Jagonino: level measuring cap for swimming-pool liquid Jags

DANIEL PUEYO SOLER
Grado en Ingeniería de Tecnología y servicios de Telecomunicación
Mención en Telemática

Consultor: Antoni Morell Pérez
Nombre Profesor/a responsable de la asignatura: *Pere Tuset Peiró*

Fecha de Entrega: 02/01/2022

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Jagonino: level measuring cap for swimming-pool liquid jags*
Nombre del autor:	*Daniel Pueyo Soler*
Nombre del consultor/a:	*Antoni Morell Pérez*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	01/2021
Titulación::	*Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación*
Área del Trabajo Final:	*TFG - Arduino*
Idioma del trabajo:	*Castellano*
Palabras clave	*IOT, Arduino, ESP8266*

Resumen del Trabajo:

Desde que en el año 1999 se escuchó hablar por primera vez de *Internet of Things*, el crecimiento en el número de dispositivos conectados en el mundo es exponencial, y se incrementa con el paso de los años a la par que la evolución en la velocidad y calidad de las comunicaciones móviles. En dicho contexto, cada vez es más fácil conseguir que las “cosas” sean inteligentes y estén conectadas.

En el mercado existen soluciones *IoT* para controlar y vigilar diferentes aspectos en el ámbito de las piscinas, como controladores de nivel de cloro o PH en el agua, medidores de calidad, o control de depuradoras. Sin embargo, no se ha podido encontrar un medidor del nivel de líquido en las garrafas de líquido (Cloro, PH).

Con este proyecto se pretende diseñar un sistema de control de nivel líquidos en garrafa, a partir de la sustitución del tapón por uno que contenga un dispositivo de medición, conectado con un dispositivo compatible con Arduino y que envíe los datos a una plataforma *IoT*. Así mismo, a partir de los datos obtenidos, se pretende diseñar un dispositivo de alarma que avise en caso de que el líquido vaya a terminarse en cualquiera de las garrafas conectadas.

Los dos retos fundamentales trabajados son: por un lado, el tamaño del dispositivo de medición, que debe integrarse en un tapón de garrafa y, por otro lado, su consumo eléctrico, con el objeto de poder usar baterías integrables en el tapón y con una duración aceptable.

Abstract (in English):

Since the Internet of Things was first heard in 1999, the growth in the number of connected devices in the world has been exponential, increasing over the years, with the evolution in the speed and quality of mobile communications. In this context, it becomes easier to make "things" smart and connected.

Nowadays, it is possible to find devices on the market to control and monitor different aspects of swimming pools, such as level controllers chlorine or PH in the water, water quality meters, or water purifiers controllers. However, it has not been possible to find a liquid level meter of the liquid jags (chlorine, PH).

The aim of this project is to design a system for monitoring the level of liquids in jags, by replacing the jag cap with one containing an Arduino-compatible connected measuring device, and sending the data to an IoT platform. In adition, based on the data obtained, the aim is to design an alarm device that advices when the liquid is about to run out in any of the connected jags.

The two main challenges are, on one hand, the size of the solution, which must be integrated into a jag cap and, on the other hand, the power consumption, in order to be able to use batteries that can be integrated into the cap and with an acceptable duration.

1. TABLA DE CONTENIDOS

FICHA DEL TRABAJO FINAL 3
1. Objetivos del proyecto: 6
1.1. Introducción 6
1.2. Objetivos 6
1.3. Planificación 8
2. Estado del Arte 9
2.1. Introducción: 9
2.2. Orígenes de *IoT y Arduino:* 9
2.3. Estado Actual 12
2.4. Tendencias 14
2.5. Caso particular de las piscinas 14
2.6. Conclusión 16
3. Justificación HW a utilizar 17
3.1. Modulo Arduino y de comunicaciones. 17
3.2. Estudio de consumo 19
3.3. Sensor para medición de nivel de líquido. 22
3.4. Step Up. 23
3.5. Batería y Medición de su nivel. 23
3.6. Dispositivo/Display para las alertas. 24
4. Decisión plataforma IoT y modelo de datos. 25
4.1. Plataforma IoT. 25
4.2. API/REST de ThingSpeak (22) 26
4.3. Modelo de datos 27
5. Dispositivo Medidor del nivel de líquidos. 29
5.1. Hardware: 29
5.2. Código del del dispositivo de medición de nivel de líquido. 32
6. Dispositivo de alerta: 38
6.1. Hardware del dispositivo de alerta: 38
6.2. Código del dispositivo de alerta 42
7. Diseño de cajas/encapsulados de los módulos 46
8. Pruebas 49
9. Valoración económica 53
10. Conclusión 54
11. Próximos Pasos 55
12. Bibliografía 56
13. Ilustraciones: 57

1. OBJETIVOS DEL PROYECTO:

1.1. INTRODUCCIÓN

La mayor parte de las piscinas constan de un sistema de control automático de cloro y de PH, con dispositivos (conectados a internet o no), que automatizan el mantenimiento de la piscina con los niveles óptimos de cloro y PH. Para ello, conectan las bombas de inyección de los líquidos por un lado al circuito de depuración de agua, y por el otro a las garrafas que contienen los propios líquidos. Todo ello controlado por sondas que miden el nivel en el agua de la piscina. Pero el control del líquido en las garrafas es manual. Hay que desplazarse periódicamente a la sala de depuración y verificar si hay que reemplazar alguna de las garrafas. En muchas ocasiones puede acabarse el líquido de una garrafa y pasar horas o días sin ser conscientes de ello, provocando la aparición de algas y suciedad en la piscina.

Para evitarlo, se va a diseñar un sistema de control del nivel de líquidos de las garrafas de cloro y PH, en el que un dispositivo muy compacto se acoplará al tapón de las garrafas y periódicamente medirá el nivel de líquido y de su batería, y enviará los datos a una plataforma IOT, manteniéndose “dormido” el resto del tiempo.

Así pues, con este TFG se busca diseñar un sistema autónomo de medición y envío de los datos, compacto y de bajo consumo y un dispositivo de alerta también autónomo y compacto. Todo ello autoconfigurable, y apoyándose en una plataforma IoT libre.

1.2. OBJETIVOS

Los objetivos detallados del proyecto son los siguientes:

- Desarrollo del sistema de medición de nivel de líquido a partir de sensores y un dispositivo compatible con Arduino, así como el dispositivo de recepción de las alertas.
- Selección de plataforma IoT y definición de un modelo de datos que permita la explotación de estos de forma sencilla. La plataforma debe ofrecer la posibilidad de permitir 2 nodos de datos de forma totalmente gratuita, así como un sistema sencillo de lectura y escritura de los datos y facilidad de uso

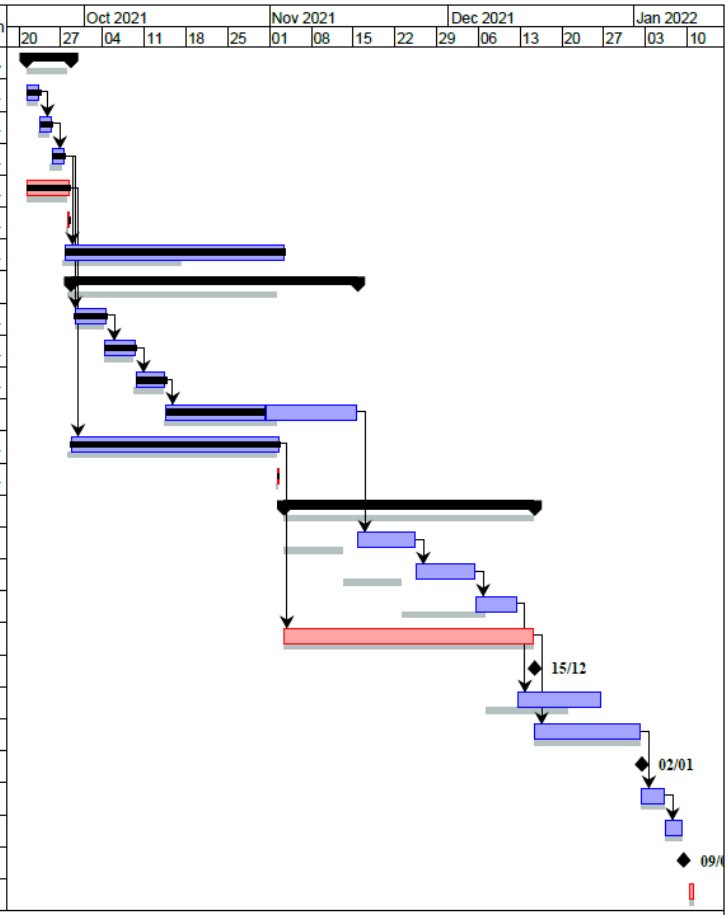
desde Arduino. Es preferible si se ofrecen librerías propias para Arduino para facilitar la explotación de los datos.

- Los dispositivos de medición se acoplarán a un tapón de garrafa de unos 6 cm de diámetro. Por ello uno de los requerimientos es utilizar dispositivos del menor tamaño posible y conseguir un ensamblaje lo más compacto posible. El dispositivo compatible con Arduino debe tener unas dimensiones máximas de 5x4 cm con el objeto de acoplarlo en en un conjunto de no más de 5,5x4,5 cm. De esta forma se podrá incorporar sin problemas en un tapón de nuevo diseño pero que no permite modificar su anchura (6cm), ya que ésta la marca la rosca de la garrafa.
- Control de consumo optimizado. Cada dispositivo estará alimentado con batería o pilas y se buscará la mayor duración posible sin necesidad de carga o sustitución. La duración de la batería deber ser de un mínimo de 4 semanas. Considerando baterías de 1800 mAh, el consumo no debe superar (24h x 28 días=672h) los 2,67 mAh en ningún caso.
- Cada dispositivo controlará su nivel de carga de batería y avisará en caso de necesidad de carga o reemplazo. Esto supone que, para poder controlar el nivel de batería, el dispositivo compatible Arduino debe de disponer de al menos de una entrada analógica.
- Los nodos estarán conectados por Wifi, y necesitan ser configurados inicialmente. Para ello se diseñará una página que será servida por el propio dispositivo arrancando en modo “punto de acceso” en el que se configurará el SSID y contraseña de la WIFI, y los datos específicos del nodo.

1.3. PLANIFICACIÓN

Se detalla a continuación el diagrama de Gantt con la planificación del presente TFG.

		Name	Duration	Start	Actual Finish
1	✔	**PEC1**	**7.375 d...**	**21/09/21...**	**28/09/2...**
2	✔	Definición de los requerimientos	2.375 d...	21/09/21 ...	23/09/21 ...
3	✔	Planificación	2 days	23/09/21 ...	25/09/21 ...
4	✔	Elección de dispositivos	2 days	25/09/21 ...	27/09/21 ...
5	✔	Escritura Introducción, planificación, m...	7.375 d...	21/09/21 ...	28/09/21 ...
6	✔	Entrega PEC1	0.375 d...	28/09/21 ...	28/09/21 ...
7	✔	Compra de los dispositivos.	37 days	27/09/21 ...	03/11/21 ...
8		**PEC2**	**48 days**	**28/09/21...**	
9	✔	Análisis de opciones IoT en la nube. Ele...	5.375 d...	29/09/21 ...	04/10/21 ...
10	✔	Estado del Arte	5.375 d...	04/10/21 ...	09/10/21 ...
11	✔	Elección y definición del Modelo de dato...	5 days	09/10/21 ...	14/10/21 ...
12		Desarrollo inicial y Pruebas de funciona...	32 days	14/10/21 ...	
13	✔	Escritura Memoria Inicial	35 days	28/09/21 ...	02/11/21 ...
14	✔	Entrega PEC2	0.333 d...	02/11/21 ...	02/11/21 ...
15		**PEC3**	**42 days**	**03/11/21...**	
16		Desarrollo de sistema WEB de configura...	10 days	15/11/21 ...	
17		Desarrollo inicial y Pruebas de funciona...	10 days	25/11/21 ...	
18		Montaje final de los dispositivos en PCB.	7 days	05/12/21 ...	
19		Desarrollo de la 2ª parte de la memoria	42 days	03/11/21 ...	
20		Entrega PEC3	0 days	15/12/21 ...	
21		Diseño de solución de ubicación y fijación	14 days	12/12/21 ...	
22		Finalización Memoria	18 days	15/12/21 ...	
23		entrega Memoria	0 days	02/01/22 ...	
24		Preparación Presentación	4 days	02/01/22 ...	
25		Depuración de código	3 days	06/01/22 ...	
26		Entrega presentación y código	0 days	09/01/22 ...	
27		Defensa	1 day	10/01/22 ...	



2. ESTADO DEL ARTE

2.1. INTRODUCCIÓN:

Vivimos en un mundo conectado. Hoy en día ya podemos asegurar que en este planeta hay más dispositivos conectados a Internet que humanos viviendo en él. Podemos asegurar que estamos viviendo la revolución de la Internet de las cosas (*IOT).* Esto es así porque se está consiguiendo aumentar la productividad en prácticamente todas las áreas introduciendo el uso de dispositivos conectados. Podríamos poner mil ejemplos en medicina, industria, agricultura, transporte, medio ambiente, etc. Y también para uso doméstico, con cada vez más dispositivos conectados a Internet (cámaras, sensores, televisiones, despertadores, relojes, neveras, lavadoras…) Pero ¿cuáles son los dispositivos que podríamos incluir en la categoría de *IoT*? Pensemos en los dispositivos que tenían acceso a Internet desde su inicio, como los ordenadores, móviles inteligentes, y cuales han incorporado su acceso a Internet en los últimos años (relojes, despertadores, televisores, neveras). Éstos últimos son los que podríamos incluir en la categoría de IoT.

Algunos de los dispositivos que podríamos considerar como parte del *IoT* son los dedicados al control desde Internet de la depuración de las piscinas. Desde hace decenas de años el sistema de depuración estaba automatizado, pero ahora es posible controlar desde dispositivos conectados a Internet sus niveles de Cloro y PH.

2.2. ORÍGENES DE IOT Y ARDUINO:

IOT son las siglas del inglés *Internet Of Things* y fue nombrado por primera vez en 1999 por Kevin Ashton, en el *Auto-ID Center* del *MIT* mientras investigaban sobre la identificación de objetos a usando identificadores de frecuencia (*RFid*). Luego el termino se extendió a cualquier dispositivo que pudiese establecer una comunicación con otro e interactuar de forma remota. (1) (2)

Pero no fue hasta 2009 cuando en un artículo firmado por Ashton para el diario RFID, cuando claramente se identificó el término “Internet of things”. En aquel momento todavía muy ligado exclusivamente a la tecnología de identificación por radiofrecuencia.

Unos pocos años antes, en el año 2005, se gestó en el Bar del Rey Arduino en Ivrea (Turín) (3) el concepto de “placa Arduino”. Se trataba de diseñar una placa electrónica microcontroladora muy económica, y que se pudiese utilizar en la enseñanza en el instituto de diseño interactivo de la localidad. Poco a poco, las placas Arduino fueron introduciendo nuevas características o módulos auxiliares de comunicación que permitieron incorporar el uso de las placas para el IOT. El éxito de “Arduino” se basa en la sencillez, el precio, y en el hecho de considerarse “HW abierto”, así como el uso de “SW abierto”. El considerar tanto el HW como el SW como “*Open Source*” ha permitido su gran expansión con millones de dispositivos originales o compatibles vendidos, así como expansiones o ampliaciones. En la siguiente imagen tomada de la página Web de Arduino (4), se puede comprobar la explicación de su licencia Creative Commons.

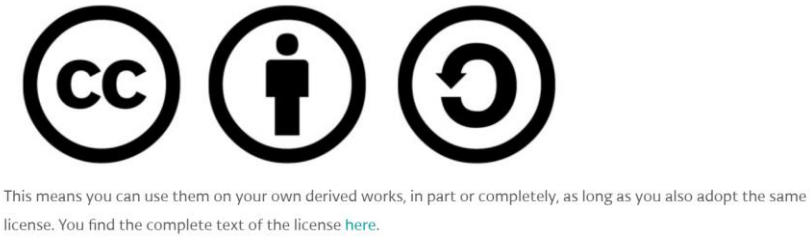


Ilustración 1. Fuente Arduino.com

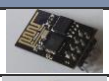
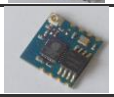
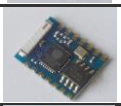
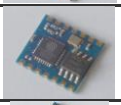
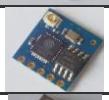
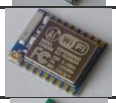
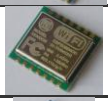
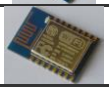
Con la aparición de las extensiones o *shields* que permitieron conectar las placas Arduino o sus clones a Internet (por cable o de forma inalámbrica), es cuando cualquier aficionado a la electrónica o a la informática pudo desarrollar sus aplicaciones IOT. No obstante, el uso de dichas extensiones para conseguir la conectividad hizo que, por un lado, los dispositivos tenían mayor tamaño y precio. Por ello en el año 2014 la compañía AI-Thinker, de la mano de la compañía de Shanghai (China) Espressif Systems introdujo en el mercado un dispositivo microcontrolador con capacidad de comunicación inalámbrica. Se trataba de la versión ESP-01 con el microchip ESP8266. Dicho módulo, permite por un lado incorporar funcionalidades inalámbricas a otros controladores como Arduino, usando comandos estándar de telefonía Hayes AT, y por otro lado, permite incluso funcionar de manera autónoma como un microcontrolador básico inalámbricamente conectado (5). Además, los módulos ESP8266 son plenamente compatibles con Arduino tras la aparición de diversas librerías, lo que facilita enormemente su uso. Hay que destacar que el precio

de dicho módulo es muy bajo, lo que ha permitido una gran popularidad de éste. Podemos ver en la siguiente imagen una foto del módulo ESP8266 ESP-01:

Ilustración 2. Fuente: https://upload.wikimedia.org/wikipedia/commons/8/84/ESP-01.jpg

Este módulo, De un tamaño muy pequeño (14x11 mm), consta de hasta 3 entradas digitales, pero no contiene ninguna entrada analógica. Tampoco permite la conexión para poder despertar del estado *deep sleep* de bajo consumo.

Las distintas evoluciones de este módulo lo hicieron aumentar ligeramente su tamaño, pero aumentando sus prestaciones en cuanto a número de entradas digitales, memoria, etc. Podemos ver en la siguiente tabla la evolución de los módulos ESP8266.

Módulo	dimensión (mm)	pins activos	LED	Antena	
ESP-01	14.3 × 24.8	6	Si	circuito	
ESP-02	14.2 × 14.2	6	No	externa	
ESP-03	17.3 × 12.1	10	No	cerámica	
ESP-04	14.7 × 12.1	10	No	no	
ESP-05	14.2 × 14.2	3	No	externa	
ESP-06	14.2 × 14.7	11	No	no	
ESP-07	20.0 × 16.0	14	Si	cerámica/ externa	
ESP-08	17.0 × 16.0	10	No	no	
ESP-09	10.0 × 10.0	10	No	no	
ESP-10	14.2 × 10.0	3	No	no	
ESP-11	17.3 × 12.1	6	No	cerámica	
ESP-12	24.0 × 16.0	14	Si	circuito	

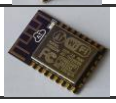
ESP-12E	24.0 × 16.0	20	Si	circuito	
ESP-12F	24.0 × 16.0	20	Si	circuito	
ESP-13	18.0 x 20.0	16	No	circuito	
ESP-14	24.3 x 16.2	22	No	circuito	

Ilustración 3: Fuente de las imágenes (6)

A partir de la aparición del microcontrolador ESP8266, y en paralelo a los módulos con ESP8266, aparecieron en el mercado placas de prototipado con ESP8266, qué, al igual que los módulos Arduino, permitían la ampliación sencilla con extensiones o *shields.* Las más utilizadas en la actualidad, y de las que existen diversas versiones son las placas Wemos, y NodeMCU (desde 2014). Concretamente la placa Wemos D1 Mini, que, con un bajo coste y tamaño pequeño, permite sen programada desde el IDE de Arduino y ser ampliada de forma muy sencilla. Además, también fueron diseñadas bajo el principio de HW abierto, por lo que al igual que las placas de Arduino han surgido multitud de “clones” o placas prácticamente idénticas. Como ejemplo, podemos ver algunas imágenes de NodeMcu y Wemos D1 Mini:

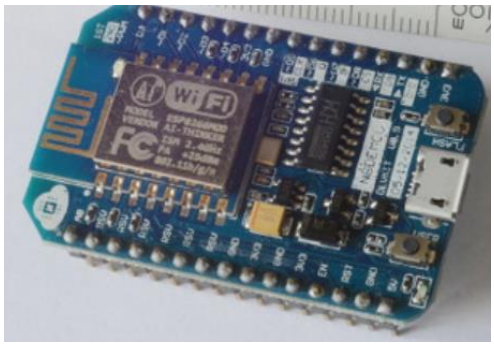


Ilustración 4. NodeMCU (6)

Ilustración 5. Wemos D1 Mini (7)

2.3. Estado Actual

En septiembre de 2016, apareció en el mercado la evolución del microcontrolador ESP8266. Se denomina ESP32 y como el anterior, lo fabrica la compañía china Espressif Systems. Consta de conectividad Bluetooth además de wifi. Hay que destacar que dicho controlador, además de algunas mejoras que

detallaremos a continuación, incorpora mejoras en el consumo junto a nuevas opciones de bajo consumo, especialmente interesantes para el desarrollo de aplicaciones IoT, como permitir entrar en modos de bajo consumo manteniendo el control de los sensores y permitiendo que éstos “despierten” al microcontrolador en determinadas condiciones.

Las principales mejoras del microcontrolador ESP32 frente al ESP8266 las podemos comprobar en la siguiente tabla:

Característica	ESP8266	ESP32
Procesador	Tensilica LX106	Tensilica Xtensa X36
Nº nucleos	Single core	Dual core
Velocidad	80Mhz (hasta 160 Mhz)	160 MHz (hasta 240 MHz)
SRAM	160 kB	512 kB
Consumo mínimo Deep Sleep	20 uA	2.5 uA
Encriptación HW	No (TLS 1.2 por SW)	Sì
Bluetooth	No	v4.2 BR/EDR + BLE
GPIO utilizables	17	36
ADC	1 (10 bits)	18 (12 bits)
DAC	No	2 (8 bits)
I2C	1	2
SPI	2	4
I2S	1	2
CAN BUS	No	1 x 2.0
Sensor Touch	No	10
Sensor temperatura	No	Sí
Sensor HALL	No	Sí

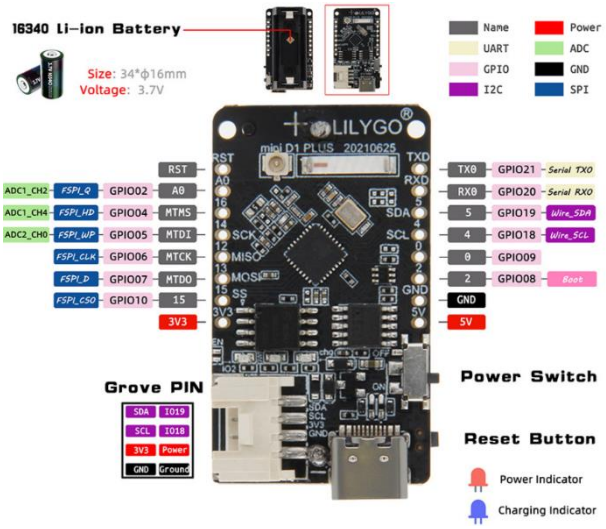


Ilustración 6. Fuente lilygo.com

El microcontrolador ESP32 también está en la actualidad en constante evolución. En concreto, en 2020 Espressif ha lanzado el módulo ESP32-C3, que está llamado a reemplazar en popularidad a los módulos ESP8266-ES0x. En él encontramos un SoC (System On Chip) mono núcleo que utiliza arquitectura Open Source RISC-V (Risk-Five) de hasta 160 Mhz. Es un cambio importante, ya que los modelos anteriores utilizaban procesadores basados en arquitectura MIPS. Ambos (RISK-V y MIPS) están basados en RISC, pero RISK-V utiliza un conjunto de instrucciones abierto y libre. A partir de dicho microcontrolador ESP32, también han evolucionado los módulos y placas compatibles como Wemos, NodeMCU, y clones, ofreciendo sus versiones para dicho microcontrolador. Un

ejemplo muy interesante es el módulo de la compañía china Lilygo, que incluye un soporte de batería del tipo 16340. Es un módulo que optimiza el espacio (45.7x25.8x15,8mm) con la máxima funcionalidad del microcontrolador ESP32-C3. (8)

2.4. TENDENCIAS

La tendencia actual es disponer cada vez más dispositivos de todo tipo conectados a Internet en todos los ámbitos y junto a la Inteligencia Artificial y el *Machine learning* conseguir mejoras de todo tipo. En el ambito laboral, la combinación del IoT con la Inteligencia artificial ayuda a las organizaciones a comprender y predecir una variedad de riesgos, así como automatizar una respuesta rápida, con ello se consigue mejorar sensiblemente la seguridad de los trabajadores, los ciberataques y posibles pérdidas financieras. Por ejemplo, Fujitsu ha conseguido con *machine learning*, analizar los datos de pulseras conectadas de sus trabajadores y así estimar el riesgo de estrés que pueden sufrir los trabajadores de las fábricas. En algunas partes del mundo se ha empezado a identificar en tiempo real actividades sospechosas, con camaras en los cajeros automaticos, en casinos, etc. (9)

En un articulo de Deloitte se destaca que los proyectos IoT pueden mejorar la eficiencia en las empresas. En el proceso de fabricación, Deloitte considera que las predicciones obtenidas a partir de estas tecnologías podrán reducir los tiempos requeridos en determinados procesos en un 20-50%, y aumentar el tiempo de actividad efectiva así como la disponibilidad del equipo en un 10-20%. También se podrán reducir los costes generales de mantenimiento en un 5-10%. Asimismo, en el informe también se comenta que la predicción basada en IA, sobre todo en el *machine learning*, ayudará a Google a reducir el 40% de los costes de refrigeración de los centros de datos. (10)

2.5. CASO PARTICULAR DE LAS PISCINAS

Podemos leer en higieneambiental.com que según la Asociación Española de Profesionales del Sector de la Piscina (ASOFAP), la demanda de productos de automatización ha crecido un 10% desde el 2017 hasta un 24% del total. Esta cifra sigue en crecimiento y tiene todavía un importante margen de crecimiento (11). Cada vez son más los fabricantes de productos para la automatización remota de piscinas.

Podemos encontrar algunos ejemplos como los productos de Fluidra, que con su Internet of Pools ha desarrollado productos para la gestión automática y remota de muchos de los elementos de la piscina. Su serie de productos FluidraConnect junto a iAquaLink permiten controlar los elementos de depuración, así como los niveles de cloro y ph, limpiafondos, y luces. Su producto Bluriiot permite controlar los parametros del agua como niveles de cloro, ph y temperatura etc. con un dispositivo que se introduce dentro de la misma piscina. (12)

Son precisamente esos dispositivos, que se introducen en la piscina con un flotador y que contienen diversos sensores para medir los parámetros de agua, los que están apareciendo en los últimos años como estandarte del IoT en el mundo de las piscinas. Podemos encontrar otros ejemplos como el de Sigfox con una función muy similar con un dispositivo flotante (13).

Para poder medir los distintos parámetros de una piscina se necesitan distintos tipos de sensores. En detalle los elementos a controlar serían los siguientes, con las características de los sensores:

- Temperatura del agua: Para ello se necesita un sensor de temperatura sumergible. Por ejemplo el DS18B20, que ademas de ser sumergible incorpora una pequeña memoria para almacenar algunos valores como los valores máximos y los mínimos. (14)
- Nivel de cloro en el agua: El cloro garantiza la desinfección del agua de la piscina si se encuentra en proporciones adecuadas, por ello es imprescindible mantener la piscina con niveles de cloro adecuados tanto para el baño, como para la desinfección. Para la medición del cloro libre el ácido hipocloroso , las cloraminas juegan un papel en un complejo sistema de reacciones. Este proceso funciona en un amplio rango de valores de pH y temperatura, pero hay que destacar que si el PH se encuentra en valores elevados (por encima de 8) las mediciones de cloro no serán correctas. Para la medición es fundamental la reacción del ácido hipocloroso. que se difunde a través de la membrana y produce una reacción. Estas reacciones químicas que se producen en una piscina en presencia de cloro, determinan el potencial de reducción de oxidación (ORP- *Oxydo Reduction Potential*). Que al final es lo que nos interesa para poder medir el nivel de cloro en la piscina. El tiempo que necesita el agua de la piscina para exterminar algas o bacterias depende del valor ORP. Cada

elemento químico posee electrones y, según las condiciones de reacción, podrá, bien cederlos o bien aceptarlos, formado así un par redox. ORP es el potencial generado en dicho intercambio de electrones y se mide en mV. El valor ideal es aproximadamente 700 mV. (15)

- Nivel de PH: El pH mide la acidez o alcalinidad de un líquido, la escala de pH varía de 0 a 14. El pH nos indica la concentración de iones hidrógeno [H]+ presentes. Se puede cuantificar de forma precisa mediante un sensor que mide la diferencia de potencial entre dos electrodos: un electrodo de referencia (de plata/cloruro de plata) y un electrodo de vidrio que es sensible al ion de hidrógeno.
- Control de parada y arranque de la depuradora:
  El control de parada y arranque se efectua simplemente con Relés.
- Para medir el nivel de agua en una garrafa de cualquier producto (cloro, ph), se puede utilizar un sensor de ultra sonidos como el HC-SR04. Se envía un pulso con ultrasonidos y se mide el tiempo que tarda en recibirse el echo del pulso. Sabiendo la velocidad del sonido es fácil calcular la distancia hasta el nivel de liquido en la garrafa.

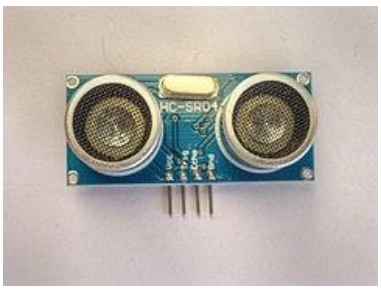


Ilustración 7. Fuente: https://commons.wikimedia.org/wiki/File:HC_SR04_Ultrasonic_sensor_1480322_3_4_HDR_Enhancer.jpg

2.6. CONCLUSIÓN

IOT está en continuo avance desde que en 1999 se empezase a hablar de el. En la actualidad son cientos las utilidades prácticas en todos los ambitos, domesticos, empresariales, sanitarios, etc. Muchas de éstas utilidades, de la mano de Arduino, ESP8266 y otros, están cambiando nuestra forma de interactuar con el entorno. Además, al unirlo con los avances en técnologias con inteligencia Artficial, el potencial se incrementa muchísimo. En los próximos años, y con la aparacición de formas de comunicación inalahambricas más rápidas, el crecimiento será aún mayor y nos afectará en muchos aspectos de nuestra vida cotidiana.

3. JUSTIFICACIÓN HW A UTILIZAR

3.1. MODULO ARDUINO Y DE COMUNICACIONES.

Poniendo sobre la mesa los requerimientos en cuanto a nivel de tamaño y dado que en realidad no son necesarias muchas entradas digitales ni analógicas, en seguida se descartaron lo módulos Arduino más comunes como el Arduino UNO. Aunque junto a un módulo ESP8266 podría hacer la función, su tamaño es superior a los máximos requeridos. Por otro lado, al examinar los módulos ESP8266 como función de módulo de comunicaciones, enseguida se puede observar que ellos mismos son compatibles con Arduino y que tienen la función de controlador con entradas y salidas digitales y analógicas que permiten el control de dispositivos y sensores.

Analizando en detalle diversos dispositivos comerciales ES8266 o ESP32, y con la premisa de un tamaño máximo de lado de 5,5 cm, se encuentran algunas alternativas:

- El módulo ESP8266 más conocido y de menor tamaño es el ESP8266 ESP01. Este dispositivo no es viable por 2 motivos, por un lado, no dispone de pin conector para activar la salida de RESET desde el modo Deep Sleep, necesario para conseguir poner el módulo en modo de bajo consumo, por otro, solo dispone de 2 pines de entrada digital, y ninguno analógico, requerido para poder medir el nivel de la batería.
- Leyendo información con comparativas de consumo de módulos ESP8266, es posible identificar que el módulo Wemos D1 Mini Lolin si cumpe con los requisitos. Una medida de 35x26mm es óptima para lo que se necesita, además consta de 1 entradas analógica y varias digitales. Con respecto al consumo, en modo Deep Sleep consumiría una media de 78 µA. (16)
- Existen clones del módulo anterior, que modifican ligeramente el HW y no consiguen consumos tan optimizados. Pasan a consumir en modo Deep sleep entre 100 y 200 µA, que si que cumplirían con los requerimientos, aunque de forma más ajustada. (16)
- Otros módulos muy conocidos como NodeMCU, tienen consumos superiores en modo Deep Sleep, del orden de mA, por lo que no serían adecuados para los requerimientos de consumo. Además, son de un

tamaño ligeramente superior, aunque el tamaño no sería un obstáculo, al medir unos 5cm de longitud algunos de los módulos, como el NodeMCU Amica.

- Si nos vamos la evolución del ESP8266, existen diversos módulos con microcontrolador ESP32. Analizando sus especificaciones, el ESP32 ofrece mejores alternativas de bajo consumo y concretamente, analizando su *datasheet*, ofrece un consumo muy inferior, de unos 10 µA en modo *Deep sleep* o incluso 5 µA en modo hibernación. (17)

Light-sleep	-	0.8 mA
Deep-sleep	The ULP coprocessor is powered on.	150 μA
	ULP sensor-monitored pattern	100 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	5 μA
Power off	CHIP_PU is set to low level, the chip is powered off.	1 μA

Ilustración 8.Fuente: Espressif ESP32 Datasheet

Por ello, algún módulo con tamaño adecuado sería óptimo para maximizar la duración de la batería.

Concretamente, existe la evolución al módulo Wemos D1 mini Lolin, Qué sería el D1 mini ESP32, con tamaño (39×31mm) ligeramente superior al ESP8266 D1 mini.

- Existe un módulo aparecido recientemente, que incluye la última revisión de ESP32 (C3), junto a un soporte de batería. Por tanto, se trata de un módulo que optimiza el espacio. Se trata del módulo LILYGO® TTGO T-OI PLUS, que con unas medidas de 46x16x26 y parece cumplir con todas las especificaciones requeridas. (18)

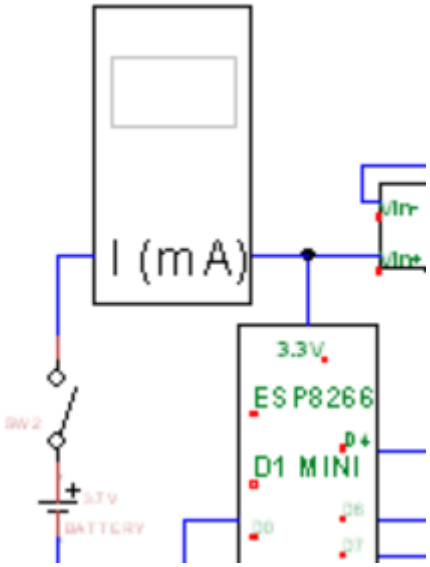
Tras analizar las opciones disponibles, se efectúa la compra del módulo LILYGO® TTGO T-OI PLUS con ESP32 solo disponible en tiendas con envío desde China con plazo de entrega inferior al 18 de Octubre (+ baterías), y un módulo clon del ESP8266 D1 MINI LOLIN con envío desde España con entrega rápida. No se localiza

Ilustración 9. Fuente Amazon.es

un módulo Wemos D1 Mini Lolin original con un plazo de entrega razonable.

3.2. Estudio de consumo

Para poder elegir de forma objetiva entre los módulos disponibles se realiza un estudio de consumo entre ellos, especialmente en el modo Deep Sleep que es en el que se encontrarán los módulos la mayor parte del tiempo. Para realizar las medidas de consumo, insertamos un multímetro entre el polo positivo de la batería y la alimentación del circuito, como muestra la siguiente imagen:



3.2.1. En primer lugar, se realizaron pruebas de consumo con el módulo ESP8266 D1-MINI. Para ello, se ha medido el consumo del módulo en diversas situaciones.

- En modo normal el módulo realiza las siguientes operaciones:
    - Conexión Wifi
    - Medidas con el sensor de distancia
    - Envío a la plataforma IoT.

Se observa un consumo esperado de unos 75mA, que de media será un poco más por los

picos de la conexión Wifi que se producen durante unas décimas de segundo.

- En modo Deep Sleep, se observa que el consumo es mucho más alto que el esperado, siendo de casi 2mA (se esperan unos 100µA). Con ese consumo, una batería de 1800 mAh se agotará en pocos días. Tras analizar las causas, se observa que el sensor de ultrasonidos, si se mantiene alimentado en todo momento. Por ello se incorpora un mosfet-N controlado desde uno de las salidas del microcontrolador. De esta forma, antes de realizar las mediciones se activa el Mosfet-N para alimentar el sensor. Concretamente, usaremos un 2N7000, que como podemos ver en sus características, soporta una corriente continua de 200mA que es más que suficiente para nuestro sensor de ultrasonidos. Así mismo es un dispositivo que tiene unas pérdidas del orden de nA.

Ilustración 10. Consumo ESP8266 Deep Sleep con el Sesnsor activo

Absolute Maximum Ratings T_A = 25°C unless otherwise noted

Symbol	Parameter	2N7000	2N7002	NDS7002A	Units
V_{DSS}	Drain-Source Voltage	60			V
V_{DGR}	Drain-Gate Voltage (R_{GS} ≤ 1 MΩ)	60			V
V_{GSS}	Gate-Source Voltage - Continuous	±20			V
	- Non Repetitive (tp < 50µs)	±40			
I_D	Maximum Drain Current - Continuous	200	115	280	mA

I_{GSSF}	Gate - Body Leakage, Forward	V_{GS} = 15 V, V_{DS} = 0 V	2N7000			10	nA
		V_{GS} = 20 V, V_{DS} = 0 V	2N7002 NDS7002A			100	nA
I_{GSSR}	Gate - Body Leakage, Reverse	V_{GS} = -15 V, V_{DS} = 0 V	2N7000			-10	nA
		V_{GS} = -20 V, V_{DS} = 0 V	2N7002 NDS7002A			-100	nA

Podemos ver la hoja completa de características en los anexos.

- Tras incorporar el Mosfet-N, el consumo baja a unos 375µA. Si bien es un consumo superior al esperado, manteniendo el microcontrolador en modo “deep sleep” la mayor parte del tiempo, la batería aguantará varias semanas sin necesidad de ser cargada, como marcan las especificaciones. Concretamente, haciendo dos mediciones por hora (una cada 30 minutos), tendremos un consumo total de unos 375 µAh en modo *Deep Sleep* a lo que habría que sumar el consumo en modo *despierto* que sería de 80mAh·40segundos/3600 que nos da un consumo de unos 900 µAh. En total tenemos un consumo de 900+375=1,275mAh. Si utilizamos baterías de alrededor de 2000mAh, teóricamente la batería duraría unas 1570 horas (unos 60 días), suficiente para nuestro objetivo de que las baterías duren más de 4 semanas sin necesidad de ser reemplazadas.

Ilustración 11. Consumo ESP8266 DeepSleep y el sensor inactivo

3.2.2. De igual manera se efectúan las mediciones del módulo ESP32 D1-mini obteniendo los siguientes resultados:

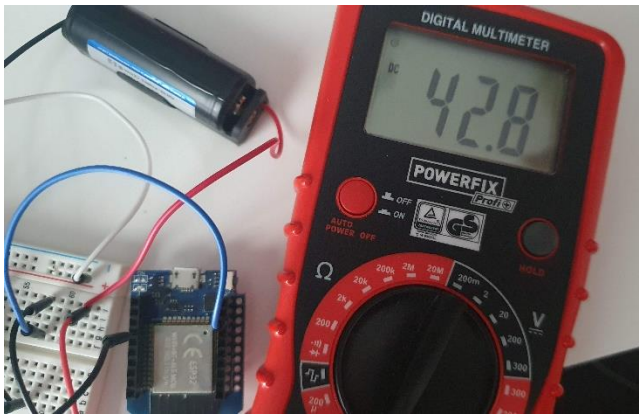


Ilustración 12. Consumo ESP32 activo

Se puede observar que el consumo en estado normal es de unos 40mA, un consumo razonable para el estado activo del módulo.

Para las pruebas de consumo del módulo en modo Deep Sleep, se realizan algunas mejoras en el código siguiendo las recomendaciones de *Espressif.* Concretamente se añadieron las siguientes líneas al código para asegurar el mínimo consumo interno del ESP32 (19).

```
esp_bluedroid_disable();
esp_bt_controller_disable();
esp_wifi_stop();
esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH,ESP_PD_OPTION_OFF);
```

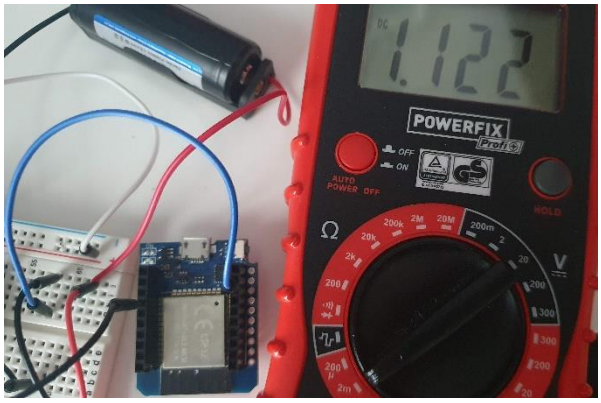


Ilustración 13. consumo en modo deep sleep del módulo ESP32

A pesar de ello, el consumo se mantuvo superior a 2mA en todo momento, resultado bastante desalentador. Leyendo información en foros de internet, se siguió la recomendación adicional de "romper" el led interno de alimentación del módulo con lo que se consiguió reducir el consumo en alrededor de 1mA. Pero con consumos en modo Deep Sleep superiores a 1mA, la batería no aguantaría más que unos cuantos días, por lo que se finalmente se descartó el uso de éste módulo.

3.3. SENSOR PARA MEDICIÓN DE NIVEL DE LÍQUIDO.

Para la medición del nivel líquido se opta por el sensor de ultrasonidos HC-SR04, que es el módulo más empleado, económico, y tiene como características (19):

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Ilustración 14. Fuente www.datasheet.es

Las medidas son apropiadas, concretamente mide 45x20x15mm, el ángulo es adecuado, ya que con 15º no se alcanza el lateral de la garrafa antes de llegar al fondo y el pulso de medición se puede controlar desde el aplicativo. Pero hay que destacar 2 puntos:

- Requiere de una alimentación de 5V, lo que obliga a incorporar un *stepUp* para subir la tensión de la batería de 3.7 a 5V.
- Tiene un consumo de 15mA, que no es asumible, por lo que se tiene que controlar su alimentación para que el consumo solo se produzca en los momentos de medición.

3.4. STEP UP.

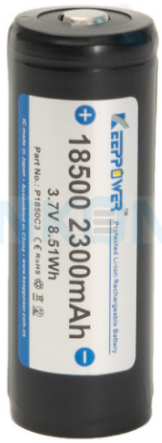
Teniendo en cuenta los 2 puntos anteriores, se elige un *step Up* ajustable que permite ajustar el voltaje de salida. Concretamente el MT3608 que ofrece salida de 5 a 28V a partir de los 2V de entrada. También se incorporará un MOSFET-N para controlar la alimentación del sensor de ultrasonidos y evitar así su consumo mientras no efectúa mediciones. Se elige el 2N7000 que permite cargas de hasta 200mA y tiene un tamaño pequeño. (20)

3.5. BATERÍA Y MEDICIÓN DE SU NIVEL.

Para conseguir que, siguiendo las especificaciones, no se deba realizar un cambio de batería en un mínimo de 4 semanas, se opta por baterías más de 1000mAh. Éstas deben tener un voltaje inferior a los 4V para poder alimentar los ESP8266 o ESP32. Las baterías de Li-Ion con una tensión nominal de 3.7V cumplirán este cometido. Aunque existen baterías tipo AAA con ese voltaje, su tamaño superior a 5,5cm (concretamente miden 6,5cm) y no cumple con las especificaciones, por lo que se opta por baterías con tamaños inferiores a 5,5cm.

En concreto, para alimentar el Lilygo T-OI Plus, se requieren baterías 16340 y para alimentar los módulos D1-Mini, se opta por baterías 18500 de 2300mAh de 3,7 V.

Pilas Recargables » / Li-Ion » / 18350 & 18500 » / Keeppower 18500 2300mAh (Protected) - 4A



Keeppower 18500 2300mAh (protected) - 4A

Sea el primero en dejar una reseña para este artículo

Información adicional

EAN / GTIN	7417940526102
Marca	Keeppower
Tamaño de la batería	18500
Química de la batería	Li-ion
Batería	Recargable
Voltaje	3.6V

Ilustración 15. Fuente Nkon.nl/es

Para medir el nivel de tensión restante en las baterías, es suficiente con configurar con Arduino el modo del pin analógico:

ADC_MODE(ADC_VCC)

que aprovecha la Resistencia Interna para poder hacer la medida con el comando:

*volt = ESP.getVcc()/1024.0*4.2/3.3*

Se usa la función getVcc para obtener el valor de tensión en la entrada análogica y se divide el valor máximo de medida (2^{10}=1024) por el valor de tensión 3.3 que es valor máximo esperado de alimentación y lo multiplicamos por el valor máximo medido de la batería (4.2V).

3.6. DISPOSITIVO/DISPLAY PARA LAS ALERTAS.

El dispositivo que recibirá el estado de los dispositivos encargados de medir el nivel de las garrafas dispondrá de un *display* para mostrar el estado de dichos dispositivos asociados. El *display* solo ha de mostrar texto, por lo que no es necesario colocar un *display* LED, y con un LCD (*Liquid Cristal Display*) de 2 líneas que muestre texto, es suficiente. Para este módulo tampoco tenemos restricción con el tamaño, por lo que podemos emplear el LCD 1602a. de la imagen.

Ilustración 16. Fuente: commons.wikipedia.com

4. DECISIÓN PLATAFORMA IOT Y MODELO DE DATOS.

4.1. PLATAFORMA IOT.

Analizando el mercado actual de plataformas de almacenamiento de datos de Sensores IOT, nos encontramos con una gran cantidad de plataformas que ofrecen ese servicio. Se busca una plataforma que permita disponer de 3 dispositivos o sensores de forma gratuita enviando o recibiendo información de forma periódica (cada varios minutos). Vamos a ver algunas características en la siguiente tabla:

Plataforma	Dispositivos gratuitos	Uso con Arduino	Precio	URL
Temboo	Periodo de pruebas	API /Librería	No disponible	https://blog.temboo.com/
myDevices	Periodo de pruebas	API /Librería	No disponible	https://mydevices.com/
TheThings	Periodo de pruebas	API	25€/año 10 dispositivos	https://thethings.io/
Kaaiot	5	MQTT /Librería. La lectura no es sencilla	15€/mes 15 dispositvos	https://kaaiot.com/
ThinkSpeak	3	API /Librería	95€/año 10 dispositivos	https://thingspeak.com/
DeviceHive	-	API	No disponible	https://www.devicehive.com/
Thingsboard	Periodo de pruebas	MQTT, HTTP	10€/mes 30 dispositivos	https://thingsboard.io/
Thinger.io	2	API	25€/mes. Sin limite de dispositivos	https://thinger.io/

Podemos ver que la mayoría de las plataformas tienen opción de pruebas, pero de un tiempo limitado. Entre las que si ofrecen un número de dispositivos sin límite de tiempo de pruebas, Kaaiot no tiene una manera sencilla para lectura de los datos. Esto lo hace con alertas, algo que no nos interesa. La plataforma Thinger.io, solo ofrece 2 dispositivos de forma gratuita, por lo que tampoco nos interesa, ya que necesitaremos 3 como veremos a continuación. Por último, Thingspeak, si ofrece 3 dispositivos (canales), tiene librerías para programar de forma simple Arduino, y además permite tanto lectura como escritura de forma sencilla.

Por todo lo expuesto, se elige Thingspeak. En caso de querer ampliarse en un futuro, Thingspeak permite escalar a un precio de los más económicos, que no llega a 1€ por dispositivo al mes.

4.2. API/REST DE THINGSPEAK (22)

ThingSpeak pone a disposición de sus usuarios una Rest API que permite el acceso a los datos de forma sencilla y directamente desde código html o JSON. Por ejemplo, cómo podemos ver la página Web de referencia de su, permite:

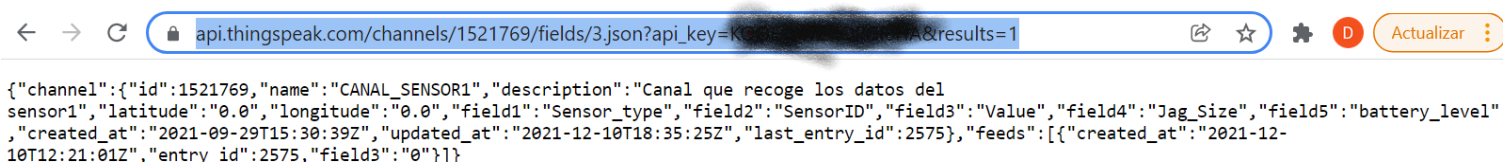
- Obtener con un http GET los datos de un registro o de un único campo. Además, se pueden obtener el último o varios de los registros seleccionados por número o fecha, e incluso obtenerlos con alguna formula incluida en la petición (máximo, mínimo, media…).

  El formato de una petición es con la URL:

  https://api.thingspeak.com/channels/<channel_id>/fields/<field_id>.<format>

  El resultado se puede obtener en formato JSON, XML o incluso CSV.

  El resultado de una petición para obtener en formato JSON el último dato del campo3 del canal_sensor1 sería el siguiente:



- Actualizar un campo o varios con un http POST o GET. La petición se puede hacer con formato html, json o csv y con la siguiente URL:

  https://api.thingspeak.com/update.<format>

- Ver los canales y sus definiciones.
- Crear y borrar canales.
- Ver el tiempo transcurrido desde la última actualización de un canal.
- Ver gráficas.

Y otras operaciones para interactuar con alertas de dispositivos y redes sociales.

Pero para trabajar con Arduino, Thingspeak ha generado una librería (Thingspeak.h) que facilita mucho el trabajo, ya que integra funciones para trabajar directamente con los datos sin necesidad de generar o leer los formatos html o json. Las funciones que utilizaremos de dicha librería son las siguientes:

- **ThingSpeak.begin**(client): Inicializa la librería para el cliente WIFI especificado.

- **ThingSpeak.readStringField**(canal,campo,APIkey) para leer el último valor de un campo en formato texto.
- **ThingSpeak.readIntField**(canal,campo,APIkey) para leer el último valor de un campo en formato Integer.
- **ThingSpeak.readSFloatField**(canal,campo,APIkey) para leer el último valor de un campo en formato Float.
- **ThingSpeak.getLastReadStatus()**, para verificar si la última lectura fue correcta.
- **ThingSpeak.readRaw()**, para enviar una petición para obtener los segundos desde la última medición.

4.3. MODELO DE DATOS

Thingspeak dedica un “canal” a cada uno de los dispositivos, por lo que se dispondrán de dos canales idénticos para los datos de Cloro y PH (u otro líquido si así fuese necesario).

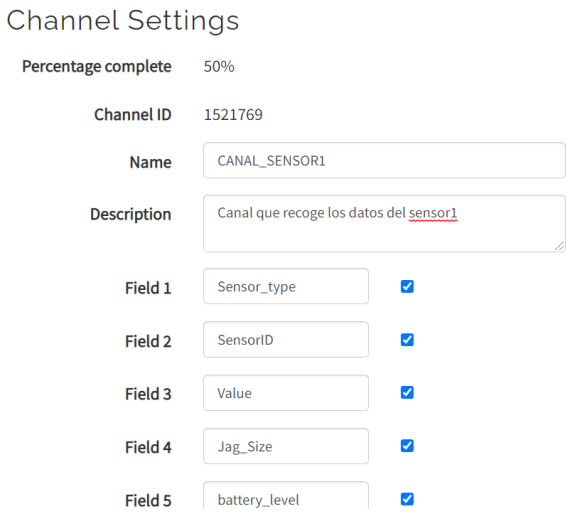
Los canales de Thingspeak, en su versión gratuita, permiten para cada uno de ellos almacenar hasta 8 campos de datos. Cada canal está identificado con un número de canal, adicionalmente para cada canal privado, dispondremos de una clave API de escritura, y otra de lectura. Con dichas claves podremos acceder a almacenar los datos leídos por el sensor. Para ello, Thingspeak proporciona una librería de libre uso (Thingspeak.h) para acceder a su API/REST de forma sencilla e integrada con Arduino, siendo esta la URL de referencia:

https://api.thingspeak.com/channels/<*channel_id*>/feeds.<*format*>

Para la medición de los datos de los sensores, se crean 2 canales iguales, uno para cada sensor, y para cada canal se definen los siguientes registros:

- Sensor_type. Tipo de garrafa de la que se mide el nivel de líquido. Es solo informativo, con el objeto de que el dispositivo receptor de alarmas pueda mostrar la información de forma más clara.
- Sensor_id. Cada placa ESP8266 contiene un número de identificación único. Durante el proceso de registro, se asociará un canal a un Sensor_id, y se añadirá un registro del nuevo sensor junto a los anteriores en el canal de configuración.

- Value. Valor de lectura del sensor, en centímetros.
- Battery_level. Nivel de tensión restante en la batería.
- Jag_size. Es el tamaño de la garrafa, para que el receptor pueda calcular, en función de la distancia al líquido, si éste se está acabando o no.



Una vez creados los canales, con sus números de canal y claves API de lectura y escritura, ya podremos hacer las llamadas para leer o escribir los datos de los sensores usando la librería propia de ThingSpeak para Arduino.

5. DISPOSITIVO MEDIDOR DEL NIVEL DE LÍQUIDOS.

5.1. HARDWARE:

Este es el esquema del dispositivo medidor de nivel de líquido, donde se puede apreciar el módulo ESP8266 D1-MINI, el Step Up MT-3608 para garantizar los 5V de alimentación del sensor, el Sensor de Ultrasonidos HC-SR04 para medir el nivel de líquido en las garrafas, el NMOS-FET 2N7000 junto a la resistencia R1 de 10K para activar el sensor solo cuando tenga que efectuar lecturas y por último, el pulsador SW1 junto al condensador, para poder controlar el tiempo que ha sido pulsado y poder así activar el modo de configuración.

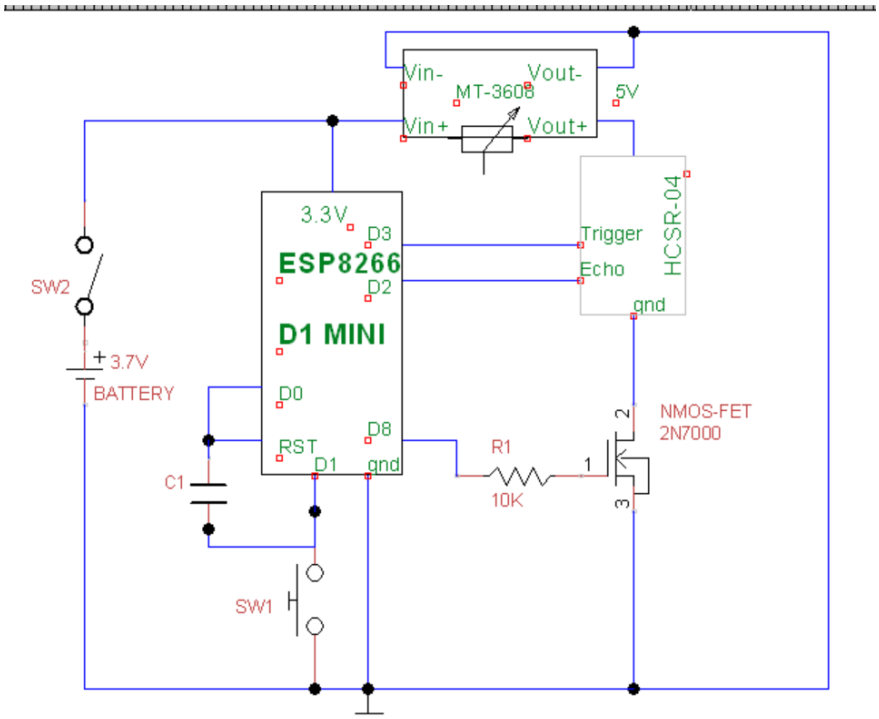


Ilustración 17. Esquema sensor

Si vemos el Pinout del módulo D1-mini, podemos observar algunos detalles sobre los pines elegidos para cada función. Concretamente, podemos observar que hay 3 Pines que no deben usarse en

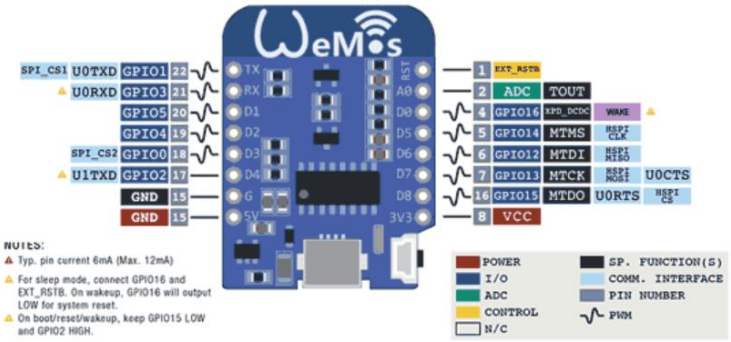


Ilustración 18. Pinout Wemos D1-Mini

el arranque del dispositivo, pues se utilizan para determinar el modo de arranque. Son GPIO03-RX, GPIO02-D4 y GPIO16-D0.

- Dichos pines no deben usarse para la lectura del pulsador durante el arranque. Para dicha función se elige el Pin D1.
- D0, lo conectaremos al pin RST para conseguir despertar el dispositivo tras el tiempo establecido.
- Para activar o desactivar la alimentación del sensor de ultrasonidos, aparentemente podríamos usar cualquier PIN de D0 a D8, excepto los 3 mencionados anteriormente que podrían quedar con alimentación en el modo *Deep Sleep*, pero tras analizar el consumo del sensor controlado por los diferentes Pines, se puede comprobar, que realmente el único que consigue mantenerse en 0V y 0mA de consumo es D8, por lo que se configura dicho PIN para dicha función.

  Para hacer funcionar el sensor, justo antes de realizar las mediciones, activamos el PIN indicado (D8), que a través de la resistencia R1 de 10K activa la circulación puerta-surtidor en el Mosfet, y así se consigue que circule corriente Drenador-Surtidor y por tanto se activa el sensor de ultrasonidos para hacer las mediciones. Tras las mediciones se vuelve a poner el nivel del PIN D8 a 0 para que no haya consumo del sensor, que recordemos, es de unos 2mA.
- Por último, D2 y D3 se utilizan para hacer las mediciones del sensor: Una vez activada la alimentación del sensor, se utiliza D2 (trigger) para enviar el tren de pulsos al sensor, y D3 para medir el tiempo de respuesta (eco) y así poder calcular la distancia a la que se encuentra el líquido.

También se han incluido en el diseño un pulsador para reiniciar, o bien para activar el modo de configuración, si se pulsa durante más de 2 segundos.

Se puede ver plasmado en la placa de pruebas

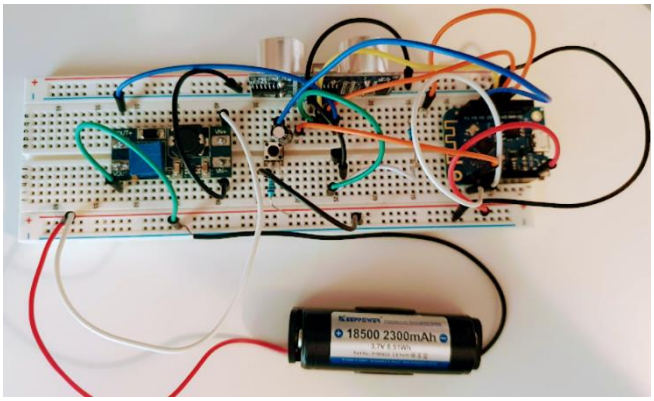
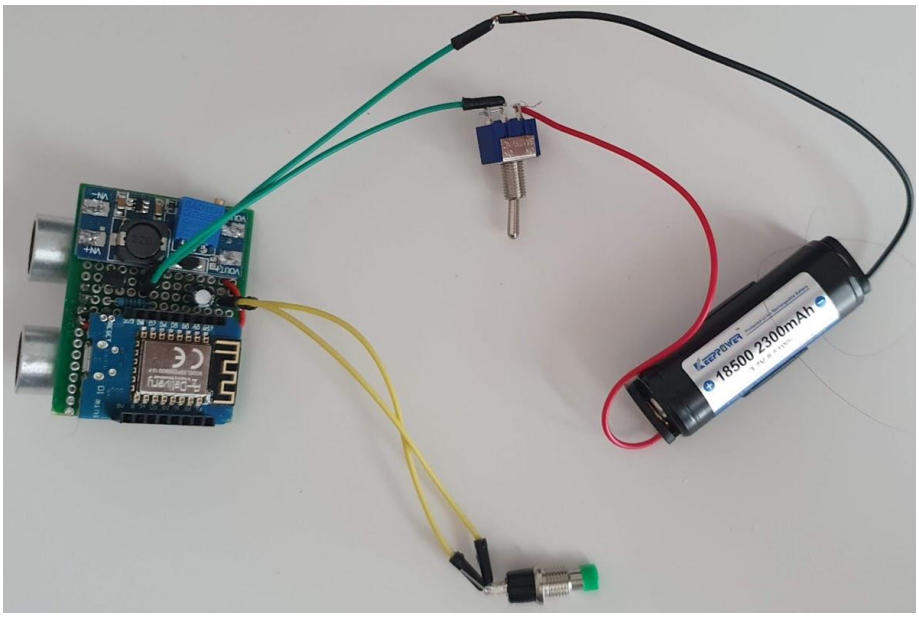


Ilustración 19. Placa de pruebas del sensor

Y una vez programado y probado, es trasladado a un PCB de soldadura para poder montarlo en el tapón de la garrafa.



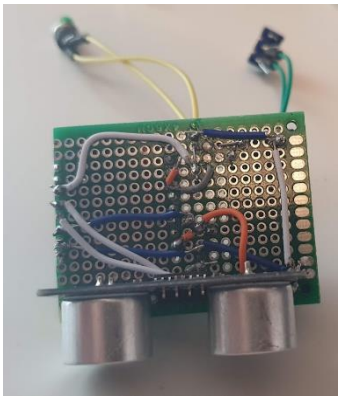
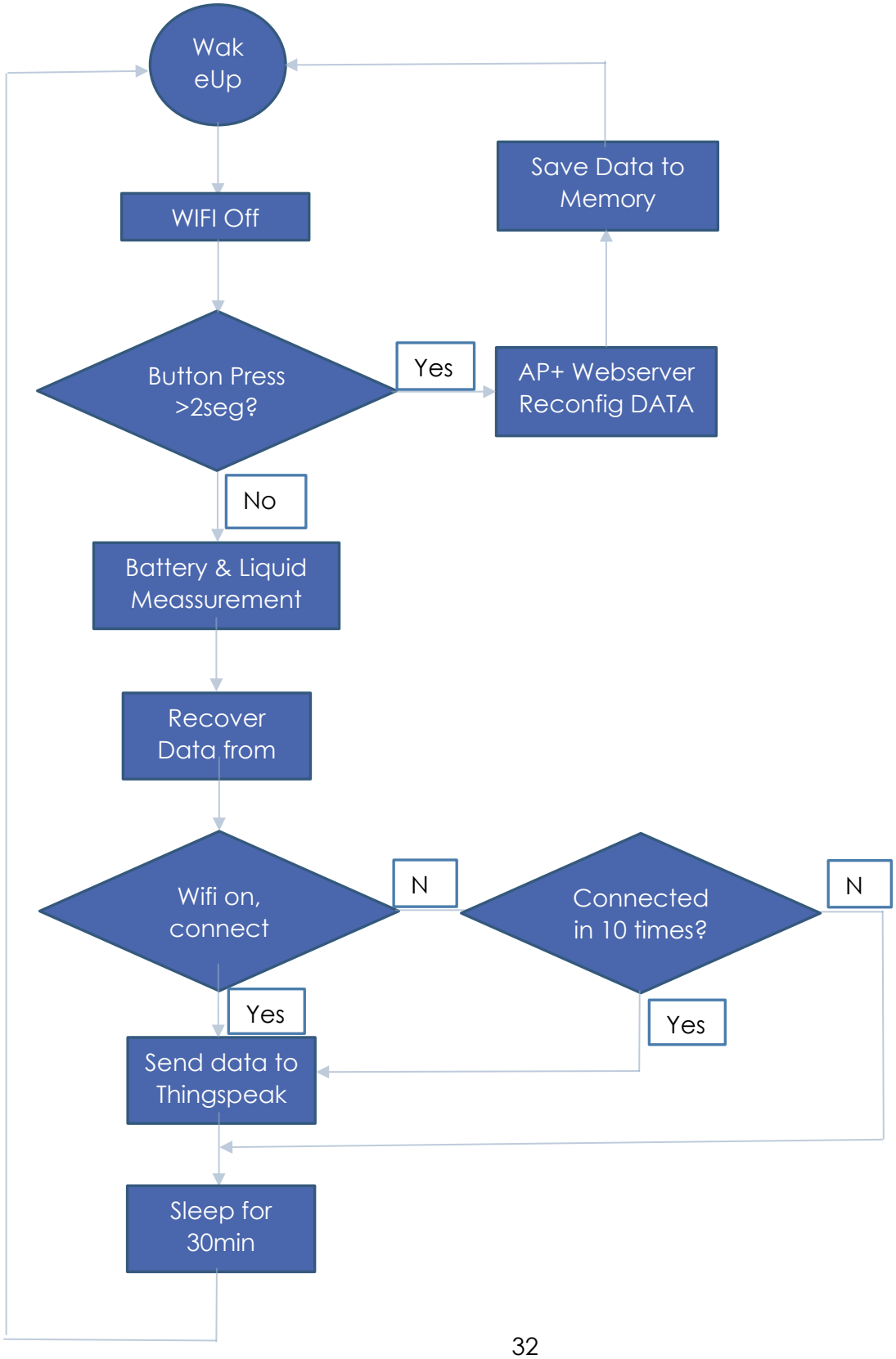


Ilustración 20. PCB de soldadura sensor

5.2. CÓDIGO DEL DEL DISPOSITIVO DE MEDICIÓN DE NIVEL DE LÍQUIDO.

En los anexos se ha incluido el código completo utilizado para los módulos con los sensores. El diagrama de flujo de funcionamiento es el siguiente:



Analizando el diagrama en detalle, podemos realizar algunas observaciones:

- Arranque: El módulo, al despertar del modo *Deep Sleep (*o al encenderse por otro motivo, como cambio de batería, o activación por interruptor), lo primero que hace es parar la Wifi mientras no se utiliza con la intención de minimizar el consumo, para a continuación verificar si ha pulsado el botón de *Reset* durante al menos 2 segundos y entrar en ese caso a la configuración del módulo.

  Existía la opción de pasar a modo configuración de forma automática si no se conseguía establecer conexión Wifi tras un número intentos, pero teniendo en cuenta que puede haber diversos motivos (como una caída temporal de la red Wifi) que impidan el acceso temporal a red wifi, se opta por entrar en modo configuración siempre a voluntad del usuario de forma controlada.

- Envío de datos a ThingSpeak:

  Para el envío de datos a la plataforma Thingspeak, se utiliza la librería propia de Thingspeak para Arduino. Una vez conectado el módulo a la red Wifi, y tras una espera de 7 segundos para asegurar que la comunicación será satisfactoria, hacemos la preparación de los datos a enviar con la siguiente función para cada uno de los campos:

  ThingsPeak.setField(campo, valor)

  A continuación, se utiliza la función ThingSpeak.writeFields(canal, APIKey) para el envío de los datos. Pero surge un problema a la hora de enviar los datos a la plataforma ThingSpeak con dicha función, que obliga a introducir un retardo de 7 segundos entre la activación de la Wifi y en el envío. El error que aparece (301) apunta a algún problema de redirección en la librería de Thingpeak, para el que, por el momento, no se ha podido encontrar solución.

```
Thingspeak API key K5OW4BXOF6V1YY3B
Starting WIFI
Succesfully Connected!!!
Problem updating channel. HTTP error code -301
Sleep for 60 seconds
```

Ilustración 21. Serial monitor, error actualizando ThingSpeak

Aquí vemos como hemos tenido que introducir un delay de 7 segundos en el código para que se realice correctamente el envío de datos a ThingSpeak.

```
ThingSpeak.begin(client);

delay(7000);
```

- Dada la importancia del consumo, se ha analizado como optimizar el código de conexión Wifi para obtener algunas mejoras y se han introducido en el código: (21).

En primer lugar, se apaga la Wifi nada más arrancar, para minimizar el consumo mientras se hacen las mediciones.

```
WiFi.mode( WIFI_OFF );

WiFi.forceSleepBegin();
```

Y la arrancamos cuando hemos de usarla:

```
WiFi.forceSleepWake();
```

Y, en segundo lugar, dado que en el proceso de arranque siempre obtenemos los datos de configuración Wifi desde la memoria, no es necesario tener los datos persistentes, por ser redundantes e incrementando el tiempo de acceso, y por tanto el consumo.

```
WiFi.persistent(false);
```

Por último, al poner el ESP en modo *Deep sleep* se le indica que arranque sin calibración de la radiofrecuencia (RF) (24):

```
ESP.deepSleep(sleepSeconds * 1000000, WAKE_RF_DISABLED);
```

Con estas medidas, se consigue reducir el consumo unos 20mA en cada proceso de lectura, y además se minimiza a unos 2 segundos el ciclo de arranque, lectura y conexión Wifi.

- Sobre las mediciones, hay que destacar en el código lo siguiente:

  La medición del nivel de líquido es muy precisa la mayor parte de las ocasiones, pero hay algunos momentos en los que por alguna razón la medición no es del todo correcta. Para minimizar dicho impacto, se calcula la media de un conjunto de 10 mediciones. De esta forma la medición será siempre lo más cercana a la realidad. Para hacer la media, tomamos las medidas con un bucle, las vamos sumando y para finalizar la luma la dividimos entre 10.

  Para hacer cada una de las pedidas, enviamos un pulso de 10 microsegundos, para lo que se activa el PIN TRIGGER del sensor:

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

  y obtenemos el tiempo observando lo que tarda en activarse el pin ECHO:

```
duration=pulseIn(echoPin, HIGH);
```

  Una vez obtenida la distancia medida media, se calcula la distancia hasta el líquido en centímetros a partir de la velocidad del sonido (dividiendo por 58,2):

```
distance = duration/58.2;
```

  La explicación para esos 58.2 es que a 340m/s que es la velocidad del sonido, la onda tarda 29.1 µs en recorrer un centímetro, por lo que para el recorrido de ida y vuelta tardará 58.2 µs.

  Para la medición del nivel de batería, se utiliza la función incorporada de la librería del ESP8266 ESP.getVcc(). El módulo, en su entrada analógica (A0) ya tiene un divisor de tensión incorporado, y consta de un conversor analógico digital de 10bits. Por tanto, tendremos un valor máximo de 2^{10}=1024. De esta forma para medir el nivel de batería, dividimos entre el valor máximo:

```
float volt = ESP.getVcc()/1024;
```

- En relación a la lectura y salvado de datos en la memoria hay que indicar que se utiliza para poder salvar los datos de conexión WIFI así como de configuración de la garrafa que son los siguientes:
    - WIFI: SSID y PASSWORD
    - ThingSpeak: Canal asociado al sensor y APIKey de escritura para ese canal.
    - Garrafa: El tipo de garrafa (grande o pequeña), para que el dispositivo de alerta sepa cuando informar de que se está acabando el líquido.

  Para el uso de la memoria se utiliza la librería EEPROM.h. Dicha librería efectúa una emulación de memoria EEPROM, dado que el ESP8266 no tiene memoria de ese tipo, sino memoria Flash. Lo que se hace para el uso de dicha memoria es definir un rango de memoria a usar:

```
EEPROM.begin(512);
```

  y se accede a cada una de las posiciones, para lectura o escritura con un índice a cada posición. También es posible definir una estructura de elementos de texto con longitud definida para hacer una lectura o escritura conjunta de los datos. Podemos ver la estructura creada para realizar la lectura de los datos de memoria en el arranque del dispositivo:

```
struct {
  char ssid[32] = ""; //(pos 0 to 31)
  char passphrase[64] = ""; //(pos 2 to 95)
  char jagSize[1]= ""; //(pos 96)
  char myChannelNumber[10] = "";// (pos 97 to 106)
  char myWriteAPIKey[20] = ""; //(pos 107 to 126)
  char jagType [1] ="""; //(pos 127) 1=chlorine, 2=PH, 3=Other
} memory
```

  Y para leer los datos usamos la siguiente función que nos rellena la estructura definida anteriormente:

```
EEPROM.get(0,memory);
```

  Para realizar la escritura de los datos desde el modo de configuración, una vez obtenidos éstos desde la página Web servida en modo Access Point, los vamos añadiendo carácter a carácter con la siguiente función:

```
EEPROM.write(posición,dato);
```

- El esquema de funcionamiento del "modo configuración" (AP+ Webserver Reconfig DATA) es el siguiente:
    - En primer lugar, se prepara la página, que es un formulario html.

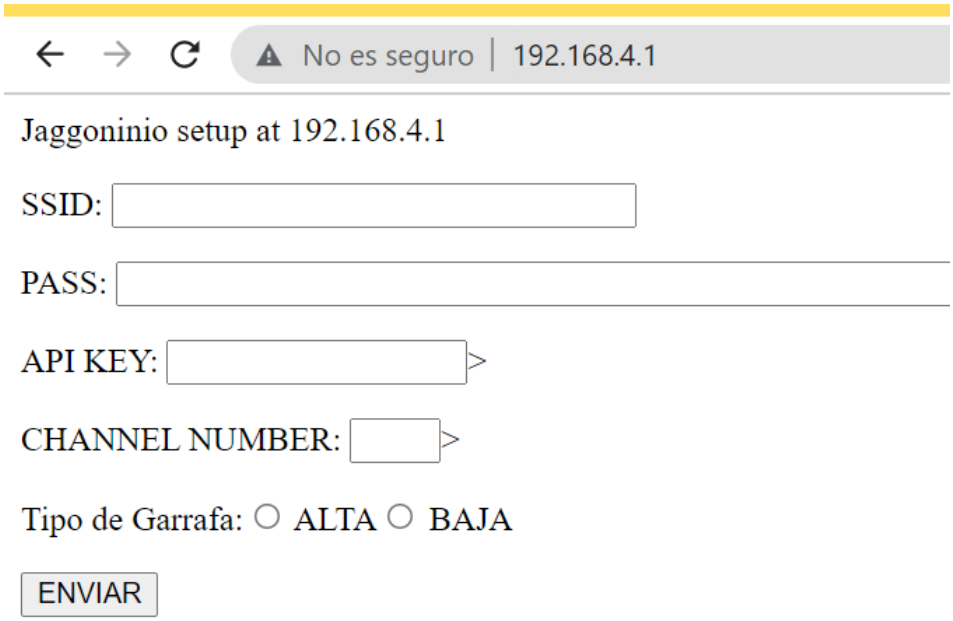


Ilustración 22. Web Access Point Sensor

    - A continuación, se arranca la Wifi del dispositivo en modo Punto de acceso:

      WiFi.mode(WIFI_STA)

      Que nos crea una red Wifi con SSID Jagonino y que asigna una IP por protocolo DHCP a los clientes dentro del rango 192.168.4.x

6. DISPOSITIVO DE ALERTA:

6.1. HARDWARE DEL DISPOSITIVO DE ALERTA:

Este es el esquema del dispositivo de alerta, donde se puede apreciar que es muy similar al dispositivo de medición. Se mantiene el módulo ESP8266 D1-MINI, el Step Up MT-3608 para garantizar los 5V de alimentación del display, el NMOS-FET 2N7000 junto a la resistencia R1 de 10K para activar la alimentación del Display con 5V solo cuando tenga que mostrar alarmas, el pulsador SW1 junto al condensador, para poder controlar el tiempo que ha sido pulsado y poder así activar el modo de configuración o simplemente para despertar al dispositivo y por último, como diferencia fundamental, en vez del sensor de ultrasonidos, en este caso tenemos el Módulo LCD (Liquid Crystal Display) 1602A.

Podemos observar una diferencia llamativa con el dispositivo de medición, y es que se encuentra “tachada” la conexión entre el pin D0 y el RST. La explicación para esto son los dos motivos siguientes:

- Este dispositivo debe mostrar alertas cuando haya algún usuario delante observándolas, por lo que no tiene sentido que se despierte periódicamente para mostrar las alertas. La conexión entre D0 y RST se realiza para que el dispositivo puede “despertarse” tras un tiempo, activando el RST desde el pin D0. Por tanto, al contrario que el dispositivo de medición que si debe despertarse cada cierto tiempo, el dispositivo de alarma no requiere de dicha conexión.
- Inicialmente se había planteado poner el dispositivo en modo “Deep Sleep” durante el máximo tiempo permitido por el ESP8266, y simplemente volver a dormirlo tras despertarlo sin realizar ninguna acción adicional. Según las especificaciones de *Espressif* sobre el módulo ESP8266, el tiempo máximo teórico que puede estar dormido viene dado por la expresión:

  $$2^{11}-1 \text{ , en } \mu s = 2147s$$

  Podemos encontrarlo en la página 15 del documento de Espressif “ESP8266 Non-OS SDK - API Reference” (24).

  Ese tiempo se corresponde con unos 35 minutos, tiempo inferior al deseado. Realizando pruebas para que el dispositivo estuviese el mayor

tiempo posible en modo *Deep Sleep,* se pudo comprobar que con valores superiores, efectivamente el dispositivo se queda en un estado “extraño” con todos los pines activados, incluido el LED interno, por lo que se descartó y se decidió dormir el dispositivo hasta que se presione el pulsador.

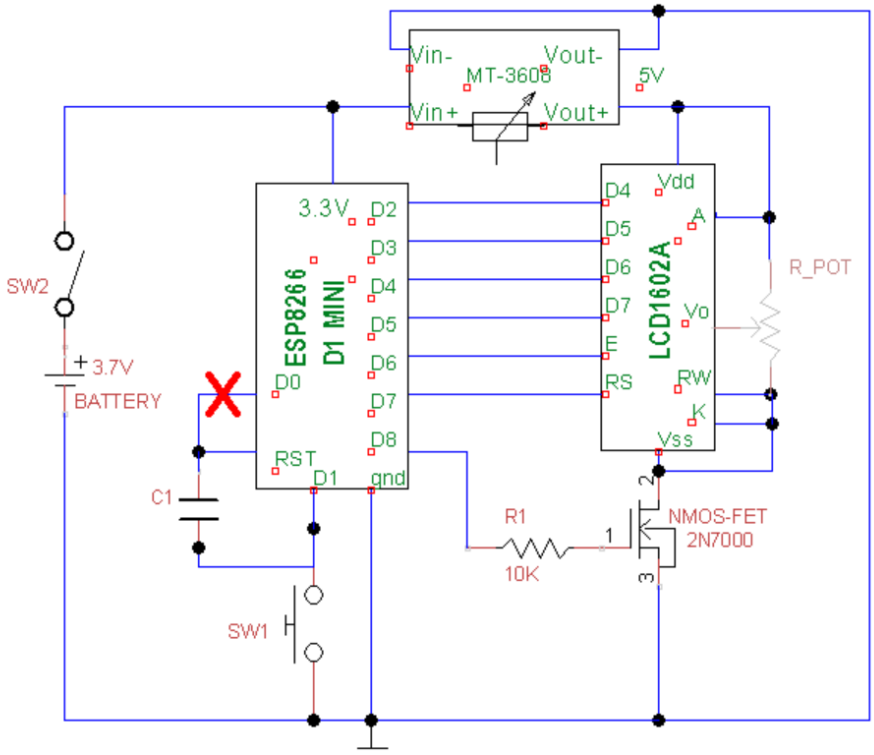


Ilustración 23. Esquema dispositivo de alarma

A continuación, se describe la funcionalidad de los 16 Pines del módulo LCD1602A y su conectividad en el módulo:

Pin No.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply Voltage for logic
3	VO	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A	–	LED +
16	K	–	LED –

- Vdd: Alimentación 5V. Va conectado con la salida del Step Up MT-3608, que tiene la salida ajustada a 5V.
- Vss: conexión a los 0V. Se conecta al drenador del MOSFET 2N7000, de esta forma, en el momento que interese activar la pantalla LCD, lo que hay que hacer es dar tensión entre la puerta y el surtidor con la finalidad de que se active la circulación drenador-surtidor.
- Vo: Permite controlar el nivel de iluminación de la pantalla. Va conectado al control de un potenciómetro que hace de divisor de tensión, y de esta manera permite controlar el nivel de tensión que le llega al Pin para ajustar el nivel de iluminación de la pantalla.
- RS: Si está con tensión, se envían datos para ser mostrados en la pantalla, en caso contrario, se le envían instrucciones. Lo controlamos con el puerto D7 del ESP8266.
- R/W permite indicar si estamos en modo lectura o escritura hacia/desde el módulo. Como se utiliza solamente en modo escritura, lo tenemos conectado, junto a Vss a drenador del Mosfet.
- E: Activa o desactiva el Display. Lo controlamos directamente con el puerto D6.
- Datos DB0 a DB7. Son el bus de datos. Solo se utilizan los 4 bits más significativos (DB4 a DB7) y van conectados a los Pins D2, D3, D4 y D5 del ESP8266.
- A y K: son los pines del led de la luz de fondo de la pantalla. A se conectará a 4 o 5 volts y K a gnd.

Al igual que en el dispositivo de medición, controlamos la alimentación del Display con el PIN D8 del ESP8266. Además, el PIN D6 del ESP8266 controla la habilitación o no del Display a través de su pin E. Se ha comprobado que ambos

Pines se mantienen sin tensión en el modo Deep Sleep y que no interfieren en el proceso de arranque del módulo.

A continuación, se muestran algunas fotos del circuito funcionando en la placa de pruebas en algunos de los diferentes puntos de funcionamiento:

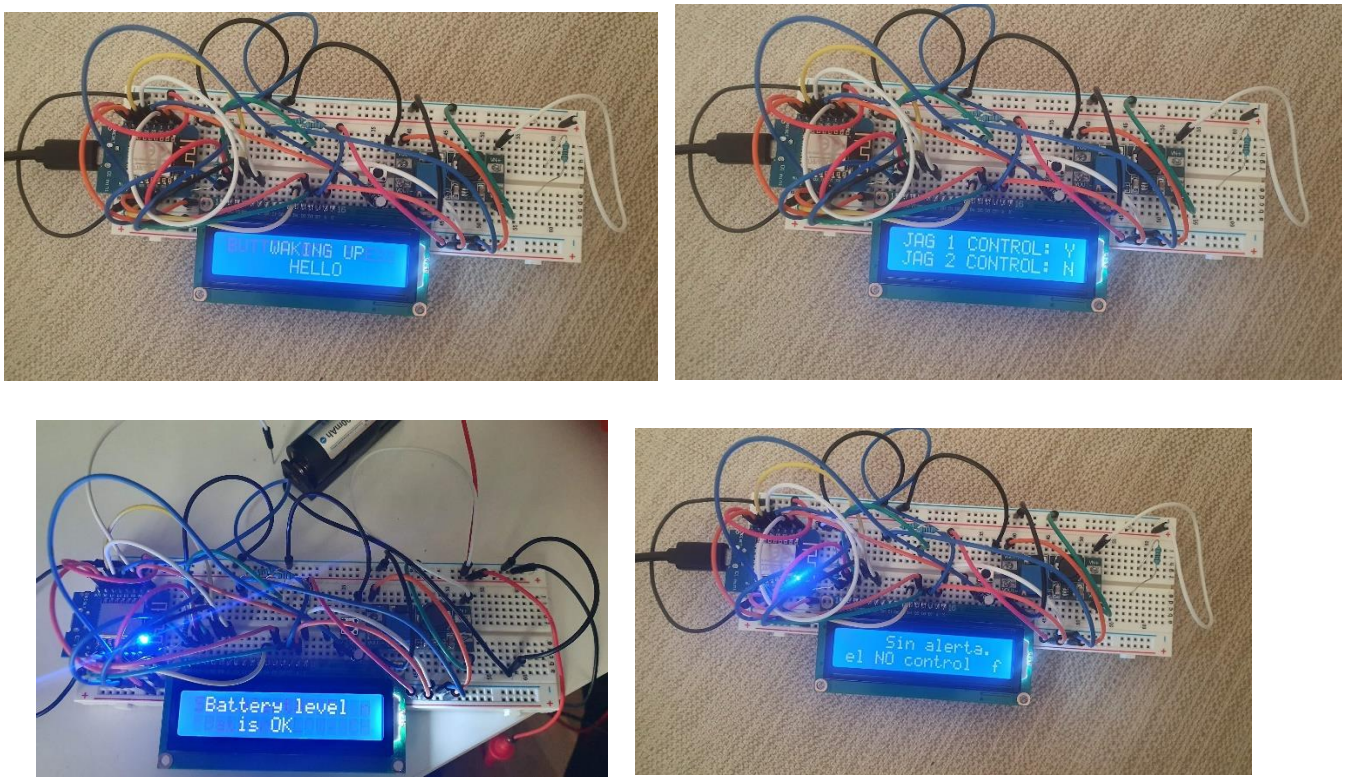


Ilustración 24. Placa de pruebas Dispositivo de alarma

Y una vez probado, se traslada al circuito PCB de soldadura:

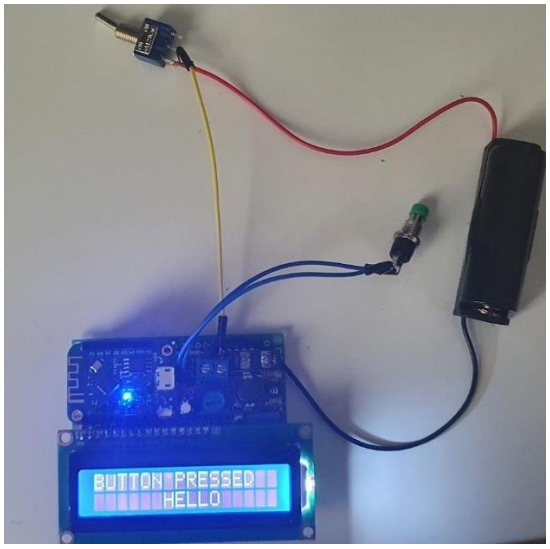
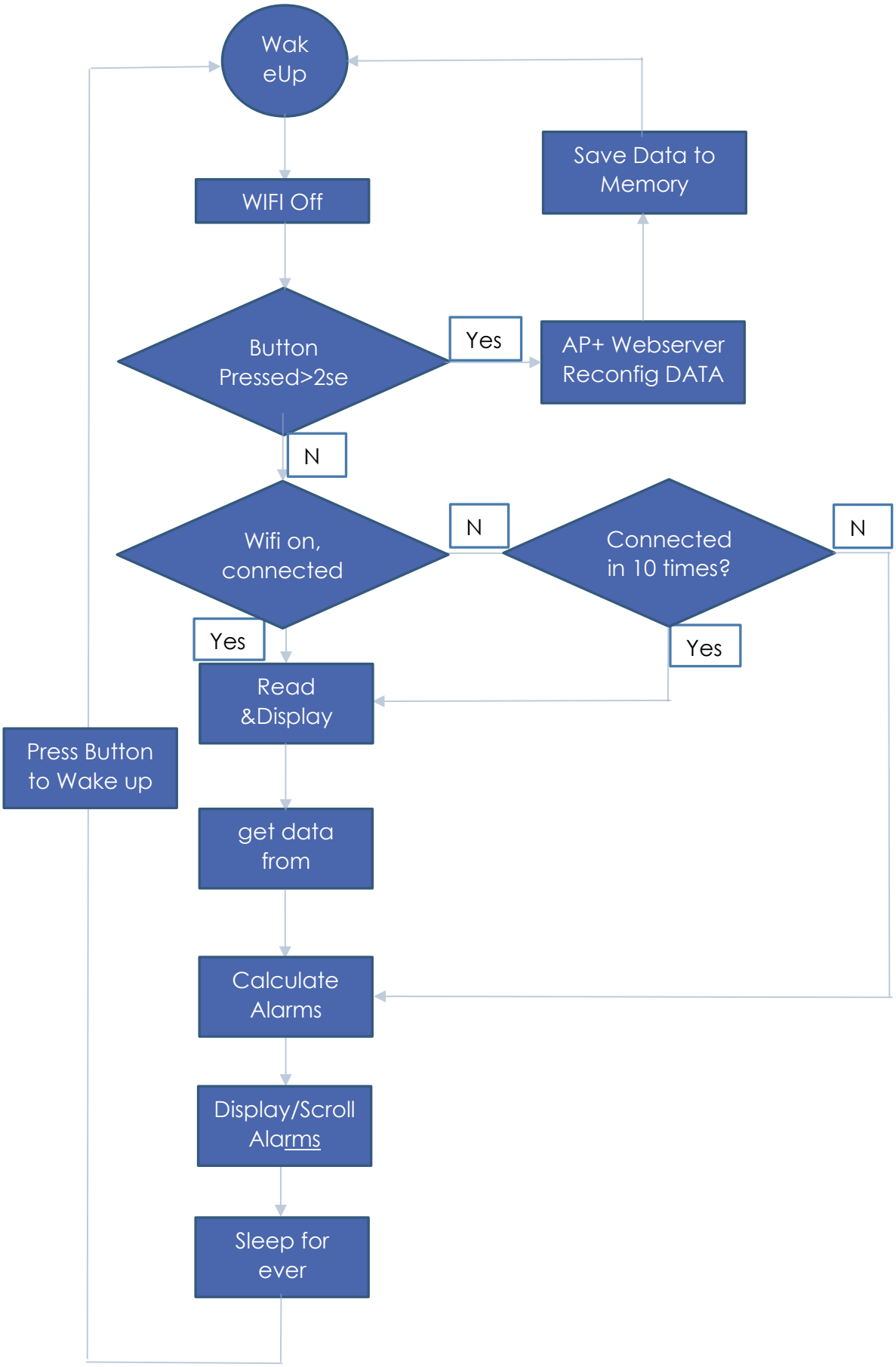


Ilustración 25. PCB soldadura dispositivo de alarma

6.2. CÓDIGO DEL DISPOSITIVO DE ALERTA



A continuación, se detalla información de los puntos más destacables del código del dispositivo de alarma:

- El primer punto destacable y diferencial del código del dispositivo de alarma es que éste permanecerá dormido mientras no se le despierte pulsando el botón. Para ello, indicamos que el tiempo en el que debe estar dormido es 0.

  ```
  const int sleepSeconds = 0;
  ```

  Como hemos visto en el punto anterior, esto es así, por un lado, para que no consumir batería cada vez que se despierte sin que nadie esté mirándolo y, por otro lado, no se puede indicar un tiempo para el estado Deep Sleep superior a 2147 segundos.
- ThingSpeak: Para realizar la lectura de los datos de los sensores, se utilizan las funciones de lectura de campos de la librería propia de ThingSpeak (readIntField, readFloatField,y readStringField) según el tipo de datos a leer sea un integer, float o String:

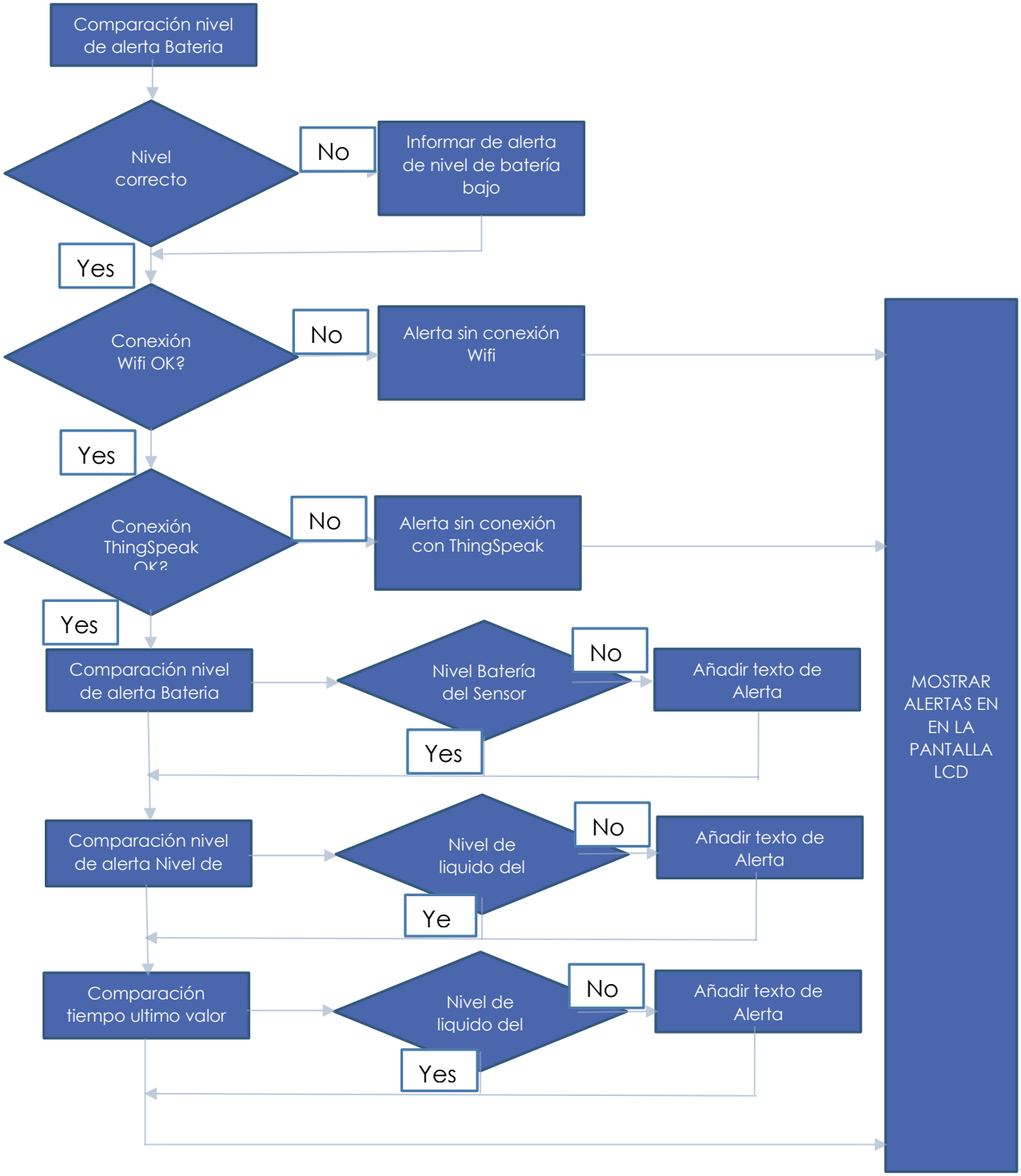
  ```
  String jagType1=ThingSpeak.readStringField(valorMemoryChannel, 1, memory.myReadAPIKey1);
  float volt1=ThingSpeak.readFloatField(valorMemoryChannel, 5, memory.myReadAPIKey1);
  jagValue=ThingSpeak.readIntField(valorMemoryChannel, 3, memory.myReadAPIKey1);
  int jagSize1=ThingSpeak.readIntField(valorMemoryChannel, 4, memory.myReadAPIKey1);
  ```

  Para leer los segundos desde la última entrada usamos la función readRaw, que permite simular el uso directo de la API, pero usando la librería de ThingSpeak

  ```
  String response = ThingSpeak.readRaw(valorMemoryChannel, String(String("/fields/") + String(3) + String("/last_data_age")), memory.myReadAPIKey1);
  ```

- Alertas

  La lógica de alertas funciona siguiendo el siguiente diagrama:



Como se puede comprobar en el diagrama anterior, primero se comprueba la propia batería y se muestra la información por pantalla para a continuación realizar la conexión WIFI y la conexión con ThingSpeak para leer los datos enviados por el módulo de medición de nivel de liquído. En caso de fallo, se muestra la información por pantalla y no se sigue con el

resto de las comprobaciones. En caso contrario se comprueba, para cada sensor, su nivel de batería, su nivel de líquido y el tiempo transcurrido desde el último envío a Thingspeak, Si cualquiera de los 3 tiene un valor que justifique la alerta, se incorpora al texto de alama, y tras las 3 comprobaciones se envía el texto completo al Display.

- Consumo. Con la intención de minimizar el consumo, se utilizan las mismas medidas que las usadas en el módulo de medición y que están relacionadas con la funcionalidad Wifi:
    - Desactivación de la persistencia
    - Desactivación de la calibración de radiofrecuencia
    - Desactivación del WIFI durante el arranque, mientras no tenga que efectuarse ninguna acción con conexión.
- Pantalla. Se emplean las siguientes 4 funciones incluidas en la librería LiquidCrystal:

  lcd.begin(16, 2), lcd.clear(); lcd.setCursor(0,0); lcd.print(" texto").

  Para mostrar el conjunto de alertas por pantalla, se ha creado una función de *"scroll"* específica para hacer un desplazamiento del texto de derecha a izquierda de todo el texto. Se incorporan las alertas de cada uno de los sensores en un String de texto e independientemente de la longitud del texto, se muestra un texto en cada fila del LCD para a continuación ir desplazando dicho texto de derecha a izquierda. Para conseguirlo usamos la función substring, y un blucle.
- La gestión de la memoria Flash para almacenar la información relativa a la conectividad y control de los sensores se realiza de la misma forma que en el dispositivo de medición de nivel de líquidos (con la librería EEPROM), pero con alguna modificación en los valores guardados. Concretamente, se mantiene el SSID y la CLAVE de conexión WIFI, pero como se pueden controlar hasta dos dispositivos de medición, se añaden los datos relativos a cada uno de los dos canales (API de lectura y numero de canal de ThingSpeak) así como la información de si deben o controlarse o no por el dispositivo.

7. DISEÑO DE CAJAS/ENCAPSULADOS DE LOS MÓDULOS

El diseño del tapón de la garrafa se ha realizado con Fusion360 de Autodesk para ser impreso por una impresora 3D. Para el diseño se crean “sketches” con las formas, a las que luego se le dan volumen. Para crear los “huecos”, se crean las figuras necesarias para luego “cortarlas” según se necesita.

En primer lugar, se crea un cilindro y a partir de ahí se van creando los huecos en el interior ajustados a las medidas del dispositivo y la pila.

Podemos ver a continuación una secuencia de imágenes de la creación del tapón de garrafa para el dispositivo de medición, así como del portapilas y de la tapa para el tapón:

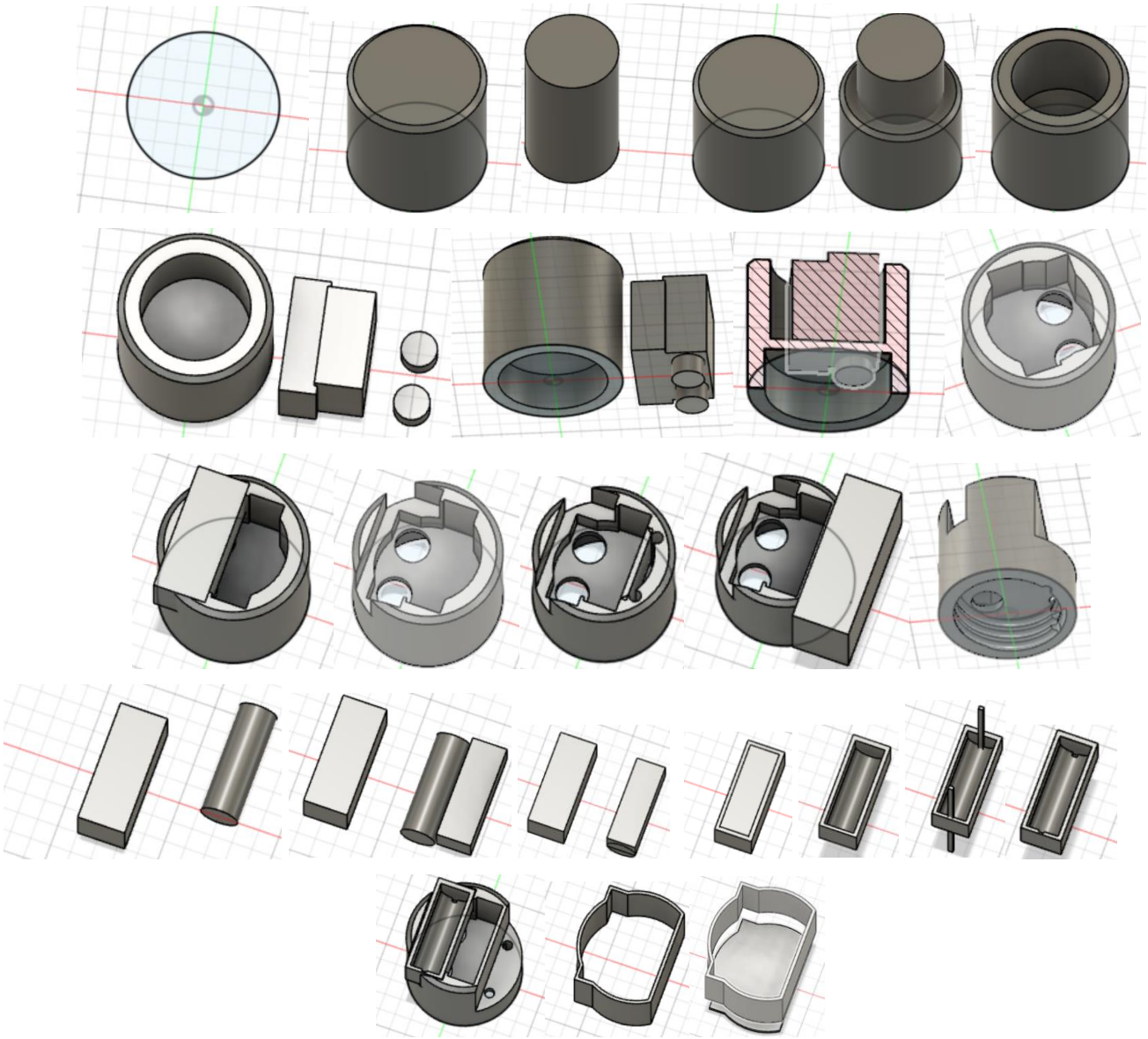


Ilustración 26. Imagenes Autodesk Fusion

El resultado final, para las 3 piezas es el siguiente:

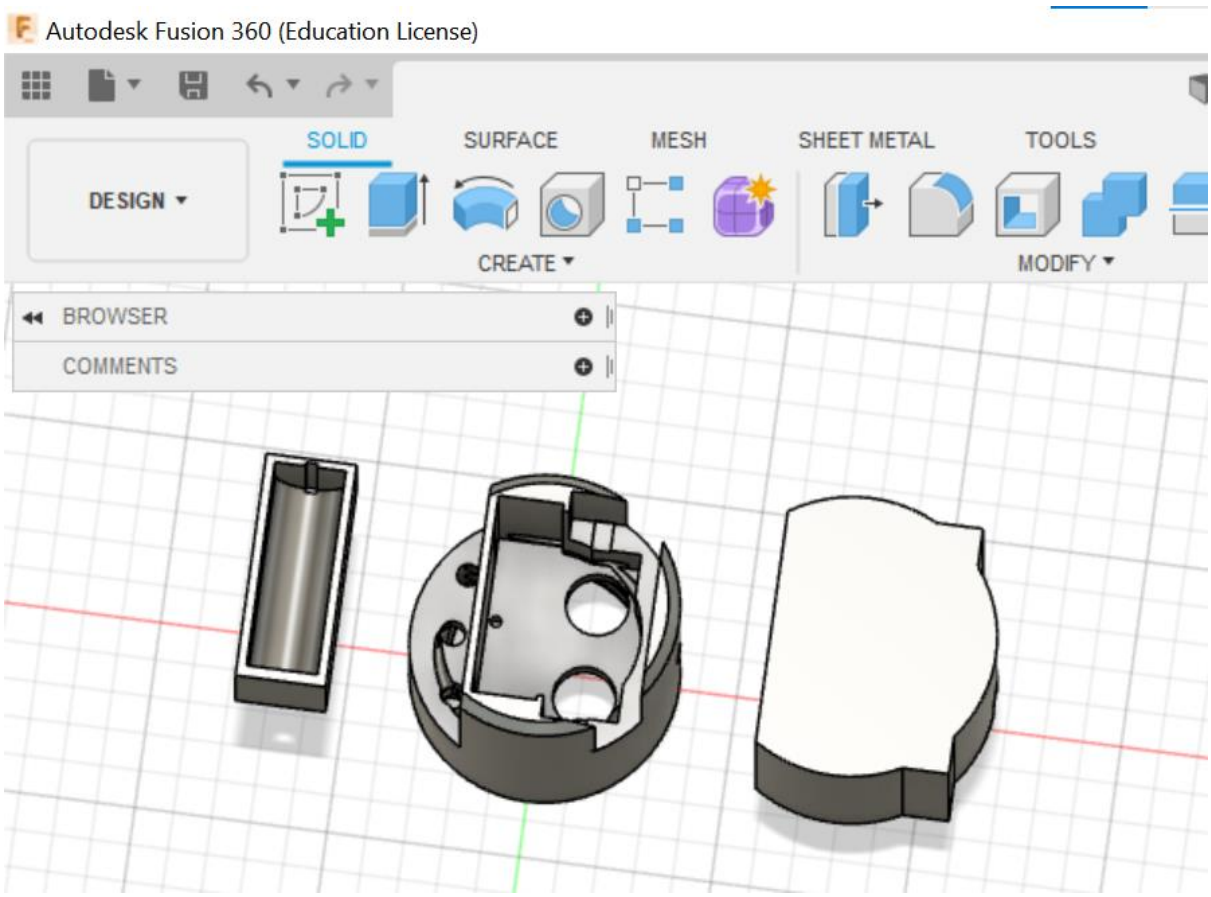


Ilustración 27. Autodesk Fusion, Tapón

El tapón, con el portapilas y la tapa, ya impresos quedan así:

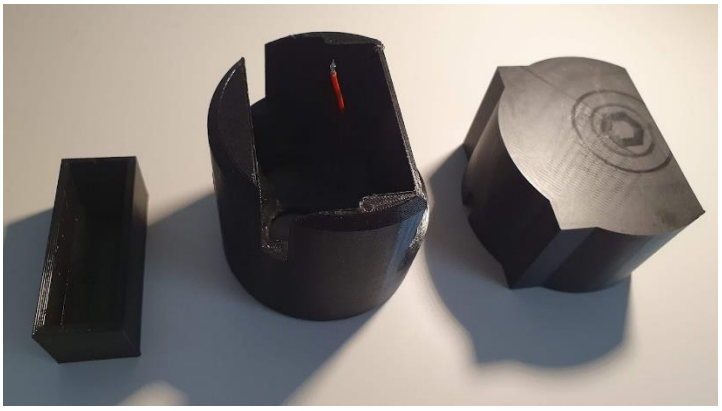


Ilustración 28. Despiece impreso en 3D del tapon

Y con el módulo de medición y la pila ya incorporados:

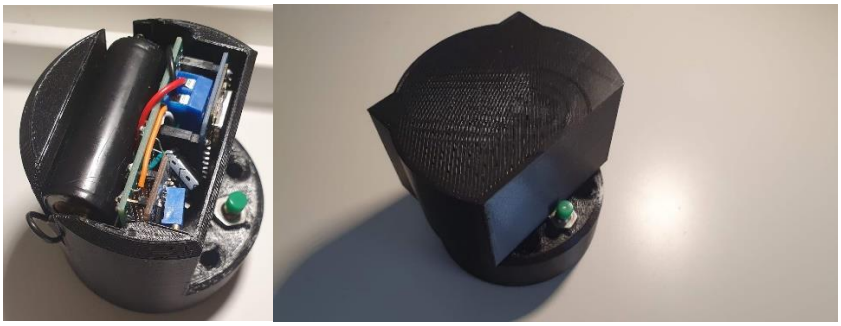


Ilustración 29. Dispositivo de medición encapsulado

Para el dispositivo de alertas, se ha comprado una caja ajustada en tamaño, con tapa transparente, para que quepan el módulo y la batería en su interior y pueda leerse el Display sin problemas. A la caja se le ha realizado un agujero para poder pasar el pulsador que activa las mediciones:

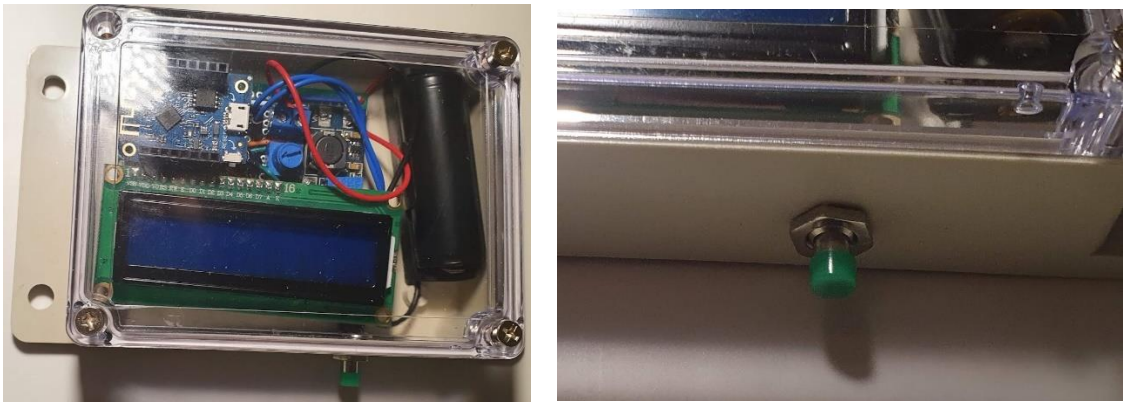


Ilustración 30. Caja dispositivo de alerta

8. PRUEBAS

El procedimiento seguido para realizar las pruebas de funcionamiento es el siguiente:

1º Preparamos el dispositivo de medición y la garrafa:

a) Pulsamos más de 2 segundos el botón para entrar en modo configuración y configuramos conexión a internet, datos de Thingspeak y del tipo de la garrafa.

Al pulsar 2 segundos, se activa el modo Access Point con el Webserver. Seleccionamos la Wifi JagoninoWIFI y nos conectamos a la web 192.168.4.1 para configurar el dispositivo:

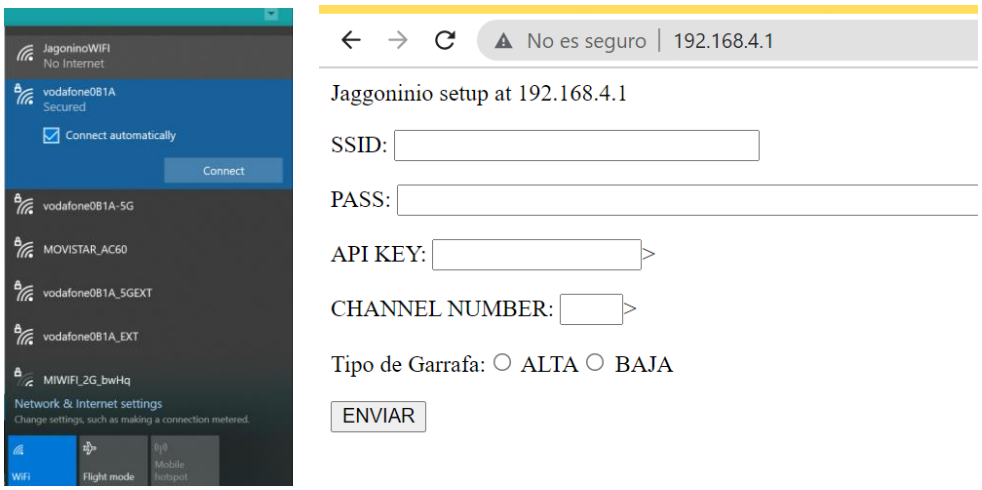


Ilustración 31. Conexión Access Point WIFI, web server Dispositivo de medición

Tras finalizar la configuración, los datos han sido almacenados en memoria y el dispositivo pasa a estado dormido de forma correcta. A partir de este momento el dispositivo se despertará cada media hora aproximadamente para tomar nuevas medidas y enviar los datos a ThingSpeak.

b) Repetimos la operación con el dispositivo de alarma.
Tras finalizar la configuración, los datos han sido almacenados en memoria y el dispositivo pasa a estado dormido de forma correcta. A partir de este momento el dispositivo se despertará cada vez que pulsemos el botón, para conectarse a ThingSpeak, recoger las últimas mediciones disponibles, y alertar cuando sea necesario.

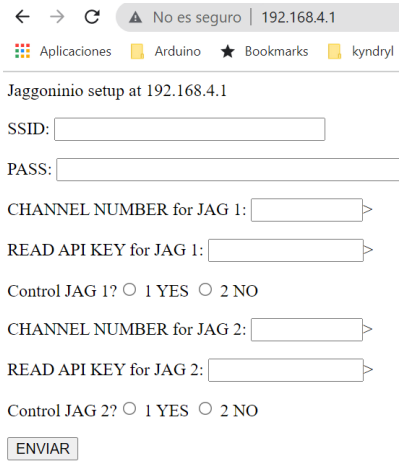


Ilustración 32. Web dispositivo alerta

c) Llenamos la garrafa con líquido (unos 24cm).
d) Colocamos en dispositivo en la tapa y la colocamos en la garrafa.
e) Verificamos que se envían los datos a Thingspeak de forma correcta.
f) Verificamos cuando corresponda que hay alerta en el dispositivo de alama (han de quedar 5 cm o menos, la distancia ha de ser superior a 20cm).

Ilustración 33. Garrafa, 24cm

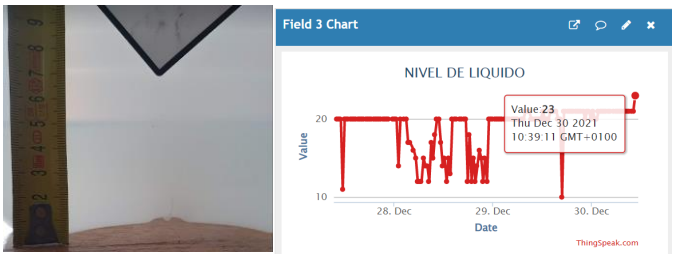
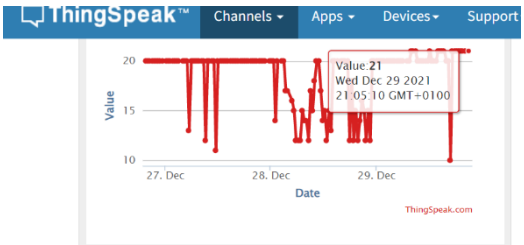


Ilustración 34. Garrafa y Thingspeak con 5cm de líquido

g) Se cambia el nivel de líquido en la garrafa y repetimos varias veces los puntos d) a g).

Se comienza poniendo en la garrafa entre 7 y 8 cm de agua, de tal forma que la distancia medida es de 21 cm y no hay alerta. Verificamos que llega la medida a Thingspeak y que no hay alerta en el dispositivo de alerta.



Al sobrepasar los 22 cm, se activa la alerta:

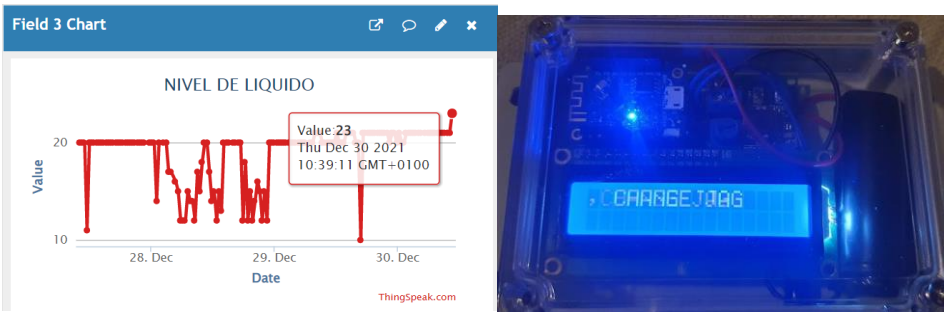
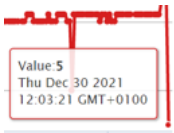
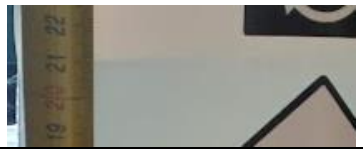
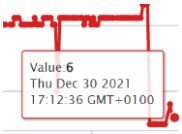
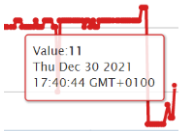
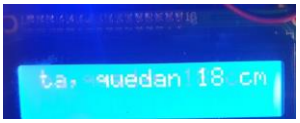
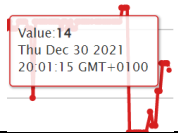
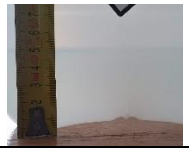
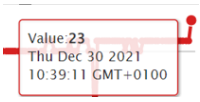


Ilustración 35. Aviso de Cambio de Garrafa

Se realizan unas cuantas mediciones más, donde podemos comprobar el correcto funcionamiento, con un margen de error de 1cm aproximadamente, más que suficiente para el dispositivo.

Nivel en garrafa	ThingSpeak	Alerta	Nivel mostrado en dispositivo
24cm	5cm 	No	24cm
21cm 	6cm 	No	23cm
19cm	11cm 	No	18cm 
15cm	14cm 	No	15cm
8cm	21cm	No	8cm
5cm 	23cm 	Si	ALERTA CAMBIO DE GARRAFA

Para poder realizar las pruebas de medición de batería, se realiza una modificación en el código del módulo de medición, para que éste despierte cada 60segundos, en vez de cada 1800, multiplicando el consumo considerablemente:

```
// sleep for this many seconds
//const int sleepSeconds = 1800;
const int sleepSeconds = 60;
```

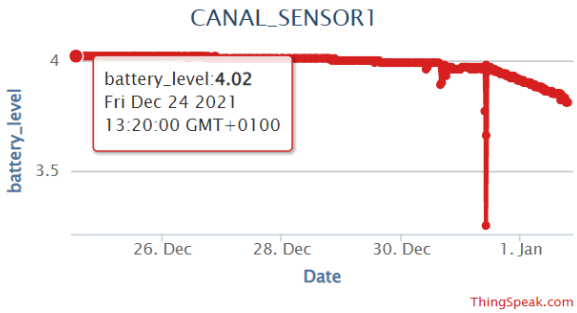
Y a continuación se compara el valor real medido con un multímetro con el obtenido por el dispositivo y enviado a Thingspeak. Se toman distintas muestras mientras va bajando la batería disponible:

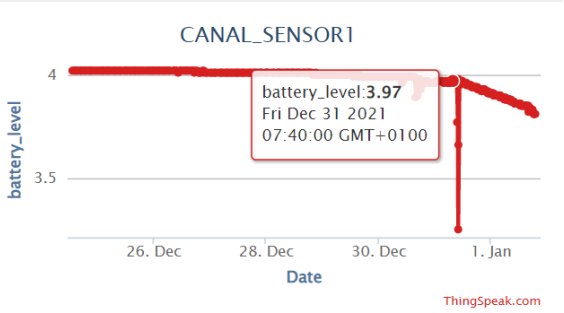
MEDIDA REAL (multímetro)	ThingSpeak	Diferencia
4.04	3.96	0.08
4.03	3.94	0.09
4.01	3.92	0.09
4.0	3.90	0.10
3.96	3.86	0.10
3.93	3.84	0.09
3.92	3.81	0.11
3.89	3.79	0.10
3.84	3.75	0.09

Se comprueba que las medidas del dispositivo difieren de las medidas del multímetro entre 0.08 y 0.11 Voltios, siendo perfectamente válidas para controlar el nivel restante de batería.

Analizando las gráficas obtenidas de Thinkgspeak a lo largo de varios días, se observa lo siguiente:

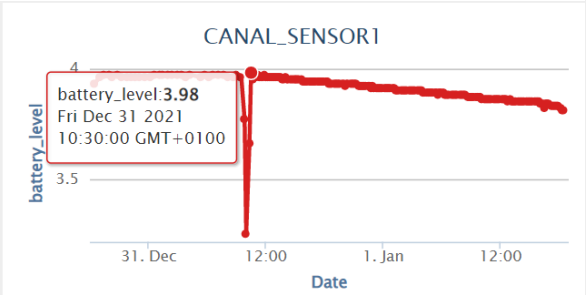
- En una semana (24/12 a las 14:20 hasta el 31/12 a las 07:40) se produce una variación de 0,05 V (baja de 4,02 a 3,97). Como la batería tiene una zona lineal de unos 1 V (4,2 a 3,2), se puede esperar un tiempo de vida de unas 20 semanas.

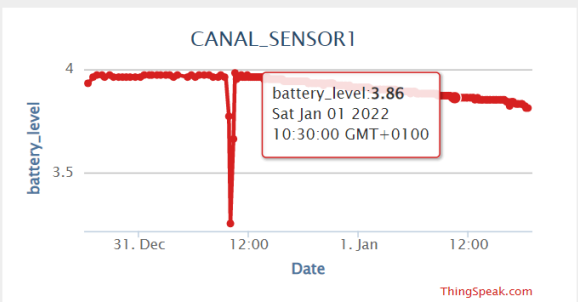




- Tras las modificaciones en el código para que las mediciones se realicen cada 60 segundos, se puede observar el aumento la pendiente de la recta. En 1 día la tensión se reduce en 0,12 V (3,98 a 3,86). Por lo que podemos

comprobar que se ha multiplicado el consumo en un factor de 34 (0,12 / 0,05/7). El valor es el esperado (hemos multiplicado por 30 el número de ejecuciones) y confirma que la batería tiene una duración estimada en unas 20 semanas





- El resultado de las pruebas de consumo mejora las previsiones considerablemente y demuestra que las optimizaciones realizadas para minimizar los picos de consumo en el arranque del dispositivo han dado sus frutos.

9. VALORACIÓN ECONÓMICA

En la siguiente tabla podemos ver los costes de los diferentes componentes empleados en el diseño de los prototipos, considerando 2 dispositivos de medición y 1 dispositivo de alerta:

Producto	Unidades	Precio unitario (€) (con IVA)	Precio Total
ESP8266 D1-MINI Compatible	3	1,25	3,75
Resistencia 10K	3	0,1	0,3
Potenciómetro 10K	1	0,5	0,5
Display LCD 1602A	1	3	3
MOSFET 2n7000	3	0,2	0,6
Sensor de ultrasonidos HC-SR04	2	1	2
StepUp Mt3608	3	0,5	1,5
Placa PCB	3	0,5	1,5
Cableado	3	0,5	1,5
Batería 18550	3	4	12
pulsador	3	1	3
interruptor	3	1	3
TOTAL			**32,65 €**

A esto habría que añadir el hilo para la creación del tapón para el dispositivo de medida y la caja para el dispositivo de alarma. Un royo tiene un coste de unos 20 euros, y el consumo de hilo es de no más de un cuarto de royo, por lo que se puede considerar un coste de 5€.

Por último, se tiene que considerar el coste de la mano de obra de unas 2 horas de trabajo de ensamblaje de piezas y programación de las placas, a 15€ por hora, son 30€ adicionales. Esos 15 € por hora salen de un coste empresarial de 33000€/año considerando 2200 horas al año de trabajo.

Por lo tanto, el coste total real sería de:

Coste de las piezas	32,65€
Coste la impresión 3D	5€
Coste de la mano de obra	30€
TOTAL	**67,65€**

Para obtener un beneficio de un 30%, el precio del producto debería ser de 88€.

Considerando que no se ha podido encontrar en el mercado un producto similar, podemos asegurar que, aunque no haya un volumen alto de ventas, estas se producirán con un precio de unos 90€ para un producto novedoso, obteniendo un beneficio de unos 22 € en cada venta.

10. CONCLUSIÓN

Los objetivos principales del proyecto, descritos al principio de este documento, que eran diseñar un dispositivo de medición de nivel de líquido de piscina, y un dispositivo de alerta, siendo ambos autónomos, de pequeño tamaño y de bajo consumo se han conseguido con la realización del proyecto. Los puntos estudiados en mayor detalle, claves en un mundo de máxima evolución de la tecnología IoT, han sido el tamaño y el consumo, siempre dentro del mundo de Arduino. Por ello se realizó un estudio previo y se compraron algunos dispositivos para poder hacer comparaciones de consumo, teniendo problemas con la entrega de uno de ellos

consecuencia de la crisis global de suministros que estamos viviendo este 2021. Ha sorprendido observar las grandes diferencias existentes entre los consumos indicados por el fabricante de los microcontroladores y el obtenido en las mediciones de los módulos probados, denotando, por parte de muchos fabricantes, una falta de compromiso con el bajo consumo requerido por el mundo IoT. Llamó especialmente la atención que, en modo bajo consumo, se mantuviese activo un LED en uno de los módulos.

Otros de los aspectos estudiados en detalle fueron la comunicación con las plataformas de almacenamiento de datos IoT, concretamente ThingSpeak, y su integración con Arduino, así como el diseño de formularios básicos html y como hacer uso de los datos obtenidos en el formulario para el código, y el almacenamiento de datos en la memoria flash del módulo ESP8266. Por último, ha sido muy enriquecedor aprender a usar el Software Fusion de Autodesk para la generación de objetos sólidos para luego poder obtener las impresiones 3D de los objetos creados.

En resumen, han sido unos meses de mucho trabajo, mucha investigación, pero con la satisfacción de haber cumplido con los objetivos marcados al inicio, obteniendo un producto real y efectivo que poder utilizar en la vida real.

11. PRÓXIMOS PASOS

Los dispositivos obtenidos, pueden evolucionarse hasta llegar a convertirse en un producto incluso comercial, ya que no existe otro equivalente en el mercado. Para ello habría que realizar una seria de mejoras sobre el producto obtenido tras el presente Trabajo de Final de Grado. Concretamente se podría avanzar en los siguientes aspectos:

- Placa. Deberá diseñarse una placa integrada profesional
- Caja. Deberá diseñarse una caja a medida para el dispositivo de alarma, al igual que se hizo para el tapón de la garrafa.
- Consumo. Para un producto comercial, debería sustituirse el módulo Wemos ESP8266 D1 mini, por un módulo ESP8266 o ESP32 sin circuitería añadida que perjudica seriamente al consumo, lo que permitiría conseguir incluso más de un año de duración de la batería.

- Seguridad. Se debería emplear sistemas más seguros y eficientes para el envío de los datos, por ejemplo, ThingSpeak soporta MQTT encriptado. Adicionalmente, para también podría emplearse https para las páginas de configuración de los dispositivos, aunque es menos prioritario por tratarse en cualquier caso de tráfico local.

12. BIBLIOGRAFÍA

1. Ramgir, M. *Internet of Things (1st edition). .* s.l. : Pearson Education India., 2019.

2. Wikipedia. [En línea] 29 de 09 de 2021. [Citado el: 30 de 09 de 2021.] https://es.wikipedia.org/wiki/Internet_de_las_cosas.

3. arduino, . *eldiario.es.* [Online] 08 7, 2014. [Cited: 10 23, 2021.] https://www.eldiario.es/hojaderouter/tecnologia/hardware/arduino-hardware-libre-revolucion-origenes-historia_1_4719250.html.

4. Arduino. Trademarks and Copywrite Notice. *Arduino.* [Online] [Cited: 10 23, 2021.] https://www.arduino.cc/en/Main/CopyrightNotice.

5. ESP8266. *Wikipedia.* [Online] Wikipedia. [Cited: 10 23, 2021.] https://en.wikipedia.org/wiki/ESP8266#:~:text=The%20ESP8266%20is%20a%20low-cost%20Wi-Fi%20microchip%2C%20with,ESP-01%20module%2C%20made%20by%20a%20third-party%20manufacturer%20Ai-Thinker..

6. Ventura, Victor. esp8266. ¿qué módulo elegir? *polaridad.es.* [Online] 2018. [Cited: 10 24, 2021.] https://polaridad.es/esp8266-modulo-wifi-elegir-caracteristicas/.

7. Wikipedia, Commons. WeMos_D1_Mini_front.jpg. *Commons Wikipedia.* [Online] 04 30, 2017. [Cited: 10 24, 2021.] https://commons.wikimedia.org/wiki/File:WeMos_D1_Mini_front.jpg.

8. T-OI PLUS ESP32-C3 with Battery Holder. *lilygo.cn.* [Online] [Cited: 10 24, 2021.] http://www.lilygo.cn/prod_view.aspx?TypeId=50044&Id=1361&FId=t3:50044:3.

9. IoT e IA en el teletrabajo. *Io University.* [Online] 11 14, 2020. [Cited: 10 30, 2021.] https://www.ie.edu/insights/es/articulos/iot-e-ia-en-el-teletrabajo/.

10. La Inteligencia Artificial, clave para el futuro del IoT. *Deloitte.* [Online] [Cited: 10 30, 2021.] https://www2.deloitte.com/es/es/pages/operations/articles/inteligencia-artificial-futuro-internet-of-things.html.

11. Las piscinas inteligentes ya son una realidad. *higieneambiental.com.* [Online] higieneambiental.com, 07 15, 2021. [Cited: 10 31, 2021.] https://higieneambiental.com/piscinas-inteligentes.

12. Fluidra. A connected, smart, responsible and modern pool. *www.Fluidra.com.* [Online] Fluidra, 09 2021. [Cited: 10 31, 2021.] https://www.fluidra.com/projects/connected-smart-responsible-and-modern-pool/.

13. Sigfox lanza una solución IoT para facilitar la gestión de piscinas. *TaiPriceBook.* [Online] TaiPriceBook Edición digital, 07 08, 2020. [Cited: 10 31, 2021.] https://taipricebook.es/sigfox-lanza-una-solucion-iot-para-facilitar-la-gestion-de-piscinas.

14. MEDIR TEMPERATURA DE LÍQUIDOS Y GASES CON ARDUINO Y DS18B20. *https://www.luisllamas.e.* [Online] 06 27, 2016. [Cited: 10 31, 2021.] https://www.luisllamas.es/temperatura-liquidos-arduino-ds18b20/.

15. Tratamiento de agua de piscinas / ORP piscina. *okreformapiscina.net.* [Online] okreformapiscina.net. [Cited: 10 31, 2021.] https://okreformapiscina.net/tratamiento-de-agua-de-piscinas/orp-piscina/.

16. Luis del Valle Hernandez. ESP8266 Deep Sleep, cuánto consumen NodeMCU y Wemos D1 Mini . *programarfacil.com .* [Online] [Cited: 09 21, 2021.] https://programarfacil.com/esp8266/esp8266-deep-sleep-nodemcu-wemos-d1-mini/.

17. ESP32 Datadheet. *Espressif.com.* [Online] 2021. [Cited: 09 14, 2021.] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

18. LILYGO® TTGO T-OI PLUS. *lilygo.cn.* [Online] [Cited: 09 15, 2021.] http://www.lilygo.cn/prod_view.aspx?TypeId=50063&Id=1361&FId=t3:50063:3.

19. Espressif. ESP32 Sleep Modes. *Espressif.* [Online] [Cited: 11 15, 2021.] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html.

20. 779948/HC-SR04-pdf. *www.datasheet.es.* [Online] [Cited: 09 20, 2021.] http://www.datasheet.es/download.php?id=779948.

21. 2N700. *Alldatasheet.com.* [Online] [Cited: 10 10, 2021.] https://pdf1.alldatasheet.es/datasheet-pdf/view/50067/FAIRCHILD/2N7000.html.

22. MathLab. Thingspeak rest-api reference. *mathworks.com.* [Online] Mathworks. [Cited: 12 10, 2021.] https://es.mathworks.com/help/thingspeak/rest-api.html.

23. reducing-wifi-power-consumption-on-esp8266. *bakke.online.* [Online] 05 21, 2017. [Cited: 10 28, 2021.] https://www.bakke.online/index.php/2017/05/21/reducing-wifi-power-consumption-on-esp8266-part-1/.

24. Espressif. 2c-esp8266_non_os_sdk_api_reference_en. *espressif.com.* [Online] 09 2020. [Cited: 12 13, 2021.] https://www.espressif.com/sites/default/files/documentation/2c-esp8266_non_os_sdk_api_reference_en.pdf.

25. Las piscinas inteligentes ya son una realidad. *higieneambiental.com.* [Online] 07 15, 2021. [Cited: 10 29, 2021.] https://higieneambiental.com/piscinas-inteligentes.

26. Internet of Pools. *Fluidra.com.* [Online] [Cited: 10 29, 2021.] https://www.fluidra.com/es/sobre-fluidra/internet-of-pools.

27. Las piscinas inteligentes ya son una realidad. [Online] Higiene.Ambiental, 07 15, 2021. [Cited: 10 31, 2021.] https://higieneambiental.com/piscinas-inteligentes.

13. ILUSTRACIONES:

Ilustración 1. Fuente Arduino.com 10
Ilustración 2. Fuente: https://upload.wikimedia.org/wikipedia/commons/8/84/ESP-01.jpg 11
Ilustración 3: Fuente de las imágenes (6) 12
Ilustración 4. NodeMCU (6) 12
Ilustración 5. Wemos D1 Mini (7) 12
Ilustración 6. Fuente lilygo.com 13
Ilustración 7. Fuente: https://commons.wikimedia.org/wiki/File:HC_SR04_Ultrasonic_sensor_1480322_3_4_HDR_Enhancer.jpg 16

Ilustración 8.Fuente: Espressif ESP32 Datasheet 18
Ilustración 9. Fuente Amazon.es 18
Ilustración 10. Consumo ESP8266 Deep Sleep con el Sesnsor activo 20
Ilustración 11. Consumo ESP8266 DeepSleep y el sensor inactivo 21
Ilustración 12. Consumo ESP32 activo 21
Ilustración 13. consumo en modo deep sleep del módulo ESP32 22
Ilustración 14. Fuente www.datasheet.es 22
Ilustración 15. Fuente Nkon.nl/es 23
Ilustración 16. Fuente: commons.wikipedia.com 24
Ilustración 17. Esquema sensor 29
Ilustración 18. Pinout Wemos D1-Mini 29
Ilustración 19. Placa de pruebas del sensor 30
Ilustración 20. PCB de soldadura sensor 31
Ilustración 21. Serial monitor, error actualizando ThingSpeak 34
Ilustración 22. Web Access Point Sensor 37
Ilustración 23. Esquema dispositivo de alarma 39
Ilustración 24. Placa de pruebas Dispositivo de alarma 41
Ilustración 25. PCB soldadura dispositivo de alarma 41
Ilustración 26. Imagenes Autodesk Fusion 46
Ilustración 27. Autodesk Fusion, Tapón 47
Ilustración 28. Despiece impreso en 3D del tapon 47
Ilustración 29. Dispositivo de medición encapsulado 48
Ilustración 30. Caja dispositivo de alerta 48
Ilustración 31. Conexión Access Point WIFI, web server Dispositivo de medición 49
Ilustración 32. Web dispositivo alerta 50
Ilustración 33. Garrafa, 24cm 50
Ilustración 34. Garrafa y Thingspeak con 5cm de líquido 50
Ilustración 35. Aviso de Cambio de Garrafa 51

ANEXOS

Hojas de datos:

Display LCD 1602A

5.Electrical Characteristics

Item	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage For Logic	V_{DD}-V_{SS}	–	4.5	–	5.5	V
Supply Voltage For LCD	V_{DD}-V_0	Ta=0℃	–	–	4.2	V
		Ta=25℃	–	3.8	–	V
		Ta=50℃	3.6	–	–	V
Input High Volt.	V_{IH}	–	2.2	–	V_{DD}	V
Input Low Volt.	V_{IL}	–	–	–	0.6	V
Output High Volt.	V_{OH}	–	2.4	–	–	V
Output Low Volt.	V_{OL}	–	–	–	0.4	V
Supply Current	I_{DD}	V_{DD}=5V	–	1.2	–	mA

2N7000

FAIRCHILD SEMICONDUCTOR™

November 1995

2N7000 / 2N7002 / NDS7002A
N-Channel Enhancement Mode Field Effect Transistor

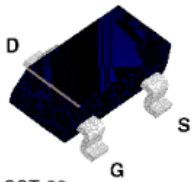
General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged, reliable, and fast switching performance. They can be used in most applications requiring up to 400mA DC and can deliver pulsed currents up to 2A. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

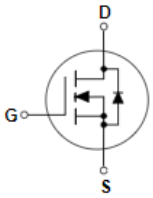
Features

- High density cell design for low $R_{DS(ON)}$.
- Voltage controlled small signal switch.
- Rugged and reliable.
- High saturation current capability.

TO-92



SOT-23



Absolute Maximum Ratings T_A = 25°C unless otherwise noted

Symbol	Parameter	2N7000	2N7002	NDS7002A	Units
V_{DSS}	Drain-Source Voltage	60			V
V_{DGR}	Drain-Gate Voltage (R_{GS} ≤ 1 MΩ)	60			V
V_{GSS}	Gate-Source Voltage - Continuous	±20			V
	- Non Repetitive (tp < 50µs)	±40			
I_D	Maximum Drain Current - Continuous	200	115	280	mA
	- Pulsed	500	800	1500	
P_D	Maximum Power Dissipation	400	200	300	mW
	Derated above 25°C	3.2	1.6	2.4	mW/°C
T_J, T_{STG}	Operating and Storage Temperature Range	-55 to 150		-65 to 150	°C
T_L	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 Seconds	300			°C
THERMAL CHARACTERISTICS					
$R_{\theta JA}$	Thermal Resistance, Junction-to-Ambient	312.5	625	417	°C/W

© 1997 Fairchild Semiconductor Corporation

2N7000.SAM Rev. A1