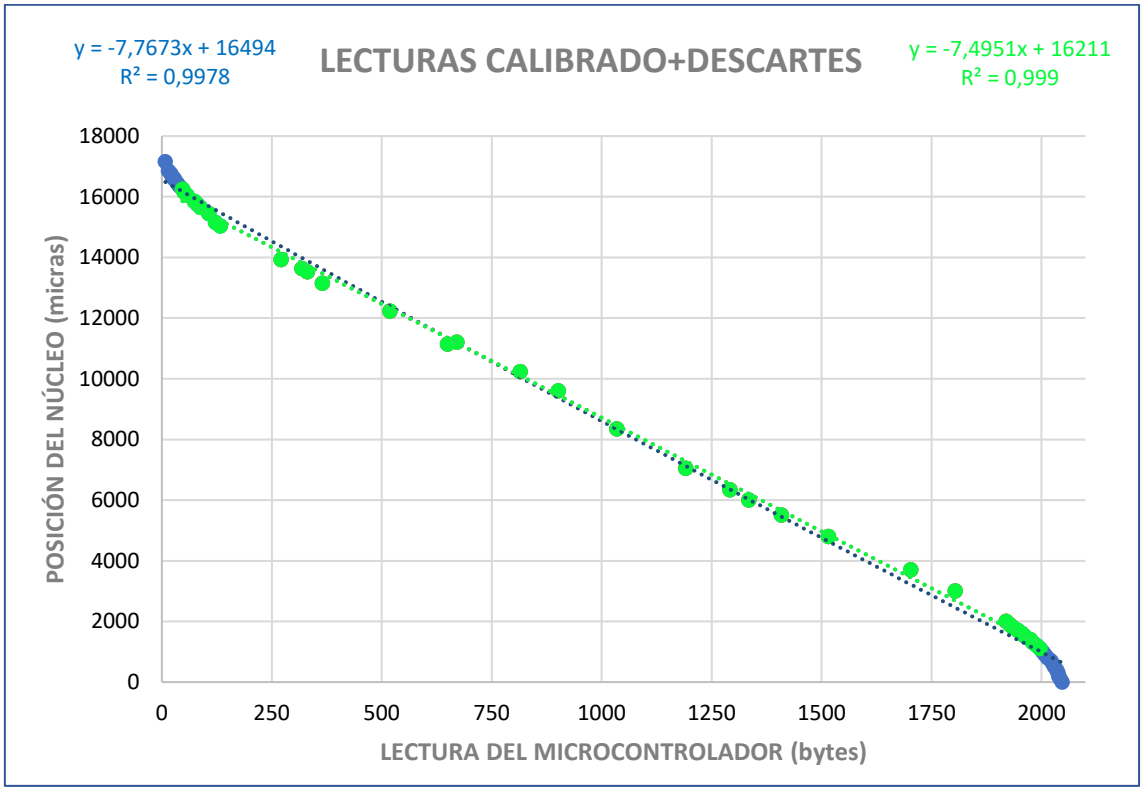


Figura 14. Representación de los resultados obtenidos de la calibración antes (valores en azul) y después (valores en verde) de su ajuste.

Con todo ello se ha obtenido que la recta de calibrado es *y=-7,4951x+16211* con un ajuste *R^2=0,999* superior al obtenido anteriormente sin eliminar las lecturas de los extremos.

Este último ajuste es el que se ha utilizado en el software como calibrado del equipo de monitorización. Para ello se emplea la función map con la que se transforman los valores obtenidos en bytes a los valores ya calibrados en micras. Ello se consigue indicándole a esta función para cada pin de lectura del sensor la equivalencia entre su byte 0 y su posición correspondiente según la recta de calibrado obtenida y la equivalencia entre su byte 1023 y su posición correspondiente según la recta de calibrado obtenida.

Hay que tener en cuenta que los bytes de cada pin analógico de lectura del sensor se corresponden con un byte de los mostrados en la calibración ya que en esta se tienen en cuenta las lecturas de ambos pines como ya se ha comentado anteriormente. Por tanto, para las lecturas realizadas con el pin A1, su byte0 se corresponde con el byte 2047 del calibrado y el byte 1023 se corresponde con el byte 1024 del calibrado. Para las lecturas realizadas con el pin

A2, su byte0 se corresponde con el byte 1023 del calibrado y el byte 1023 se corresponde con el byte 0 del calibrado.

Teniendo en cuenta este aspecto, se procede al cálculo de los valores de posición por el microcontrolador utilizando la función map para mostrar sus lecturas directamente en micras.

Cuando el microcontrolador tome lectura del sensor mediante el pin analógico A1 se tiene que:

-El byte 0 de lectura de este pin es igual al byte 1024 de la calibración (zona intermedia del sensor). Sustituyendo este valor en la recta de calibración se obtiene que la posición equivalente al byte 0 de lectura del pin A1 es 8536 micras.

$$y=-7,4951x1024+16211=8536\mu$$

-El byte 1023 de lectura de este pin es igual al byte 2047 de la calibración, sustituyendo este valor en la recta de calibración se obtiene que la posición equivalente al byte 1023 de lectura del pin A1 es 869 micras.

$$y=-7,4951x2047+16211=869\mu$$

Con ello la función map a emplear para las lecturas efectuadas con el pin A1 será: sensor1=map(sensor1,0,1023,8536,869). Con ella se transforma el byte 0 a 8536 micras y el byte1023 a 869 micras. En base a esta relación de transformación todas las lecturas efectuadas por el microcontrolador con el pin A1 se transformarán a micras.

En cambio, cuando el microcontrolador tome lectura del sensor mediante el pin analógico A2 se tiene que:

-El byte 0 de lectura de este pin es igual al byte 1023 de la calibración (zona intermedia del sensor). Sustituyendo este valor en la recta de calibración se obtiene que la posición equivalente al byte 0 de lectura del pin A2 es 8544 micras.

$$y=-7,4951x1023+16211=8544\mu$$

-El byte 1023 de lectura de este pin es igual al byte 0 de la calibración. Sustituyendo este valor en la recta de calibración, se obtiene que la posición equivalente al byte 1023 de lectura del pin A2 es 16211 micras.

$$y=-7,4951x0+16211=16211\mu$$

Con ello la función map a emplear para las lecturas efectuadas con el pin A2 será: sensor2=map(sensor2,0,1023,8544,16211). Con ella se transforma el byte 0 a 8544 micras y el byte1023 a 16211 micras. En base a esta relación de transformación todas las lecturas efectuadas por el microcontrolador con el pin A2 se transformarán a micras.

De este modo queda el equipo transforma las lecturas en bytes del sistema en diferencia de posición en micras. De esta forma, ya es posible guardar las lecturas del sensor en micras. Hay que tener en cuenta que a pesar de que el equipo esta calibrado el problema anteriormente mencionado de que en las posiciones de los extremos las lecturas son poco fiables persiste, dado que este problema es intrínseco del sensor. Por tanto, a la hora de instalar posteriormente el sensor con el equipo se ha de procurar que el núcleo no se posicione en las zonas más extremas, procurando tomar las lecturas con el núcleo en posiciones más centradas.

Teniendo en cuenta el rango de distancia que es capaz de leer este sistema en micras, y que este realiza la lectura de dicho rango con 2048 bytes se obtiene que la precisión del sistema es de 7,5 micras. A continuación, se muestra el cálculo de la precisión:

$$Precisión\ dendrómetro(\mu) = \frac{16211 - 869}{2048} = 7{,}5\mu$$

Puesto que en campo los dendrómetros se emplean para medir fluctuaciones de los troncos y frutos, y estos suelen tener variaciones diarias de decenas e incluso centenas de micras, la precisión obtenida con el equipo y el sensor es más que suficiente para monitorizar estas variaciones puesto que son capaces de obtener medidas inferiores a la decena, llegando a medir variaciones de 7,5 micras.

2. PUESTA A PRUEBA DEL PROTOTIPO

La puesta a prueba del prototipo en este caso consiste en comprobar el funcionamiento en campo del equipo desarrollado con la solución propuesta. Con ello se verifica que la solución desarrollada no contiene errores que puedan hacerla fallar, y se confirma que cumple los objetivos concretados anteriormente. Además, permite comprobar posibles mejoras a efectuar.

Para poder llevar a cabo la puesta prueba es necesario fabricar un prototipo de la solución propuesta, el cual será sometido a prueba en campo. Puesto que el prototipo es un equipo de monitorización de sensores, además de este, es necesario un sensor a monitorizar por él, para poder llevar a cabo el testeo. En el apartado de implementación se explica cómo se ha llevado a cabo el proceso de fabricación de dicho prototipo, así como del equipo del sensor y la puesta a punto de ambos, con su calibración, para su correcto funcionamiento. En ambos casos se han fabricado los prototipos en base a los componentes detallados en el Anejo I y a los esquemas y conexiones del Anejo II haciendo uso del software de los Anejos III y IV.

En los siguientes apartados se detallan las características principales del prototipo, las cuales se someterán a prueba en este testeo. Además, se detalla información acerca del tipo de sensor seleccionado para ser monitorizado por el prototipo en esta prueba, justificando dicha elección. Posteriormente se indican las condiciones generales del testeo, indicando como se va a testear el prototipo, el lugar en el que se realiza el testeo y la duración del mismo. Por último, se exponen los resultados obtenidos en el proceso de comprobación y en base a estos resultados se dan unas conclusiones del ensayo.

2.1. PROTOTIPO PUESTO A PRUEBA

Como ya se ha indicado, el prototipo de testeo se ha fabricado siguiendo al detalle los esquemas y haciendo uso de los componentes y software de la solución propuesta y desarrollada. En el apartado de implementación se puede ver el proceso de fabricación del mismo.

Como resultado, se ha obtenido un prototipo capaz de leer señales de sensores, que, en este caso, tiene el software preparado para monitorizar un sensor de desplazamiento L.V.D.T. Además, es autónomo, obteniendo su energía de funcionamiento de una placa solar y almacenándola en una batería. También es capaz de entrar en modo de bajo consumo en periodos de inactividad, con el fin de ahorrar energía y aumentar así su autonomía en momentos de radiación solar insuficiente. Por otra parte, es capaz de almacenar los datos obtenidos tanto del sensor de desplazamiento como de los distintos sensores que controlan su estado en internet de las cosas, IoT, más concretamente en la plataforma Thingspeak, lo que permite la

consulta de los datos de forma remota. Además, también guarda estos datos en una tarjeta microSD. Asimismo, es capaz de realizar avisos vía SMS al número de teléfono deseado, bien por falta de energía en su batería o en el sensor, o bien, por exceso de temperatura interna.

Además de estas funciones el prototipo ofrece una buena protección a sus componentes frente a las inclemencias ambientales, aspecto de gran importancia dado que se encontrará en campo donde estas pueden llegar a ser muy acusadas. Igualmente dispone de interruptores para su encendido/apagado y para el control del indicador de batería. Asimismo, las conexiones del equipo del sensor y del módulo solar con el equipo se realizan de forma sencilla y segura mediante el uso conectores. De la misma forma, los equipos y el módulo solar se pueden instalar con facilidad en campo gracias a los soportes desarrollados para ello.

Todas estas características serán las que se someterán a prueba con el testeo de dicho prototipo en campo con un dendrómetro basado en un sensor de desplazamiento L.V.D.T.

2.2. SENSOR MONITORIZADO EN CAMPO

Como se ha comentado, para llevar a cabo el ensayo del prototipo de la solución propuesta es necesario un sensor con el que se comprueba que el prototipo es capaz de monitorizar sensores sin fallos.

En este caso, para la comprobación, se va a monitorizar un dendrómetro, basado en un sensor de desplazamiento lineal L.V.D.T. (de las siglas en inglés Linear Variable Differential Transformer).

Este tipo de sensores es empleado en el campo agronómico, para la monitorización precisa de micro-variaciones del diámetro del tronco o de frutos a lo largo del tiempo. Estas variaciones proporcionan una información precisa del crecimiento y estado hídrico de la planta (ORTUÑO *et al.*, 2010), obtenida a escala diaria de los ciclos de contracción y expansión del diámetro de tronco (KOZLOWSKI, 1967) provocados, principalmente, por cambios del contenido de humedad de la planta(SIMONNEAU *et al.*, 1993). (IRVINE y GRACE, 1997) observaron que más del 90% de las fluctuaciones diarias del diámetro de tronco tienen lugar en los tejidos del floema.

Estas medias, de gran precisión, se han utilizado para el establecimiento de parámetros sensibles a los aportes de agua de riego (IRVINE y GRACE, 1997), siendo la máxima contracción diaria de tronco (MCD; (HUGUET *et al.*, 1992)) el más utilizado como indicador de la intensidad de estrés hídrico(CONEJERO *et al.,* 2010; INTRIGLIOLO y CASTEL, 2007; ORTUÑO *et al.*, 2010).(GOLDHAMER y FERERES, 2004) consideraron que era posible establecer precisos programas de riego basados en las variaciones del diámetro de tronco. Esta nueva estrategia surge a partir del concepto de intensidad de señal (IS), que relaciona los valores de MCD de árboles sin limitaciones hídricas con los valores de árboles sometidos a un estrés(DE LA ROSA *et al.*, 2015), siendo posible el control de la duración e intensidad del estrés al que es sometido el árbol (CONEJERO *et al.,* 2011; CONEJERO *et al.,* 2010; DE LA ROSA *et al.*, 2015; GOLDHAMER y FERERES, 2004; MORIANA *et al.*, 2000; ORTUÑO *et al.*, 2010; PUERTO *et al.*, 2013).

Por tanto, se puede usar un dendrómetro basado en un sensor de tipo L.V.D.T. para definir varios parámetros, los cuales se relacionan con el estado hídrico de la planta. Por ello, este tipo de sensores pueden ser utilizados, como indicador del estado hídrico de la planta, para la programación del riego como ya apuntaba(GONZALEZ-ALTOZANO, 1998).

Con todo ello este tipo de sensores permite por tanto llevar a cabo las siguientes aplicaciones:

-Monitorización en continuo del proceso de crecimiento de la planta.

-Monitorización del estado hídrico de la planta.

-Análisis de la influencia de los factores ambientales en el crecimiento.

-Determinación del inicio y final del periodo de crecimiento.

-Gestión del riego.

Para que este método sea útil debe estar integrado en un sistema de monitorización en tiempo real, como es el caso del dispositivo desarrollado, lo que hace a este tipo de sensores idóneos para el testeo del equipo. Ese hecho unido a todas las virtudes de este tipo de sensores en el campo agronómico, comentadas anteriormente, hace que para llevar a cabo la verificación del funcionamiento del equipo se emplee un dendrómetro como sensor a monitorizar por el prototipo desarrollado.

El dendrómetro seleccionado para tal fin es el dendrómetro basado en el sensor de desplazamiento lineal L.V.D.T., Solartron Metrology, modelo DFg ± 2,5 mm, Bognor Regis, UK, precisión ± 10 µm. Véase apartado 2.1.2.Dendrómetro seleccionado del Anejo I para mayor información acerca de este sensor y su principio de funcionamiento.

Para que el prototipo pueda monitorizar este sensor en el ensayo es necesario que el sensor disponga de un sistema de alimentación y de acondicionamiento de la tensión de alimentación y de la señal propios, cuyos componentes y esquemas se han descrito en los apartados 2.1.Sensor de puesta a prueba, dendrómetro del Anejo I y 2.2.Esquemas específicos del sistema de acondicionamiento de señal del Anejo II. Este equipo desarrollado para el sensor es instalado junto al sensor y al prototipo principal para llevar a cabo el ensayo del mismo. En el apartado de implementación se puede ver el proceso de fabricación del mismo.

Además del equipo del sensor, para el ensayo, se instalará el sensor en un portasensor construido con material invar (aleación de Ni y Fe) y aluminio (coeficiente de dilatación próximo a 0).

2.3. PROCESO DE COMPROBACIÓN. INSTALACIÓN EN CAMPO, PUESTA EN FUNCIONAMIENTO Y PUESTA A PRUEBA

Con el prototipo y el equipo del sensor resultantes de la implementación ya puestos a punto y calibrados, junto con sus accesorios y el dendrómetro, se procede a llevar a cabo la comprobación de la solución propuesta. Para ello se instala el prototipo en campo donde se va a poner en funcionamiento y se va a encargar de la monitorización de la señal del dendrómetro con el que se medirán las fluctuaciones del diámetro del tronco de un cerezo durante un mes. A partir de este proceso de monitorización, y de los datos obtenidos de él, se realiza un análisis en busca de fallos o mejoras a realizar en el prototipo, así como se comprueba si este ha cumplido los objetivos planteados.

La instalación y ensayo se realiza en una parcela dedicada al cultivo de cerezos. Concretamente, en la localidad de Biar, al noroeste de la provincia de Alicante, en la Parcela 13, del Polígono11, zona conocida como El Toll, Véase Figura 15.

La instalación se realiza en el centro de la parcela. La placa solar se instala entre filas de árboles y el prototipo, junto al equipo del sensor, bajo el cerezo en el que se instala a su vez el dendrómetro. En la Figura 16 se muestra la instalación del equipo llevada a cabo.

Una vez instalado el equipo, este se pone en funcionamiento realizando las lecturas del dendrómetro periódicamente cada diez minutos y almacenando los datos obtenidos del dendrómetro y sus sensores internos tanto en la tarjeta microSD como en la plataforma Thingspeak. Además, también se deja preparado para el envío de avisos por batería baja y exceso de temperatura interna, los cuales se realizarán tras la lectura y almacenamiento de datos.

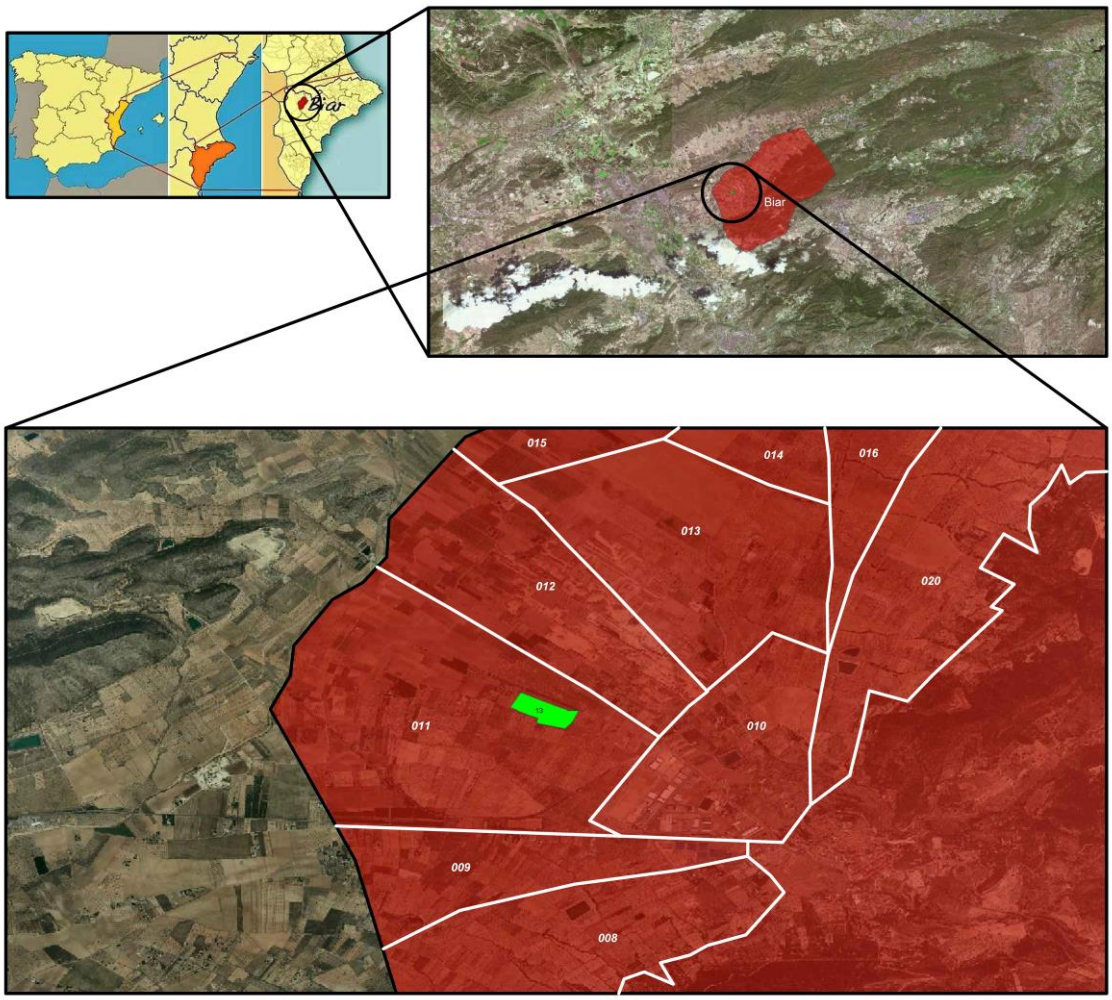


Figura 15. Situación y emplazamiento de la parcela de ensayo. Fuente: elaboración propia.

Con esta puesta en funcionamiento del prototipo se comienza el monitoreo del dendrómetro, lo que da lugar al comienzo del ensayo que se prolongó un mes aproximadamente. Pasado este tiempo se decidió concluir el testeo del mismo debido a que en el tiempo transcurrido ya se había:

-Solucionado los problemas y errores que hacían fallar al prototipo, habiéndose puesto a punto y comprobado el funcionando correcto durante diez días seguidos sin errores.

-Se solucionó un problema derivado de la entrada de agua de lluvia en el equipo del sensor. Tras la reparación y solución se comprobó el correcto funcionamiento durante otra semana.

-Recabado información suficiente para comprobar el cumplimiento de los objetivos concretados para la solución propuesta.

-Obtenido información suficiente para realizar mejoras en el equipo.

① Vista general de la instalación.

② Instalación de los equipos con el dendrómetro.

③ Instalación sensor con portasensor.

④ Instalación del núcleo al tronco del árbol en el sensor.

Figura 16. Instalación de equipos y dendrómetro en campo. Fuente: elaboración propia.

2.4. RESULTADOS DE TESTEO

En el presente apartado se presentan los resultados obtenidos en la puesta a prueba del equipo.

Hay que tener en cuenta que el periodo de puesta a prueba se prolongó aproximadamente un mes. En algunas de las semanas del testeo surgieron problemas y fallos en el equipo que impidieron su correcto funcionamiento y que se fueron solventando a medida que surgían. En otras, tras solventar estos problemas y fallos se continuó con la puesta a prueba del equipo, obteniendo información del prototipo funcionando correctamente para comprobar posteriormente si este cumple los objetivos fijados y para conocer las posibles mejoras se le pueden efectuar a la solución propuesta.

Por ello los resultados del testeo del equipo se van a presentar en dos bloques, el primero de ellos dedicado a los problemas y fallos que han tenido lugar durante el periodo de prueba. El segundo de ellos se dedica a la información obtenida del monitoreo.

2.4.1. Fallos y problemas detectados durante el periodo de prueba del equipo.

A continuación, se presentan tanto fallos o errores detectados en el prototipo, como los problemas surgidos durante la prueba, así como las soluciones llevadas a cabo en cada caso.

El primero de ellos fue que el equipo se había instalado con una placa solar cuya potencia no era suficiente para recargar adecuadamente las baterías. Con la placa inicialmente seleccionada, las baterías se descargaban a mayor velocidad que podían cargarse con la placa, por lo que quedaban descargadas a los pocos días. Esto sucedió porque inicialmente se utilizó una placa de la que se disponía con anterioridad a la realización del proyecto. Debido a este problema se descartó el uso de esa placa y se instaló la placa seleccionada en el apartado de componentes. La placa definitiva se seleccionó teniendo en cuenta los requerimientos del regulador de carga, el cual requiere una potencia en la placa de 40 a 60W. Con esta el problema quedo solucionado, cargándose las baterías adecuadamente.

El segundo problema que surgió fue con la capacidad de la batería del equipo. En esta ocasión la batería instalada en un principio estaba mal seleccionada. Esta funcionaba correctamente, pero no era capaz de almacenar suficiente energía para toda la noche, quedando desabastecido el equipo por las noches. Como solución a este problema se instaló la batería seleccionada en el apartado de componentes. Para ello se tuvo en cuenta que la batería antigua tiene 3Ah de capacidad de almacenaje y que su autonomía en el equipo de monitoreo era de unas 5 horas, con lo que se obtiene un consumo por parte del equipo de aproximadamente 0,6A. A partir de este consumo se estableció la capacidad mínima que debía tener la batería que se iba a seleccionar. Para ello se ha considerado la situación más desfavorable posible, que es en invierno cuando hay 15 horas de oscuridad y 9 horas de luz al día aproximadamente. En este caso es necesario que la batería tenga una autonomía mínima igual a las 15 horas de oscuridad más las horas que tarde en cargarse. En el caso de la batería finalmente seleccionada se tarda 4,5 horas en cargar por lo se considera que al menos ha de tener una autonomía de 19,5 horas. Puesto que el consumo del equipo es de 0.6 A y este ha de tener una autonomía de 19,5 horas se obtiene multiplicando estas variables que la batería ha de tener al menos 11,7 Ah de capacidad de almacenamiento. Con este dato se seleccionó la batería comentada en el Anejo I con una capacidad comercial inmediatamente superior a este valor, en este caso de 12 Ah. De este modo se solucionó este segundo problema.

Solucionados estos dos problemas surgió uno nuevo. Cuando el equipo llevaba aproximadamente un día funcionando dejaba de enviar los datos a la plataforma Thingspeak. Además, cuando esto sucedía, se comprobó que simplemente reiniciando el equipo volvía a funcionar correctamente y se enviaban y guardaban los datos en Thingspeak. Este último hecho hizo sospechar que se trataba de un problema de llenado de la memoria RAM. Para averiguarlo se decidió hacer que el equipo enviara dos variables adicionales: el tiempo que llevaba en funcionamiento y la memoria RAM que le quedaba libre, además de los datos del dendrómetro y de sus sensores de control. Esto se realizó mejorando la programación del equipo (software), véase apartado 1.3.1.Código función principal loop del Anejo III. De este modo se comprobó que cuando se dejaban de almacenar datos en Thingspeak la memoria RAM estaba prácticamente llena, lo que impedía el correcto funcionamiento del equipo de monitoreo. Para solucionar este problema se decidió programar el resetear el equipo automáticamente de tal modo que cuando el equipo de monitoreo lleva 6.500.000 milisegundos ejecutando el programa, este se resetea. Se decidió resetear el equipo transcurrido ese tiempo siendo prudentes, puesto que al superar los 7.000.000 milisegundos la memoria comienza a llegar a su límite y puede dejar de funcionar. De este modo se solucionó este problema y el equipo no volvió a dar fallos de este tipo.

A partir de este momento el equipo comenzó a funcionar prácticamente bien, salvo porque con cierta frecuencia surgían problemas con el módulo DS3231 del que se obtienen la hora y temperatura interna y con el envío de datos a la plataforma Thingspeak. Dado que se trataba de problemas puntuales que se repetían sin motivo aparente, se pensó que podía tratarse de un problema con las conexiones de los cables, concretamente en los pines de los cables. Por ello se decidió soldar directamente con estaño al microcontrolador los cables de las conexiones de los módulos en los que surgía este problema. De este modo se solucionó este problema, por lo que se deduce que los pines del microcontrolador estaban fallando.

Tras resolver todos los problemas mencionados, se continuó con el ensayo y comprobación del equipo. Estuvo funcionando correctamente durante diez días, leyendo datos del dendrómetro y almacenándolos junto al resto de información de control del equipo. Pero en el décimo día surgió un nuevo problema. Se produjeron tormentas estivales, y como consecuencia de estas entró agua en el equipo del sensor, cortocircuitando el sistema de acondicionamiento de señal, así como los conectores. Como consecuencia de ello los datos obtenidos y almacenados por el prototipo del sensor comenzaron a ser erráticos y erróneos. Para solucionar este problema fue necesario desmontar el equipo del sensor, limpiar todos sus componentes y se tuvo que volver fabricar y montar la placa de acondicionamiento del sensor. Se mejoró la estanqueidad de la caja donde se montaba toda la circuitería del equipo del sensor. Una vez hecho esto se volvió a montar todo y se instaló en campo, continuando con el testeo. Tras la reparación del equipo del sensor el equipo de monitorización volvió a funcionar adecuadamente obteniendo datos correctos hasta que se dio por finalizado el testeo del prototipo una semana después.

2.4.2. Información resultante de la puesta a prueba

Entre la información obtenida en el proceso de testeo se encuentran los datos recogidos del dendrómetro por el prototipo, los datos del estado del prototipo y los avisos emitidos por este durante el periodo de prueba en campo.

2.4.2.1. Datos leídos y almacenados del dendrómetro y estado del prototipo

Como resultado dela puesta a prueba del testeo del equipo se tiene que este ha sido capaz de almacenar tanto en la tarjeta microSD como en la plataforma IoT Thingspeak las lecturas del

dendrómetro, temperatura interna del equipo de monitoreo, voltajes de la batería del equipo de monitoreo y del equipo del sensor, voltaje de la placa solar, tiempo en funcionamiento y memoria RAM disponible.

A continuación, se muestran y comentan todos estos datos almacenados en la plataforma Thingspeak, visualizados con la app ThingViewFull, durante el periodo de testeo comprendido entre el 28 de mayo y el 3 de julio del año 2019.

-Dendrómetro

Los primeros datos que se van a tratar son los obtenidos de la monitorización del dendrómetro y almacenados en la plataforma de IoT, Thingspeak. En la Figura 17 se muestran en la parte superior todos los datos del monitoreo del sensor. Además, en la parte inferior izquierda se amplía la visualización de estos mismos datos para el periodo comprendido entre el 27 de mayo y el 6 de junio del año 2019. De igual modo en la parte inferior derecha de la Figura 17 se amplían los datos del periodo comprendido entre el 24 de junio y el 3 de julio.

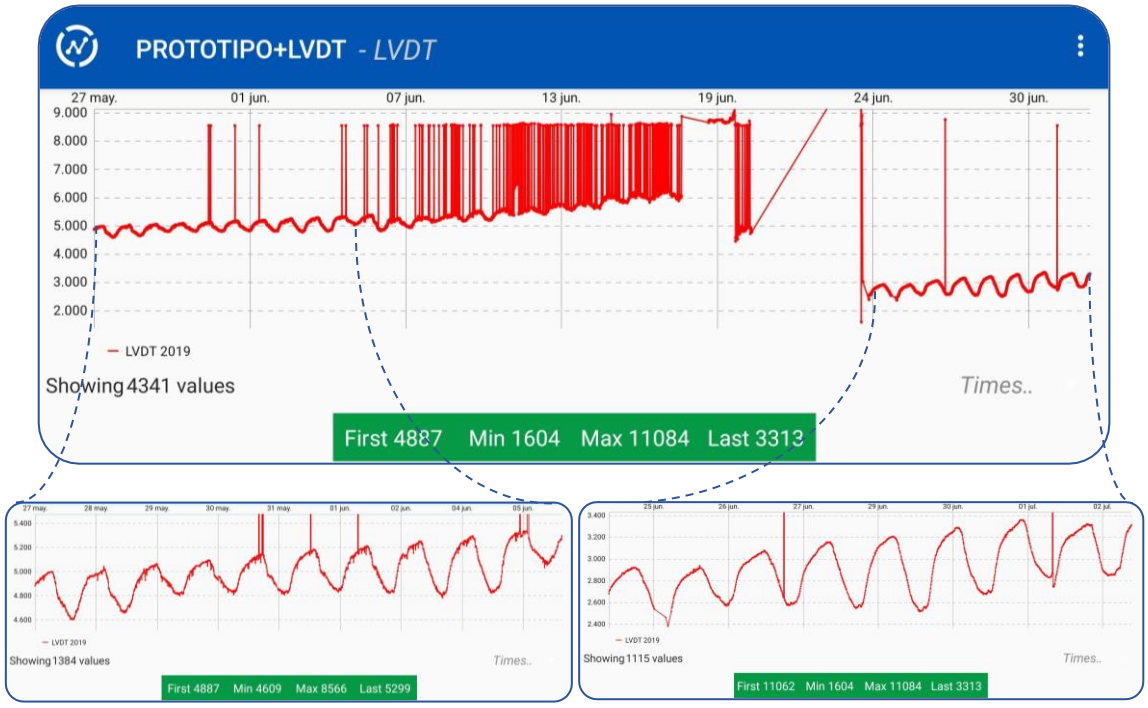


Figura 17. Datos almacenados en Thingspeak, monitorización del dendrómetro.

Como se puede observar en la Figura 17, el equipo ha sido capaz de registrar correctamente las variaciones de diámetro del tronco del cerezo en el que se ha instalado el equipo para su comprobación durante los periodos de tiempo comprendidos entre el 27 de mayo y el 6 de junio, y entre el 24 de junio y el 3 de julio, ambos ampliados en la parte inferior de la Figura 17.

Desde el día 27 de mayo hasta el 6 de junio del año 2019 se ve que los valores del dendrómetro rondan las 5000 micras. En cambió entre el 24 de junio y el 3 de julio rondan las 3000 micras. Este hecho se debe a que las lecturas de los dendrómetros son relativas a una posición concreta. Por ello para su utilización en agronomía hay que hacer continuos reposicionamientos. En este caso concreto, es debido a que el dendrómetro se reinstaló, por lo que el núcleo quedo reubicado en otra posición en este segundo periodo. A priori puede parecer que los datos estén

mal tomados, pero no es así. Además, para hacer uso de estos datos suele emplearse un parámetro denominado Amplitud de Contracción que no es más que la diferencia en valor absoluto que hay entre el valor máximo y el mínimo de cada día. Es decir, para su utilización se emplea la distancia que ha recorrido el núcleo del dendrómetro entre dos momentos, no su distancia al punto considerado como origen, que en este caso es uno de los extremos del sensor y es el valor que se observa en el monitoreo del dendrómetro.

En estos periodos de tiempo, tan solo se dan ciertas lecturas puntuales erróneas, las cuales se han de descartar a la hora de emplear los datos, puesto que se deben a fallos puntuales de conexión del sensor con el equipo o a alguna manipulación del mismo. A pesar de ello, se observa con claridad la dinámica de dilatación y contracción diaria del tronco del cerezo monitorizado. La variación de la Amplitud de Contracción del tronco del cerezo promedio durante estos días es de alrededor de 500 micras. Puede observarse como durante las horas nocturnas el valor de las lecturas aumenta, lo que quiere decir que el núcleo del dendrómetro se adentra en él, hecho que sucede al aumentar el diámetro del troco. Por tanto, en estas horas se registra un aumento del diámetro del troco, debido a que durante las horas nocturnas hay una disminución de la evapotranspiración y el árbol es capaz de tomar más agua del suelo a través de sus raíces que la que pierde por transpiración. En cambio, en las horas diurnas sucede lo contrario, las lecturas disminuyen su valor, lo que significa que el núcleo del dendrómetro es extraído de él, hecho que sucede al disminuir el diámetro del troco. Por tanto, en las horas diurnas se registra una contracción del tronco del árbol que es producida por el aumento de la transpiración, que el árbol no puede compensar con el agua que toma por las raíces. Ello supone la pérdida relativa de agua del árbol y la contracción del tronco.

Puesto que las distancias recorridas diariamente en ambos periodos son similares, y la dinámica de dilatación y contracción diaria del tronco del cerezo registrados con el prototipo y el dendrómetro se corresponde con la registrada habitualmente por este tipo de sensores, a priori, se puede considerar que los datos han sido tomados correctamente entre el los días 27 de mayo y 6 de junio y entre los días 24 de junio y 3 de julio.

En cambio, entre los días 6 y 24 de junio se observan gran cantidad de lecturas caóticas en el monitoreo del dendrómetro. Como se ha visto en el apartado anterior, estos errores se deben a que en los días 6 y 7 de este mes tuvieron lugar tormentas estivales y como consecuencia de ellas se estropeó el equipo del sensor. En este mismo periodo se ve como desde el día 18 al 24 de junio no se registran datos, debido a que el prototipo se desconectó para efectuar las reparaciones del equipo del sensor. Una vez reparado, de nuevo volvió a funcionar adecuadamente. Por lo que presumiblemente, de no ser por el problema derivado de las lluvias la toma de datos se habría realizado en todo el periodo de testeo sin registrar datos erróneos.

Por tanto, se observa que los datos almacenados por el equipo del dendrómetro en el testeo tienen unos valores y dinámicas correspondientes a los de este tipo de sensores, a excepción del periodo comprendido entre en 6 y 24 de junio en el que, a causa de las lluvias, el equipo del sensor se había estropeado y ofrecía una señal errónea.

-Voltajes baterías y módulo solar

Los siguientes datos leídos y almacenados son los datos de voltaje, tanto de la batería del prototipo, de la batería del equipo del sensor, como del módulo solar.

Tanto en la Figura 18 como en la Figura 19 se muestra la evolución de la carga de las baterías de los equipos. En esta evolución se observan los ciclos de carga y descarga. Durante las horas

nocturnas, en las que no hay radiación solar, se produce un descenso de la carga de la batería reflejada por la disminución del voltaje, aunque normalmente sin sobrepasar el límite mínimo establecido de carga(11,05V). Únicamente los días 24 y 25 de junio, la carga de la batería disminuyó por debajo del umbral establecido. Aunque, en las horas diurnas se produce un aumento de su carga con altibajos debidos a la diferencia entre la energía disponible para la carga y el consumo intermitente. En todo caso, en ningún momento se supera el límite máximo de carga que pueden soportar las baterías (12,6V).

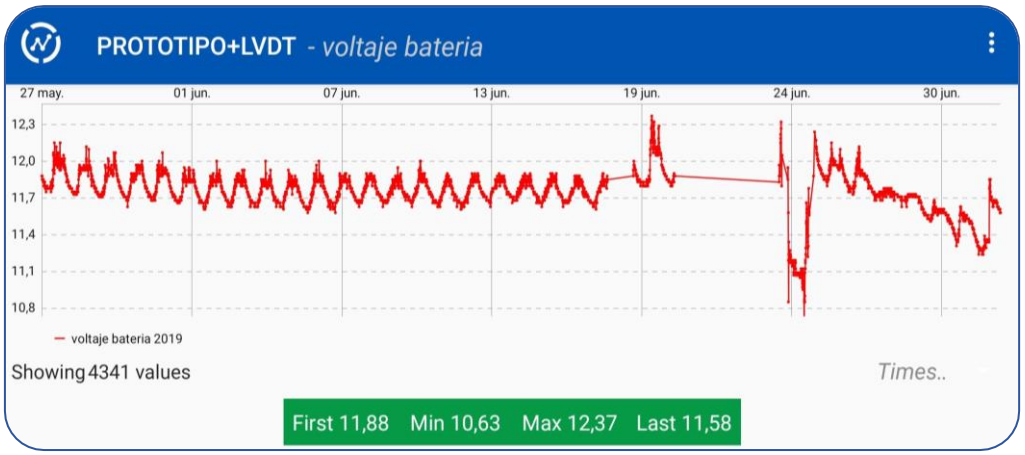


Figura 19. Datos almacenados en Thingspeak, voltaje batería del prototipo.

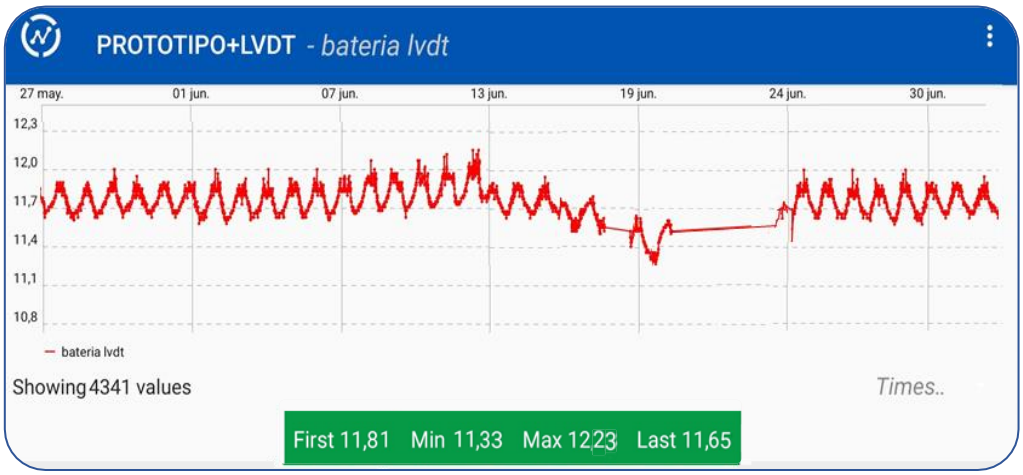


Figura 18.Datos almacenados en Thingspeak, voltaje batería del equipo del sensor.

Cabe comentar que en el periodo de tiempo comprendido entre el día 18 y el 24 de junio apenas hay datos de los niveles de las baterías, así como del módulo solar, debido a que se estaba llevando a cabo la reparación del equipo del sensor como ya se ha comentado.

Si nos fijamos en la Figura 18, los días 24 y 25 de junio, en que se volvió a poner en funcionamiento el prototipo, tras la reparación del equipo del sensor, este tenía los niveles de carga muy bajos, llegando a estar por debajo del mínimo establecido de 11,05V. Este descenso del voltaje de la batería fue debido a la desconexión del módulo solar debido a las labores de reparación comentadas. El prototipo se reinstaló y conectó con el módulo solar en la tarde del 24 estando su nivel de carga por debajo del mínimo. En esa misma tarde se recargó parcialmente la batería con el módulo saliendo de esta situación, pero sin llegar a realizar una recarga completa de la misma. Esto dio lugar a una mayor descarga de la batería en la noche del día 24 al 25 de junio. Como consecuencia de ello, en las primeras horas de la mañana del día 25 de

junio el nivel de batería volvía a estar por debajo del nivel mínimo de carga. A pesar de ello, como se puede apreciar en la Figura 18, ese día la batería aumentó sus niveles de carga una vez salido el sol hasta alcanzar niveles de carga normales, saliendo de esta situación, de nivel de carga excesivamente bajo. No obstante, mientras el equipo ha estado en funcionamiento y conectado al módulo solar, ninguna de las baterías ha llegado a tener niveles de carga excesivamente bajos o se ha descargado por completo, sino todo lo contrario.

En cuanto al registro del voltaje ofrecido por el módulo solar, este se puede visualizar en la Figura 20.

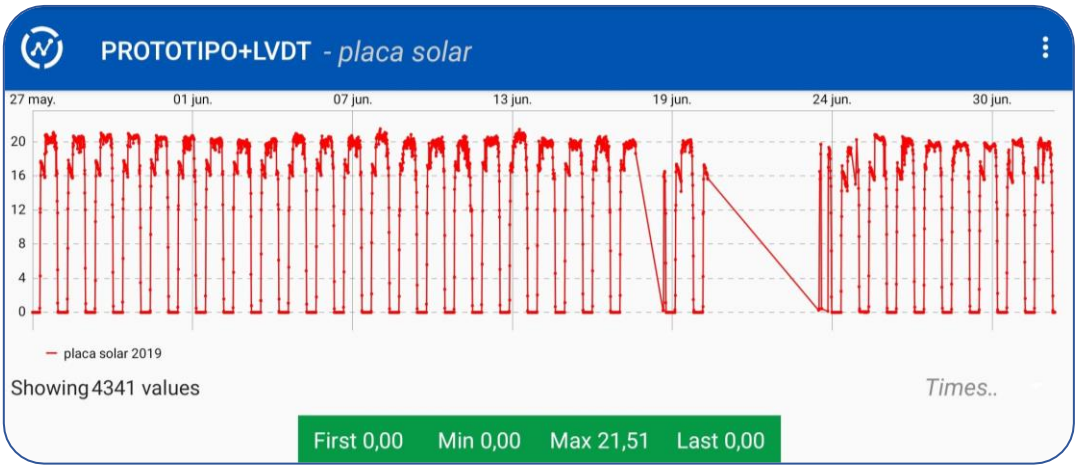


Figura 20. Datos almacenados en Thingspeak, voltaje ofrecido por el módulo solar.

Como se puede observar, el prototipo ha registrado valores de voltaje durante todo el periodo de prueba a excepción, como sucede en el resto de casos, del periodo comprendido entre los días 18 y el 24 de junio, en los cuales se estaba llevando a cabo la reparación del equipo del sensor por lo que el prototipo no estaba en funcionamiento, y no se registraron a penas, valores del voltaje emitido por el módulo solar. Durante el tiempo en el que se registró el voltaje del módulo, se observa que, durante las horas nocturnas el módulo da un voltaje igual a 0, como es lógico puesto que sin luz solar este no genera electricidad. En cambio, en las horas diurnas, el voltaje alcanza valores que van desde 16 a 21V aproximadamente, dependiendo del nivel de carga de la batería y de la radiación solar, siendo el voltaje máximo registrado durante el periodo de prueba de 21,51V. Normalmente, al principio del día se registran un mayor número de veces 16V, voltaje que se lee cuando se está cargando la batería. Puesto que al principio de la mañana se encuentran las baterías menos cargadas, porque por la noche no se han cargado y se ha consumido energía de ellas. Es en este momento cuando más tiempo se está cargándolas y por ello es el momento en que mayor número de veces se registran voltajes de aproximadamente 16V. El resto de horas de luz tan solo se cargan las baterías de vez en cuando para mantener su nivel de carga motivo por el cual se recogen mayoritariamente valores de aproximadamente 21V, voltaje que emite la placa solar a plena luz cuando no se realiza carga.

-Temperatura interna del prototipo

Por otro lado, el prototipo ha registrado la temperatura de su interior, como se puede ver en la Figura 21. Al igual que en los casos anteriores se han registrado valores de temperatura en grados centígrados durante todo el periodo de prueba a excepción del periodo comprendido entre los días 18 y el 24 de junio en los cuales se estaba llevando a cabo la reparación del equipo del sensor. Viendo el resto del periodo de testeo en el que si se registran valores se observa que la temperatura oscila diariamente entre 11 y 40°C aproximadamente con una temperatura mínima registrada de 11°C y una máxima de 43°C. Los valores obtenidos así como su evolución

se corresponden de forma general con las temperaturas típicas registradas en la zona en esa época del año. Estas descienden por la noche alcanzando sus valores mínimos entre las 4 y las 6 de la mañana y ascendiendo estas durante el día y alcanzando su valor máximo entre las 3 y las 6 de la tarde, coincidiendo con la evolución y valores registrados por esta sonda.

-Tiempo de ejecución del programa y memoria RAM libre

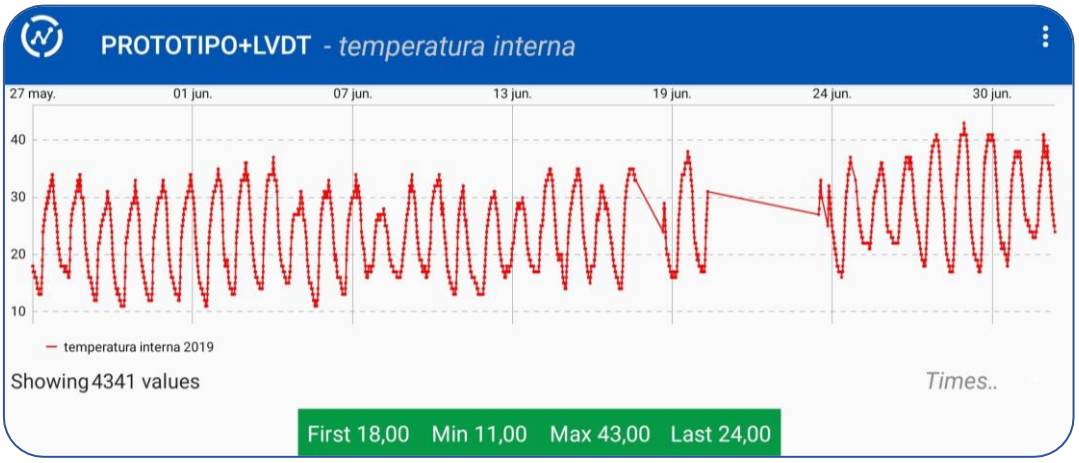


Figura 21. Evolución de la temperatura interna del prototipo. Datos almacenados en Thingspeak,

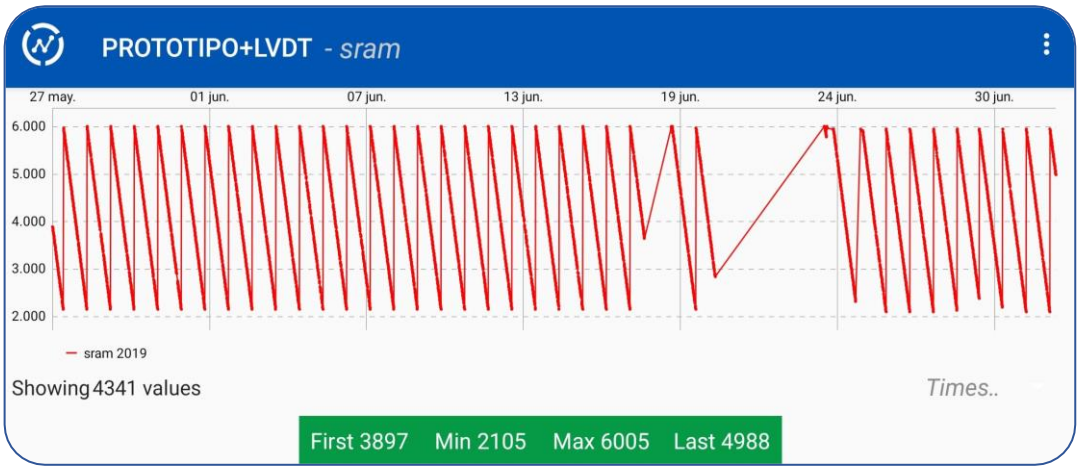


Figura 22.Datos almacenados en Thingspeak, memoria RAM(bytes) del microcontrolador libre.

Las últimas dos variables que se obtienen y almacenan sus valores son, el tiempo en milisegundos que lleva ejecutándose el microcontrolador del equipo y la memoria RAM libre disponible en cada momento. Ambas variables se comenzaron a monitorizar para dar solución al problema provocado por el llenado de la memoria RAM, que producía un bloqueo del microcontrolador, impidiendo su correcto funcionamiento. Con ellas se comprobó que efectivamente el llenado de la memoria RAM provocaba este problema. Para solucionarlo también se hace uso de estas variables. Estas se monitorizan continuamente y cuando han transcurrido 6.500.000 milisegundos de ejecución del programa, el microcontrolador se resetea, liberando la memoria RAM y evitando así el problema. Por ello resulta necesario monitorizar estas dos variables, para poder comprobar en todo momento que no reaparece el problema solucionado con el reseteo del equipo. En la Figura 23 se muestra la evolución de la lectura del tiempo en milisegundos que lleva ejecutándose el programa. De igual modo en la Figura 22 se muestra la evolución de los valores de la memoria RAM disponible.

En el caso del tiempo en ejecución del programa se observa como los valores aumentan siempre entre la toma de un dato y el siguiente aproximadamente 52.000milisegundos, puesto que el tiempo que se tarda en ejecutar el programa es prácticamente el mismo siempre. El valor de este va aumentando siempre en esta cantidad aproximada hasta alcanzar un valor máximo próximo, pero nunca igual o superior a 6.500.000 milisegundos, punto en el cual el programa realiza el reseteo del microcontrolador por lo que vuelve a iniciarse el contador de tiempo a cero. Cuando esto pasa el primer valor guardado es de aproximadamente 2.100milisegundos, que es el tiempo que tarda en iniciarse y almacenar este primer dato. El tiempo que tarda en tomar el primer dato es inferior al del resto porque en él no se ejecuta el programa por completo, queda programa por ejecutar después. En cambio, la cantidad de tiempo en ejecución para tomar el resto de datos es mayor (aproximadamente 52.000milisegundos), puesto que hay que tener en cuenta el tiempo empleado en ejecutar el resto de programa no ejecutado en la toma anterior más el tiempo necesario para ejecutar la parte de programa que lee y almacena el nuevo dato.

Por tanto, a lo largo del testeo se almacenan los datos de tiempo de ejecución del programa, pudiéndose observar los periodos en los que no se resetea el microcontrolador. Los valores del tiempo de ejecución van aumentando poco a poco hasta el momento en que se realiza el reseteo del mismo. Estos se suceden siempre del mismo modo y con valores muy similares a excepción del periodo comprendido entre los días 18 y el 24 de junio en los cuales se estaba llevando a cabo la reparación del equipo del sensor por lo que el prototipo no estaba en funcionamiento y no registraron a penas valores de tiempo de ejecución del programa.

En la Figura 22 se muestra la evolución de los datos de la memoria RAM . En ella se observa que los valores comienzan en aproximadamente 6.000 bytes y van descendiendo de forma continua con cada lectura a razón de unos 30 bytes por ciclo hasta que se lleva a cabo el reinicio del microcontrolador, volviendo de nuevo al valor aproximado de 6Kbytes disponibles inicialmente y volviendo a disminuir del mismo modo. Esta dinámica se observa a lo largo del testeo por igual, salvo entre los días 18 y el 24 de junio en los cuales se estaba llevando a cabo la reparación del equipo del sensor por lo que el prototipo no estaba en funcionamiento y no se registraron los valores de memoria RAM libre. En todo el testeo el valor mínimo registrado para la memoria RAM libre es de 2.105 bytes, lo que da un margen más que suficiente para evitar que el llenado de la memoria RAM produzca los problemas de bloqueo del microcontrolador inicialmente detectados.

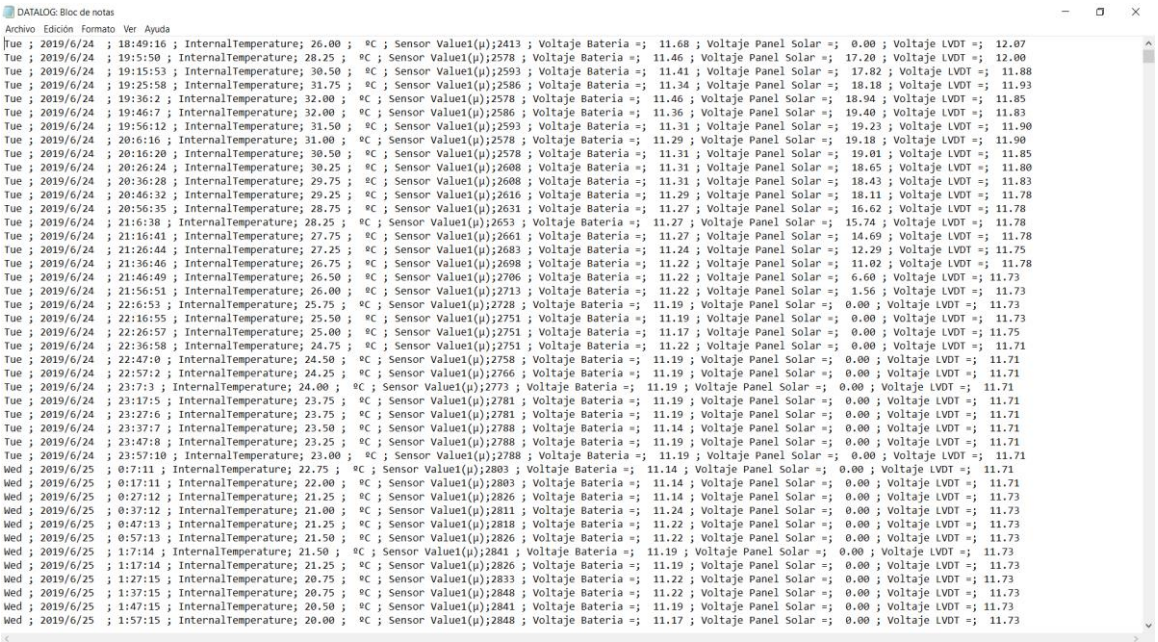
Figura 23.Evolución del tiempo(milisegundos) que lleva el programa del microcontrolador en ejecución. Datos almacenados en Thingspeak

Todos los datos obtenidos en el proceso de testeo anteriormente mostrados han sido obtenidos de la plataforma IoT Thingspeak a través de la app ThingViewFull. Estos valores pueden ser consultados tanto a través de la app como mediante un ordenador a través de la página web https://thingspeak.com/. Si se hace uso de la página web, es posible descargar los datos en formatos json, xml y csv. Además, es posible realizar en la propia página análisis y visualización de los datos con Matlab entre otras aplicaciones.

En cuanto a la información almacenada en la tarjeta microSD, se ha almacenado la fecha y hora en que se realiza la lectura y almacenamiento de cada dato, así como los valores de todas las variables almacenadas en la plataforma Thingspeak a excepción del tiempo que lleva en ejecución el programa y la memoria RAM libre del microcontrolador. Para su almacenamiento en la tarjeta microSD los datos se guardan delimitados por puntos y comas en un archivo de tipo texto llamado Datalog.

Cabe destacar que los datos almacenados en la tarjeta microSD coinciden con los datos almacenados en la plataforma Thingspeak, aunque no a la perfección, puesto que en algunas ocasiones los datos tienen pequeñas diferencias, que en ningún caso llegan a ser significativas. Dichas diferencias en algunos de valores, son debidas a ínfimos errores producidos por el desfase de tiempo que se produce entre la toma de los datos subidos a Thingspeak y la de los datos almacenados posteriormente en la tarjeta microSD. A pesar de ello y como ya se ha indicado, estas diferencias no son significativas por lo que puede considerarse que los datos almacenados coinciden (tarjeta microSD con los de la plataforma Thingspeak). Por ello se puede afirmar, que la información anteriormente detallada sobre los datos obtenidos de Thingspeak es igualmente aplicable a los datos obtenidos en la tarjeta microSD.

En la Figura 24se muestran en el editor de texto, a modo de ejemplo, algunos de los datos almacenados en la tarjeta microSD en el testeo. De este modo se aprecia el modo en que son guardados los datos en el archivo de texto.

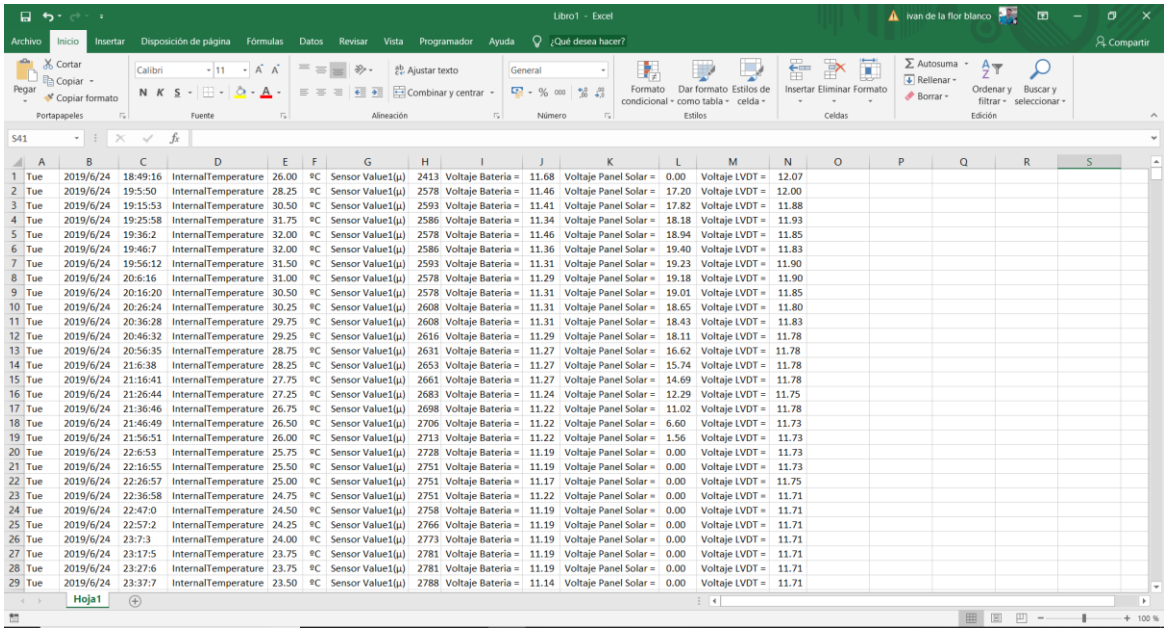

DATALOG: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Tue ; 2019/6/24  ; 18:49:16 ; InternalTemperature; 26.00 ;  ºC ; Sensor Value1(µ);2413 ; Voltaje Bateria =;  11.68 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  12.07
Tue ; 2019/6/24  ; 19:5:50 ; InternalTemperature; 28.25 ;  ºC ; Sensor Value1(µ);2578 ; Voltaje Bateria =;  11.46 ; Voltaje Panel Solar =;  17.20 ; Voltaje LVDT =;  12.00
Tue ; 2019/6/24  ; 19:15:53 ; InternalTemperature; 30.50 ;  ºC ; Sensor Value1(µ);2593 ; Voltaje Bateria =;  11.41 ; Voltaje Panel Solar =;  17.82 ; Voltaje LVDT =;  11.88
Tue ; 2019/6/24  ; 19:25:58 ; InternalTemperature; 31.75 ;  ºC ; Sensor Value1(µ);2586 ; Voltaje Bateria =;  11.34 ; Voltaje Panel Solar =;  18.18 ; Voltaje LVDT =;  11.93
Tue ; 2019/6/24  ; 19:36:2 ; InternalTemperature; 32.00 ;  ºC ; Sensor Value1(µ);2578 ; Voltaje Bateria =;  11.46 ; Voltaje Panel Solar =;  18.94 ; Voltaje LVDT =;  11.85
Tue ; 2019/6/24  ; 19:46:7 ; InternalTemperature; 32.00 ;  ºC ; Sensor Value1(µ);2586 ; Voltaje Bateria =;  11.36 ; Voltaje Panel Solar =;  19.40 ; Voltaje LVDT =;  11.83
Tue ; 2019/6/24  ; 19:56:12 ; InternalTemperature; 31.50 ;  ºC ; Sensor Value1(µ);2593 ; Voltaje Bateria =;  11.31 ; Voltaje Panel Solar =;  19.23 ; Voltaje LVDT =;  11.90
Tue ; 2019/6/24  ; 20:6:16 ; InternalTemperature; 31.00 ;  ºC ; Sensor Value1(µ);2578 ; Voltaje Bateria =;  11.29 ; Voltaje Panel Solar =;  19.18 ; Voltaje LVDT =;  11.90
Tue ; 2019/6/24  ; 20:16:20 ; InternalTemperature; 30.50 ;  ºC ; Sensor Value1(µ);2578 ; Voltaje Bateria =;  11.31 ; Voltaje Panel Solar =;  19.01 ; Voltaje LVDT =;  11.85
Tue ; 2019/6/24  ; 20:26:24 ; InternalTemperature; 30.25 ;  ºC ; Sensor Value1(µ);2608 ; Voltaje Bateria =;  11.31 ; Voltaje Panel Solar =;  18.65 ; Voltaje LVDT =;  11.80
Tue ; 2019/6/24  ; 20:36:28 ; InternalTemperature; 29.75 ;  ºC ; Sensor Value1(µ);2608 ; Voltaje Bateria =;  11.31 ; Voltaje Panel Solar =;  18.43 ; Voltaje LVDT =;  11.83
Tue ; 2019/6/24  ; 20:46:32 ; InternalTemperature; 29.25 ;  ºC ; Sensor Value1(µ);2616 ; Voltaje Bateria =;  11.29 ; Voltaje Panel Solar =;  18.11 ; Voltaje LVDT =;  11.78
Tue ; 2019/6/24  ; 20:56:35 ; InternalTemperature; 28.75 ;  ºC ; Sensor Value1(µ);2631 ; Voltaje Bateria =;  11.27 ; Voltaje Panel Solar =;  16.62 ; Voltaje LVDT =; 11.78
Tue ; 2019/6/24  ; 21:6:38 ; InternalTemperature; 28.25 ;  ºC ; Sensor Value1(µ);2653 ; Voltaje Bateria =;  11.27 ; Voltaje Panel Solar =;  15.74 ; Voltaje LVDT =;  11.78
Tue ; 2019/6/24  ; 21:16:41 ; InternalTemperature; 27.75 ;  ºC ; Sensor Value1(µ);2661 ; Voltaje Bateria =;  11.27 ; Voltaje Panel Solar =;  14.69 ; Voltaje LVDT =;  11.78
Tue ; 2019/6/24  ; 21:26:44 ; InternalTemperature; 27.25 ;  ºC ; Sensor Value1(µ);2683 ; Voltaje Bateria =;  11.24 ; Voltaje Panel Solar =;  12.29 ; Voltaje LVDT =; 11.75
Tue ; 2019/6/24  ; 21:36:46 ; InternalTemperature; 26.75 ;  ºC ; Sensor Value1(µ);2698 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  11.02 ; Voltaje LVDT =;  11.78
Tue ; 2019/6/24  ; 21:46:49 ; InternalTemperature; 26.50 ;  ºC ; Sensor Value1(µ);2706 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  6.60 ; Voltaje LVDT =; 11.73
Tue ; 2019/6/24  ; 21:56:51 ; InternalTemperature; 26.00 ;  ºC ; Sensor Value1(µ);2713 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  1.56 ; Voltaje LVDT =;  11.73
Tue ; 2019/6/24  ; 22:6:53 ; InternalTemperature; 25.75 ;  ºC ; Sensor Value1(µ);2728 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Tue ; 2019/6/24  ; 22:16:55 ; InternalTemperature; 25.50 ;  ºC ; Sensor Value1(µ);2751 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Tue ; 2019/6/24  ; 22:26:57 ; InternalTemperature; 25.00 ;  ºC ; Sensor Value1(µ);2751 ; Voltaje Bateria =;  11.17 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =; 11.75
Tue ; 2019/6/24  ; 22:36:58 ; InternalTemperature; 24.75 ;  ºC ; Sensor Value1(µ);2751 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 22:47:0 ; InternalTemperature; 24.50 ;  ºC ; Sensor Value1(µ);2758 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 22:57:2 ; InternalTemperature; 24.25 ;  ºC ; Sensor Value1(µ);2766 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:7:3 ; InternalTemperature; 24.00 ;  ºC ; Sensor Value1(µ);2773 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:17:5 ; InternalTemperature; 23.75 ;  ºC ; Sensor Value1(µ);2781 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:27:6 ; InternalTemperature; 23.75 ;  ºC ; Sensor Value1(µ);2781 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:37:7 ; InternalTemperature; 23.50 ;  ºC ; Sensor Value1(µ);2788 ; Voltaje Bateria =;  11.14 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:47:8 ; InternalTemperature; 23.25 ;  ºC ; Sensor Value1(µ);2788 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Tue ; 2019/6/24  ; 23:57:10 ; InternalTemperature; 23.00 ;  ºC ; Sensor Value1(µ);2788 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Wed ; 2019/6/25  ; 0:7:11 ; InternalTemperature; 22.75 ;  ºC ; Sensor Value1(µ);2803 ; Voltaje Bateria =;  11.14 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Wed ; 2019/6/25  ; 0:17:11 ; InternalTemperature; 22.00 ;  ºC ; Sensor Value1(µ);2803 ; Voltaje Bateria =;  11.14 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.71
Wed ; 2019/6/25  ; 0:27:12 ; InternalTemperature; 21.25 ;  ºC ; Sensor Value1(µ);2826 ; Voltaje Bateria =;  11.14 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 0:37:12 ; InternalTemperature; 21.00 ;  ºC ; Sensor Value1(µ);2811 ; Voltaje Bateria =;  11.24 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 0:47:13 ; InternalTemperature; 21.25 ;  ºC ; Sensor Value1(µ);2818 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 0:57:13 ; InternalTemperature; 21.50 ;  ºC ; Sensor Value1(µ);2826 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 1:7:14 ; InternalTemperature; 21.50 ;  ºC ; Sensor Value1(µ);2841 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 1:17:14 ; InternalTemperature; 21.25 ;  ºC ; Sensor Value1(µ);2826 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 1:27:15 ; InternalTemperature; 20.75 ;  ºC ; Sensor Value1(µ);2833 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =; 11.73
Wed ; 2019/6/25  ; 1:37:15 ; InternalTemperature; 20.75 ;  ºC ; Sensor Value1(µ);2848 ; Voltaje Bateria =;  11.22 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
Wed ; 2019/6/25  ; 1:47:15 ; InternalTemperature; 20.50 ;  ºC ; Sensor Value1(µ);2841 ; Voltaje Bateria =;  11.19 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =; 11.73
Wed ; 2019/6/25  ; 1:57:15 ; InternalTemperature; 20.00 ;  ºC ; Sensor Value1(µ);2848 ; Voltaje Bateria =;  11.17 ; Voltaje Panel Solar =;  0.00 ; Voltaje LVDT =;  11.73
```

Figura 24.Muestra de datos almacenados en la tarjeta microSD, visualización en editor de texto.

Además, en la Figura 25 se muestran estos mismos datos, pero en este caso abriendo el archivo con una hoja de cálculo. En ella se muestra cómo quedan dispuestos los datos para trabajar con ellos cuando se abren con una hoja de cálculo.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Tue	2019/6/24	18:49:16	InternalTemperature	26.00	ºC	Sensor Value1(µ)	2413	Voltaje Bateria =	11.68	Voltaje Panel Solar =	0.00	Voltaje LVDT =	12.07
2	Tue	2019/6/24	19:5:50	InternalTemperature	28.25	ºC	Sensor Value1(µ)	2578	Voltaje Bateria =	11.46	Voltaje Panel Solar =	17.20	Voltaje LVDT =	12.00
3	Tue	2019/6/24	19:15:53	InternalTemperature	30.50	ºC	Sensor Value1(µ)	2593	Voltaje Bateria =	11.41	Voltaje Panel Solar =	17.82	Voltaje LVDT =	11.88
4	Tue	2019/6/24	19:25:58	InternalTemperature	31.75	ºC	Sensor Value1(µ)	2586	Voltaje Bateria =	11.34	Voltaje Panel Solar =	18.18	Voltaje LVDT =	11.93
5	Tue	2019/6/24	19:36:2	InternalTemperature	32.00	ºC	Sensor Value1(µ)	2578	Voltaje Bateria =	11.46	Voltaje Panel Solar =	18.94	Voltaje LVDT =	11.85
6	Tue	2019/6/24	19:46:7	InternalTemperature	32.00	ºC	Sensor Value1(µ)	2586	Voltaje Bateria =	11.36	Voltaje Panel Solar =	19.40	Voltaje LVDT =	11.83
7	Tue	2019/6/24	19:56:12	InternalTemperature	31.50	ºC	Sensor Value1(µ)	2593	Voltaje Bateria =	11.31	Voltaje Panel Solar =	19.23	Voltaje LVDT =	11.90
8	Tue	2019/6/24	20:6:16	InternalTemperature	31.00	ºC	Sensor Value1(µ)	2578	Voltaje Bateria =	11.29	Voltaje Panel Solar =	19.18	Voltaje LVDT =	11.90
9	Tue	2019/6/24	20:16:20	InternalTemperature	30.50	ºC	Sensor Value1(µ)	2578	Voltaje Bateria =	11.31	Voltaje Panel Solar =	19.01	Voltaje LVDT =	11.85
10	Tue	2019/6/24	20:26:24	InternalTemperature	30.25	ºC	Sensor Value1(µ)	2608	Voltaje Bateria =	11.31	Voltaje Panel Solar =	18.65	Voltaje LVDT =	11.80
11	Tue	2019/6/24	20:36:28	InternalTemperature	29.75	ºC	Sensor Value1(µ)	2608	Voltaje Bateria =	11.31	Voltaje Panel Solar =	18.43	Voltaje LVDT =	11.83
12	Tue	2019/6/24	20:46:32	InternalTemperature	29.25	ºC	Sensor Value1(µ)	2616	Voltaje Bateria =	11.29	Voltaje Panel Solar =	18.11	Voltaje LVDT =	11.78
13	Tue	2019/6/24	20:56:35	InternalTemperature	28.75	ºC	Sensor Value1(µ)	2631	Voltaje Bateria =	11.27	Voltaje Panel Solar =	16.62	Voltaje LVDT =	11.78
14	Tue	2019/6/24	21:6:38	InternalTemperature	28.25	ºC	Sensor Value1(µ)	2653	Voltaje Bateria =	11.27	Voltaje Panel Solar =	15.74	Voltaje LVDT =	11.78
15	Tue	2019/6/24	21:16:41	InternalTemperature	27.75	ºC	Sensor Value1(µ)	2661	Voltaje Bateria =	11.27	Voltaje Panel Solar =	14.69	Voltaje LVDT =	11.78
16	Tue	2019/6/24	21:26:44	InternalTemperature	27.25	ºC	Sensor Value1(µ)	2683	Voltaje Bateria =	11.24	Voltaje Panel Solar =	12.29	Voltaje LVDT =	11.75
17	Tue	2019/6/24	21:36:46	InternalTemperature	26.75	ºC	Sensor Value1(µ)	2698	Voltaje Bateria =	11.22	Voltaje Panel Solar =	11.02	Voltaje LVDT =	11.78
18	Tue	2019/6/24	21:46:49	InternalTemperature	26.50	ºC	Sensor Value1(µ)	2706	Voltaje Bateria =	11.22	Voltaje Panel Solar =	6.60	Voltaje LVDT =	11.73
19	Tue	2019/6/24	21:56:51	InternalTemperature	26.00	ºC	Sensor Value1(µ)	2713	Voltaje Bateria =	11.22	Voltaje Panel Solar =	1.56	Voltaje LVDT =	11.73
20	Tue	2019/6/24	22:6:53	InternalTemperature	25.75	ºC	Sensor Value1(µ)	2728	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.73
21	Tue	2019/6/24	22:16:55	InternalTemperature	25.50	ºC	Sensor Value1(µ)	2751	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.73
22	Tue	2019/6/24	22:26:57	InternalTemperature	25.00	ºC	Sensor Value1(µ)	2751	Voltaje Bateria =	11.17	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.75
23	Tue	2019/6/24	22:36:58	InternalTemperature	24.75	ºC	Sensor Value1(µ)	2751	Voltaje Bateria =	11.22	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
24	Tue	2019/6/24	22:47:0	InternalTemperature	24.50	ºC	Sensor Value1(µ)	2758	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
25	Tue	2019/6/24	22:57:2	InternalTemperature	24.25	ºC	Sensor Value1(µ)	2766	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
26	Tue	2019/6/24	23:7:3	InternalTemperature	24.00	ºC	Sensor Value1(µ)	2773	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
27	Tue	2019/6/24	23:17:5	InternalTemperature	23.75	ºC	Sensor Value1(µ)	2781	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
28	Tue	2019/6/24	23:27:6	InternalTemperature	23.75	ºC	Sensor Value1(µ)	2781	Voltaje Bateria =	11.19	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71
29	Tue	2019/6/24	23:37:7	InternalTemperature	23.50	ºC	Sensor Value1(µ)	2788	Voltaje Bateria =	11.14	Voltaje Panel Solar =	0.00	Voltaje LVDT =	11.71

Figura 25.Muestra de datos almacenados en la tarjeta microSD, visualización en hoja de cálculo.

2.4.2.2. Información del envío de avisos

Entre la información obtenida durante el periodo de ensayo se encuentran, además de los datos recogidos por el prototipo, tanto del dendrómetro como de su propio estado, los avisos emitidos por este. Durante el periodo de prueba se recibieron avisos del prototipo principalmente por exceso de temperatura interna, aunque también se recibieron por falta de voltaje en la batería del prototipo. En cambio, no se recibieron avisos de la batería del sensor debido a que no se registraron datos de su voltaje por debajo del umbral establecido de 11,05V.

En cuanto a los avisos por temperatura interna excesiva el prototipo está programado para mandar el aviso cuando la temperatura interna es superior a 40°C. El aviso se realiza mediante el envío de un SMS en el cual se muestra el texto ”ALERTA, ALTA TEMPERATURA INTERNA, TEMPERATURA(°C)” seguido de la temperatura que registra en ese momento el prototipo en su interior.

Este mensaje de aviso se recibió un total de 46 veces durante el periodo de prueba, aproximadamente entre las 13:30 y las 18:00, que suelen ser las horas de máximas temperaturas en el lugar donde se ha realizado el ensayo y en las fechas en que se realizó. Del total de avisos recibidos por alta temperatura se concentran en unos pocos días seguidos, 3 se recibieron el 28 de junio, 30 el 29 de junio, 11 el 30 de junio y 2 el día 2 de julio. Estos días coincidieron con el fenómeno denominado ola de calor, que hizo que se alcanzaran altas temperaturas. En la Figura 26 se muestran algunos de los mensajes recibidos por exceso de temperatura.

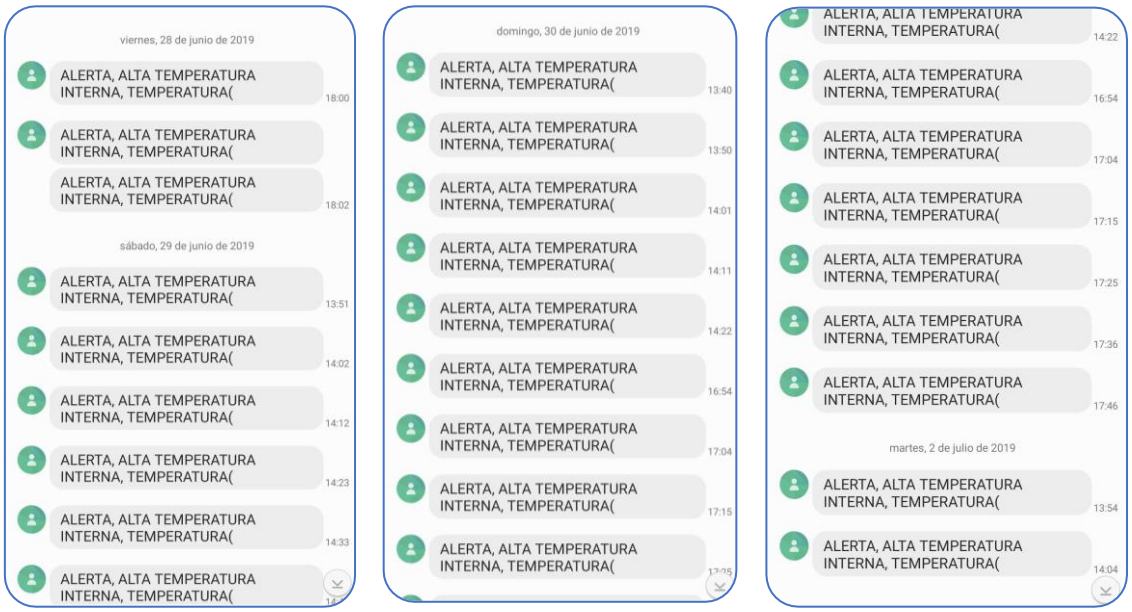


Figura 26. Mensajes de aviso recibidos en testeo por alta temperatura interna.

Como puede observarse en la Figura 26, los mensajes de aviso por alta temperatura se enviaron incompletos, dado que falta el final del mensaje junto con el valor de la temperatura a la que se encuentra el prototipo. Este fallo se debe a que el mensaje es demasiado extenso, por lo que es recomendable modificarlo haciéndolo más escueto.

Por otro lado, el prototipo está preparado para enviar avisos por batería baja cuando el voltaje de esta es inferior a 11,05V tanto si sucede esto con la batería del sensor como con la del prototipo. Durante el periodo de ensayo, tan solo se realizaron avisos por batería baja en el prototipo. Como ya se ha comentado, no se recibieron avisos de la batería del sensor ya que en este caso no fue necesario. Los avisos recibidos por SMS en el testeo por batería baja en el prototipo contienen el mensaje ”ALERTA, BATERÍA CENTRALITA BAJA, VOLTAJE(V):” seguido del valor del voltaje de la batería en ese momento. El envío de este tipo de avisos tan solo tuvo lugar los días 24 y 25 de junio, véase Figura 27.

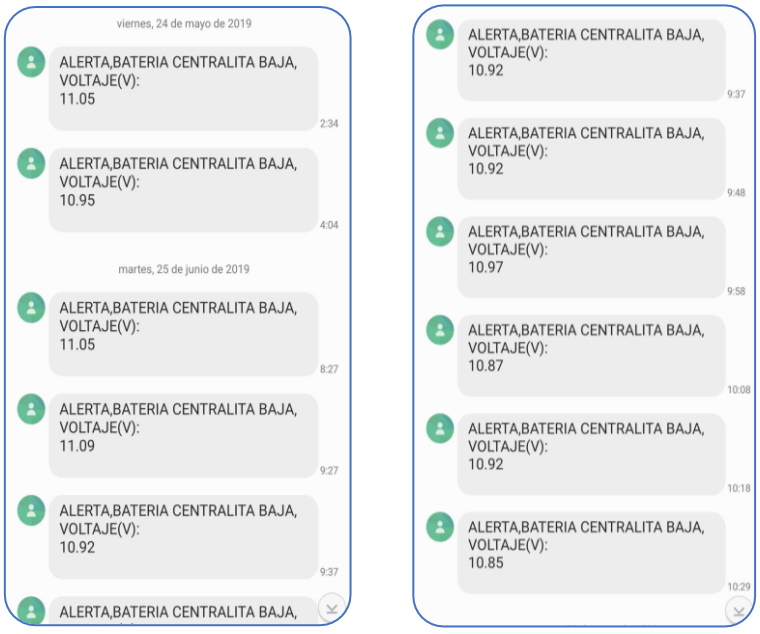


Figura 27. Mensajes de aviso recibidos durante el periodo de prueba por nivel de carga de la batería del prototipo bajo.

Estos avisos se enviaron correctamente, puesto que la batería del prototipo se encontraba descargada, con un nivel de carga inferior al mínimo establecido de 11,05V. Esta batería se encontraba descargada no por un fallo del prototipo o del sistema de carga, sino porque se desconectó el prototipo para realizar una reparación en el equipo del sensor, con lo que no se cargaron las baterías durante un tiempo. Como consecuencia cuando se volvió a instalar y poner en marcha el prototipo para continuar con su testeo este tenía la batería descargada, haciendo que el prototipo enviara los mensajes de aviso.

Los mensajes de aviso se recibieron el día 24 de junio al principio de la tarde y al día siguiente por la mañana. Esto es debido a que el prototipo se reinstaló y conectó con el módulo solar en la tarde del 24 con la batería descargada a un nivel de carga por debajo del mínimo, por lo que se recibieron los avisos pertinentes a esta situación. En esa misma tarde se recargó la batería con el módulo saliendo de esta situación, pero sin llegar a realizar una recarga completa de la batería. Esto dio lugar a una descarga de la batería en la noche del día 24 al 25 de junio. Como consecuencia, y puesto que durante la noche se habían usado las baterías con la consecuente descarga, en las primeras horas de la mañana del día 25 de junio el nivel de batería volvía a estar por debajo del nivel mínimo de carga. Por tanto, se volvieron a realizar los pertinentes avisos. A pesar de ello, ese día la batería aumentó el nivel de carga una vez salido el sol hasta alcanzar niveles normales, saliendo de la situación de nivel de carga excesivamente bajo y no volviendo a enviar más avisos.

De no haber sido por el problema comentado seguramente no se habría realizado ningún aviso de este tipo en el testeo puesto que como se ha visto con los datos de nivel de carga de la batería del prototipo, véase apartado anterior las cargas y descargas de la batería se realizan adecuadamente siempre y cuando estén conectados y funcionando los elementos de control de carga y de descarga.

2.5. CONCLUSIÓN DEL TESTEO

La prueba de control del funcionamiento del equipo desarrollado se ha realizado con éxito, a pesar de que han surgido diversos contratiempos, obteniendo las siguientes conclusiones de él:

-Con el testeo se han detectado problemas y fallos en el prototipo, como son los fallos en el dimensionado de la alimentación, problemas de bloqueo del microcontrolador por llenado de la memoria RAM, fallos en conexiones y falta de estanqueidad en el equipo del sensor. Todos estos problemas se han solucionado, y al final del testeo se ha verificado que el prototipo ya no contiene errores y funciona perfectamente.

-Se ha comprobado que el prototipo cumple con sus objetivos, puesto que, ha sido capaz de realizar con éxito la lectura y almacenamiento de los valores de todas las variables deseadas, incluida la señal del dendrómetro, tanto en la plataforma Thingspeak como en la tarjeta microSD. Se considera que estos se han registrado satisfactoriamente porque se han obtenido datos con valores correctos.

Los datos obtenidos se han podido consultar de forma remota, tanto mediante el uso de un ordenador a través de la página web de la plataforma Thingspeak, como con la app ThingViewFull con la que se consulta esta misma plataforma.

El prototipo también ha sido capaz de realizar avisos mediante el envío de SMS en los momentos en que se alcanzaron los valores umbral programados, tanto por exceso de temperatura, como por falta de carga en la batería.

Asimismo, se ha podido comprobar que el prototipo es realmente autónomo, puesto que una vez instalado no ha sido necesario recargar las baterías del mismo. Consultando los datos del apartado 2.4.2.1. se comprueba como el sistema ha ido recargando las baterías, así como este no ha fallado en ningún momento por falta de energía. Incluso cuando se instaló por la tarde con las baterías descargadas, ha sido capaz de funcionar, llegando a recargar las baterías por completo.

Adicionalmente, se ha comprobado que el equipo entra y sale de su modo de bajo consumo una vez transcurridos los diez minutos entre un registro de datos y el siguiente.

Igualmente, se ha comprobado que el prototipo es capaz de proteger sus componentes frente a las inclemencias ambientales, puesto que ha soportado tanto tormentas estivales como olas de calor sin que se vea afectado ninguno de sus componentes. En cambio, no ha sucedido lo mismo con el equipo del sensor, que emplea otro tipo de caja, al cual le entro agua y estropeó algunos componentes. Por ello se debe utilizar el mismo tipo de caja que la instalada en el prototipo para ambos equipos. Además, es necesario emplear conectores que ofrezcan una buena estanqueidad frente al agua.

Por último, al llevar a cabo la instalación del prototipo y sensores se ha podido comprobar que esta se puede llevar a cabo con facilidad en campo.

-De este periodo de ensayo se plantean diversas mejoras. Entre ellas, se encuentra hacer uso de cajas y conectores de gran estanqueidad para evitar problemas como los surgidos en el testeo con el equipo del sensor. Otra mejora a poder realizar sería instalar un módulo GSM/GPRS sim900 capaz de encenderse con el microcontrolador por software, para evitar de este modo tener que abrir el equipo cada vez que este se conecta para encender dicho módulo. Por otro lado, también se le podría instalar una pantalla lcd en la que visualizar los datos que se están registrando y junto con algunos botones poder configurar las distintas funciones del equipo.

A modo de conclusión final, podemos afirmar que con el ensayo del prototipo se ha verificado que éste carece de errores que puedan hacerlo fallar, funciona perfectamente y cumple sobradamente los objetivos inicialmente planteados.