Universitat Oberta de Catalunya uoc.edu

Diseño de un sistema de monitorización remota de un depósito de agua mediante LoRa.

MEMORIA TFM.

Rubén Adrián de la Cámara
Master en Ingeniería de Telecomunicaciones.
Sistemas de Comunicaciones.

Tutor: Raul Parada Medina
PRA: Carlos Monzo Sánchez
12/06/2019

Ficha del trabajo final

Título del trabajo:	*Diseño de un sistema de monitorización remota de un depósito de agua mediante LoRa.*
Nombre del autor:	*Rubén Adrián de la Cámara*
Nombre del consultor/a:	*Raul Parada Medina*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Master Universitario en Ingeniería de Telecomunicaciones.*
Área del Trabajo Final:	*Sistemas de Comunicaciones*
Idioma del trabajo:	*Español*
Palabras clave	*Depósito, monotorizacion, Arduino, LoRa, LoraWAN, GSM/GPRS*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En este trabajo se propone el diseño de un sistema que permite monitorizar, automatizar procesos y transmitir datos de un depósito de agua para consumo humano, típico de pequeñas poblaciones. La finalidad es poder tener un mayor control sobre el agua que se consume en un municipio, donde normalmente solo se recurre a controles periódicos de la calidad, que pueden ser semestrales e incluso anuales.

Por un lado, se pretende poder conocer con mayor frecuencia parámetros como el nivel del depósito y el pH del agua, sin necesidad de estar en las instalaciones. De este modo se logra una monitorización remota que facilita en gran medida el acceso a la información.

Por otro lado, se recurre al entorno Arduino como sistema de monitorización y automatización y se emplea la tecnología LoRa para poder tener acceso a los datos

de forma remota. Para ello, se recurre a una metodología en cascada, donde partiendo de la contextualización del proyecto y del estudio de las tecnologías necesarias, se diseña el sistema, se construye el prototipo y se verifica su funcionamiento.

Finalmente, una vez alcanzados los objetivos planteados inicialmente, se implementan mejoras como emplear LoRaWAN para subir los datos a TTN y añadir comunicaciones GSM/GPRS que permitan enviar mensajes de alarma, dando lugar a un sistema más completo que el inicialmente planteado.

Abstract (in English, 250 words or less):

In this work, we propose the design of a system that allows monitoring, automate processing and transmiting data from a water tank for human consumption, typical of small populations. The purpose is to be able to have greater control over the water consumed in a municipality, where normally only recurrent quality controls are used, which can be semi-annual or even annual.

On the one hand, it is intended to be able to know more frequently parameters such as the level of the tank and the pH of the water, without needing to be in the facilities. In this way, remote monitoring is achieved, which greatly facilitates access to information.

On the other hand, the Arduino environment is used as a monitoring and automation system and the LoRa technology is used to access the data remotely. Then, a cascade methodology is used, that is, starting from the contextualization of the project and the study of the necessary technologies, the system is designed, the prototype is built and its operation verified.

Finally, once the initial objectives have been achieved, some improvements are implemented, such as using LoRaWAN to upload the data to TTN and add GSM / GPRS communications that allow sending alarm messages, giving rise to a more complete system than initially proposed.

Índice

Índice de figuras vii
Índice de tablas x
1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.1.1 Ejemplo de caso de aplicación. 2
1.2 Objetivos del Trabajo. 4
1.3 Enfoque y método seguido. 4
1.4 Planificación del Trabajo 5
1.4.1 Alcance. 6
1.4.2 Hitos. 6
1.4.3 Calendario de trabajo. 7
1.4.4 Tareas y Diagrama de Gantt. 8
1.4.5 Riesgos e incidencias. 11
1.4.6 Recursos. 11
1.5 Breve sumario de productos obtenidos 12
1.6 Breve descripción de los otros capítulos de la memoria 13
2. Estado del arte 14
2.1 Contexto actual. 14
2.1.1 Parámetros de control de calidad del agua potable. 14
2.1.2 Sensores y sistemas de control del agua. 17
2.2 Trabajos relacionados 19
2.3 Resumen del capitulo 24
3. Diseño del sistema 25
3.1 Tecnologías necesarias 25
3.1.1 Entorno Arduino 25
3.1.2 LoRa 28
3.2 Monitorización y automatización. 32
3.2.1 Sensores. 33
3.2.2 Módulo Arduino. 38
3.2.3 Sistema de monitorización y posible automatización. 40
3.3 Comunicaciones LoRa. 41
3.3.1. Módulos LoRa 42
3.3.2 Configuración y pruebas de enlace punto a punto. 47
3.3.3 Pruebas con sensor DHT11 y TTGO. 50
3.4. LoRaWAN y TTN 51

3.4.1 The Things Network. 51
3.4.2 TTN en este proyecto. 53
3.4.3 Pruebas con TTN. 54
3.5 Resumen del capítulo 60
4. Prototipo y pruebas 61
4.1 Ubicación. 61
4.2 Prototipo inicial. 62
4.3 Prototipo definitivo 66
4.4 Pruebas. 68
4.5 Comentarios sobre los resultados de las pruebas 70
4.6 Presupuesto. 71
4.7 Resumen del capítulo. 72
5. Mejora del sistema: alarmas SMS 73
5.1 Contexto y requisitos 73
5.2 Implementación en el sistema y pruebas 74
5.3 Pruebas con el prototipo en el depósito 76
5.4 Resumen del capítulo. 78
6. Conclusiones y líneas futuras 79
Glosario 83
Bibliografía 84
Anexos 90

Índice de figuras

FIGURA 1. ESQUEMA DE ABASTECIMIENTO DE AGUA 3
FIGURA 2. BLOQUES DE SISTEMA 5
FIGURA 3. DIAGRAMA DE GANTT 10
FIGURA 4. KITS PORTÁTILES DE MEDICIÓN. [5] 15
FIGURA 5. ESQUEMA DE MONITORIZACIÓN EN GRANDES REDES DE AGUA [9] 17
FIGURA 6. SENSOR CLORO Y PH [41] 17
FIGURA 7. SENSOR TURBIDEZ DEL AGUA [42] 18
FIGURA 8. SISTEMAS COMPLEJOS DE MONITORIZACION [43] 18
FIGURA 9. ENTORNO ARDUINO [39] 26
FIGURA 10. MÓDULOS ARDUINO [39] 26
FIGURA 11. INTERFAZ DE PROGRAMACIÓN DE ARDUINO 27
FIGURA 12. RELACIÓN LORA-LORAWAN [26] 29
FIGURA 13. ARQUITECTURA LORAWAN [26] 30
FIGURA 14. SENSOR HC-SR04 33
FIGURA 15. ESQUEMA FUNCIONAMIENTO DEL HC-SR04 34
FIGURA 16. CONEXIÓN SENSOR DISTANCIA 34
FIGURA 17. SENSOR DE PH [48] 35
FIGURA 18. ESQUEMA CONEXIÓN SENSOR DE PH 36
FIGURA 19. PUEBAS SENSOR DE PH 38
FIGURA 20. ARDUINO UNO R3 [39] 39
FIGURA 21. PROTOTIPO CON ATMEGA328 [39] 40
FIGURA 22. EJEMPLOS DE RELÉ Y CONTACTOR [39] 40
FIGURA 23. ESQUEMA SISTEMA DE MONITORIZACIÓN Y AUTOMATIZACIÓN 41
FIGURA 24. MÓDULO RYL869 PINOUT [44] 42
FIGURA 25. MÓDULO E32-868T20D [45] 43

FIGURA 26. MÓDULO DRAGINO LORA SHIELD [46] 44
FIGURA 27. TTGO LORA32 V2.1_1.6 [47] 45
FIGURA 28. PINOUT DEL MÓDULO TTGO LORA32 V2.1_1.6 [47] 46
FIGURA 29. PRUEBAS ENLACE PUNTO A PUNTO 48
FIGURA 30. PRUEBAS CON TTGO Y DTH11 50
FIGURA 31.COBERTURA TTN EN ESPAÑA [49] 51
FIGURA 32. ARQUITECTURA DE RED LORAWAN [50] 52
FIGURA 33. GATEWAY DRAGINO LG01 [51] 53
FIGURA 34. APLICACIÓN EN TTN 55
FIGURA 35. DISPOSITIVO FINAL EN TTN 55
FIGURA 36. GATEWAY REGISTRADO EN TTN 56
FIGURA 37. EJEMPLO DE PAQUETE RECIBIDO 57
FIGURA 38. SIMULACION DE PAQUETES RECIBIDOS 57
FIGURA 39. PRUEBAS CON TTGO Y TTN 58
FIGURA 40. TTN - INTEGRACION DE ALMACENAJE DE DATOS 59
FIGURA 41. TTN - CONSULTA A LA BASE DE DATOS 59
FIGURA 42. DEPÓSITO DE AGUA MUNICIPAL 61
FIGURA 43. VISTA INTERIOR DEL DEPÓSITO. 62
FIGURA 44. PROTOTIPO INICIAL 63
FIGURA 45. PRUEBAS TTGO EMISOR 64
FIGURA 46. PRUEBA_UNO TTGO RECEPTOR 65
FIGURA 47. PRUEBA_DOS TTGO RECEPTOR. 65
FIGURA 48. PROTOTIPO EN EL DEPÓSITO 66
FIGURA 49. ARQUITECTURA DEL PROTOTIPO 67
FIGURA 50. TTN - TRÁFICO EN EL GATEWAY 68
FIGURA 51. TTN - DATOS RECIBIDOS POR LA APLICACIÓN 69
FIGURA 52. TTN - PAYLOAD Y CARACTERÍSTICAS DE UN PAQUETE 69
FIGURA 53. TARJETA SIM900 GSM/GPRS [39] 74

FIGURA 54. PROTOTIPO CON TARJETA GSM INCLUIDA 75
FIGURA 55. EJEMPLO DE SMS RECIBIDOS 76
FIGURA 56. TTN - VACIADO DEL DEPÓSITO 77
FIGURA 57. SMS DE ALARMA RECIBIDOS 77
FIGURA 58. TTN - LLENADO DEL DEPÓSITO 78
FIGURA 59. CURVA DE CALIBRACION DEL SENSOR DE PH 93
FIGURA 60. CONSOLA EN TTN 108
FIGURA 61. REGISTRO DE UNA APLICACIÓN EN TTN 108
FIGURA 62. VISTA PREVIA DE LA APLICACIÓN 109
FIGURA 63. REGISTRO DE UN DISPOSITIVO 110
FIGURA 64. VISTA PREVIA DEL DISPOSITIVO REGISTRADO 110
FIGURA 65. REGISTRO DE UN GATEWAY 111
FIGURA 66. VISTA PREVIA DEL GATEWAY 112
FIGURA 67. DRAGINO GAETWAY CONFIGURACION ACCESO A INTERNET 112
FIGURA 68. DRAGINO GATEWAY CONFIGURACIÓN LORAWAN 113
FIGURA 69. DRAGINO GATEWAY CONFIGURACIÓN LORA 113
FIGURA 70. DRAGINO GATEWAY SERVER IOT 114
FIGURA 71. TARJETA SIM 900 GSM/GPRS 128
FIGURA 72. VERIFICACIÓN TARJETA SIM900 131
FIGURA 73. SIMULACIONES CON COMANDOS AT 132
FIGURA 74. COMANDOS AT PRINCIPALES 133
FIGURA 75. PRUEBAS CON SENSOR DHT11 Y ENVÍO DE SMS 137
FIGURA 76. SENSOR DE NIVEL EN EL PROTOTIP 144
FIGURA 77. SENDOR DE PH EN EL PROTOTIPO 144
FIGURA 78. ARDUINO, GSM900 Y EMISORES LORA 145

Índice de tablas.

TABLA 1. HITOS 7
TABLA 2. DISTRIBUCIÓN DE HORAS MENSUAL 7
TABLA 3. FRECUENCIA DE MUESTREO SEGÚN LA OMS [5] 16
TABLA 4. DOSIFICACIÓN DE LEJÍA [6] 16
TABLA 5. COMPARATIVA DE SISTEMAS 23
TABLA 6. PARÁMETROS LORA EUROPA Y USA [27] 31
TABLA 7. CARACTERÍSTICAS SENSOR PH 36
TABLA 8. TENSIONES EN LA SALIDA DE LA SONDA DE PH 37
TABLA 9. CARACTERÍSTICAS DEL MÓDULO RYL869 [44] 42
TABLA 10. CARACTERÍSTICAS DEL MODULO E32-868T201D [45] 43
TABLA 11. CARACTERÍSTICAS MÓDULO TTGO LORA V2.1_1.6 45
TABLA 12. RELACIÓN DISTANCIA - POTENCIA RECIBIDA 49
TABLA 13. PRESUPUESTO 71
TABLA 14. CONEXIONES ENTRE DISPOSITIVOS 143

1. Introducción

Una vez finalizadas las asignaturas correspondientes al Master en Ingeniería de Telecomunicación, el último paso para obtener la titulación consiste en desarrollar un proyecto que ponga de manifiesto las competencias adquiridas. De este modo, se persigue que el alumno demuestre sus capacidades de investigación y análisis de información, así como los conocimientos adquiridos durante los distintos semestres. Es por tanto, el momento de demostrar la capacidad de resolver un problema dentro de un ámbito de conocimiento o aportar alguna mejora, análisis o innovación, sobre algún objeto de dicho ámbito. En este caso, el presente documento detalla el proceso de creación de un sistema de automatización y monitorización remota de parámetros relativos al agua potable almacenada en un depósito. Así, se pretende exponer el conjunto de elementos hardware y software que serán necesarios, para monitorizar, automatizar y acceder a ciertos datos de forma remota empleando el entrono Arduino y la tecnología LoRa.

1.1 Contexto y justificación del Trabajo.

Sin duda, la calidad del agua para el consumo humano es un factor determinante en la salud, al igual que contar con los suficientes recursos o, al menos, conocer su disponibilidad. Por ello, en este trabajo se plantean una serie de iniciativas que permitan tener un mayor control sobre algunos de los parámetros de la calidad y sobre la disponibilidad que hay en un depósito de agua potable para el suministro municipal.

En primer lugar, conocer el nivel del depósito de agua que abaste un municipio puede parecer que no es crítico, pero hay que darse cuenta de que cualquier incidencia que se descubre tarde puede dar lugar a una suspensión del abastecimiento, con lo que esto conlleva. El ejemplo más contundente es una fuga en las tuberías de abastecimiento que se descubre tarde, donde se produce una disminución drástica del agua almacenada en el depósito de forma repentina. En este caso, si se dispone de un mecanismo de alarma, se puede recurrir a revisar la red de suministro con mayor

antelación para localizar la fuga y evitar males mayores. Además, donde ahora se emplea una simple boya que activa y desactiva el mecanismo de bombeo que trae el agua al depósito en función del nivel, se puede evolucionar a un sistema de monitorización del nivel que además de automatizar la puesta en marcha del bombeo permita transmitir la información a un servidor y emplear los datos para su análisis y procesado.

En segundo lugar, los parámetros de calidad del agua son bastantes y difíciles de medir, donde normalmente se requieren laboratorios para poder realizarlos. Como se puede ver en [1] la legislación es bastante clara y precisa, pero no es exigente en cuanto a la frecuencia de toma de muestras. A modo de ejemplo en [2] se puede tener acceso a casi la totalidad de los abastecimientos de agua en España, donde pueden verse los análisis, tratamientos y periodicidad de los mismos. En este proyecto no se trata de entrar en detalle en los distintos parámetros, sino simplemente aplicar alguno de ellos como es el potencial de Hidrógeno (pH) del agua. Por tanto, se trata de poder medir este parámetro con mayor regularidad que la que se propone desde la administración y, en su caso, actuar sobre mecanismos que permitan su corrección.

Finalmente, dada la distancia a la que normalmente se encuentran los depósitos de agua respecto a los núcleos urbanos, se trata de poder tener conocimiento de los parámetros mencionados sin necesidad de desplazar operarios, más aun cuando se pretende tener una captación de datos con mayor frecuencia. Para ello, se requiere recurrir a las TIC como herramientas que permitan realizar todos los procesos necesarios para esta monitorización y gestión, lo que optimizaría tanto el acceso a los datos como su tratamiento, al igual que la eficiencia y eficacia en la gestión del agua.

1.1.1 Ejemplo de caso de aplicación.

Se considera como caso de aplicación del sistema desarrollado en este proyecto un ejemplo real de sistema de suministro de agua perteneciente a la localidad de Gumiel de Izan, situada en el sur de la provincia de Burgos [3]. Se trata de un municipio de muy baja población con aproximadamente 600 personas residentes de forma continua.

Sin embargo, en determinados periodos esta población se triplica, siendo esto un pequeño hándicap para el suministro de agua, sobre todo en verano.

Por un lado, en la figura 1 se presenta un pequeño esquema del conjunto de unidades que forman el servicio de aguas. Para explicarlo hay que decir que el municipio se abastece de agua que procede de dos manantiales situados a unos 4 Kms del depósito. Estos manantiales están a un nivel superior que el depósito, por lo que el agua puede llegar a este por su propio peso, pero en caso necesario hay un equipo de bombeo que aumenta el caudal si la cantidad de agua almacenada baja de un nivel prefijado. Además, en el depósito existe un rebosadero, de forma que si está lleno no de desborda, dado que el agua viene ininterrumpidamente por su peso. Así, es en el depósito donde se realiza la adecuación del agua para el consumo humano, donde básicamente se emplea la cloración como medio desinfectante. La capacidad del depósito es de unos 50 metros cúbicos.

FIGURA 1. ESQUEMA DE ABASTECIMIENTO DE AGUA

Por otro lado, la aplicación del proyecto a este caso concreto consiste en la construcción del prototipo que permita medir el nivel de agua almacenada con una periodicidad determinada, actuar sobre el equipo de bombeo si es necesario, transmitir datos de nivel y pH empleando LoRa y poder enviar alarmas sobre eventos programados.

En resumen, se propone que una vez diseñado el sistema sea posible emplearlo en un entorno real como el propuesto y comprobar su funcionalidad. Se considera que es una buena forma de prevenir sucesos no deseados como son interrupciones del suministro por desconocer la existencia de grandes fugas o tener un mayor control

sobre el pH. En ambos casos, las alertas pueden ser muy importantes, pero también tener un perfil estadístico de ambos parámetros.

1.2 Objetivos del Trabajo.

El objetivo fundamental del presente trabajo es el **diseño de un sistem**a que permita **monitorizar el nivel de agua** de un depósito, así como **conocer el pH**. Además, se trata de **automatizar procesos** relativos a las condiciones del agua en el depósito y debe ser posible emplear la tecnología **LoRa para tener acceso a estos datos de forma remota**, incluso si hay suficiente tiempo poder emplear GSM para el envío de alarmas mediante SMS. Por ello, los objetivos concretos son:

- Realizar una aproximación a la tecnología LoRa.
- Diseñar el sistema.
- Construir el prototipo y realizar su simulación.
- Probar el prototipo en un contexto real.

1.3 Enfoque y método seguido.

En primer lugar, es necesario aclarar que se trata de un trabajo académico, de modo que el proceso consistirá en el aprendizaje y la aplicación al proyecto de lo aprendido. Es decir, partiendo de que el objetivo fundamental del proyecto está definido, se requiere una investigación continua y su aplicación al diseño del sistema de todas las partes que conformarán el todo. Así, conocer el entorno Arduino y la tecnología LoRa son los pilares de este proyecto, suponiendo la mayor carga de trabajo de estudio y aplicación.

Por un lado, una vez planteado el diagrama de bloques que se presentan en la figura 2, se procederá con la resolución de cada bloque siguiendo lo explicado anteriormente. De este modo, en cada bloque se buscará la información necesaria, se realizará el diseño y simulaciones pertinentes, bien sea mediante software o mediante prototipo, con la finalidad de poder comprobar la funcionalidad del bloque diseñado.

Por otro lado, se afronta el trabajo empleando una metodología en cascada donde resolviendo los bloques debe lograrse un prototipo funcional disponible para pruebas. Por tanto, a lo largo del trabajo se profundiza en cada uno de los bloques mediante el estudio de las tecnologías, el análisis de requisitos y la búsqueda de soluciones que permitan resolver cada bloque y la interconexión entre ellos.

FIGURA 2. BLOQUES DE SISTEMA

Finalmente, para mayor claridad se describe cada bloque:

- **Monitorización**: sensores que se emplearán en el sistema.
- **Control**: tarjeta Arduino que incorpora el microcontrolador para procesado.
- **Comunicaciones**: módulos LoRa y GSM.
- **Actuación**: posible actuación sobre bombas de agua y dosificación de cloro.

Todo esto puede variar ligeramente respecto a lo inicialmente aquí expuesto, según se profundice en cada bloque y se adquieran mayores conocimientos. Aparte, no se incluye un bloque de alimentación, dado que se plantea la posibilidad de alimentar el sistema con una pequeña batería. Aun así, si durante el trabajo se acaba considerando imprescindible se puede diseñar una pequeña fuente de alimentación para el sistema, partiendo de la existencia de una tensión de red de 220 V en la ubicación del depósito.

1.4 Planificación del Trabajo

A continuación se presentan los elementos que se han considerado para realizar la planificación completa del proyecto. Partiendo de los objetos que se consideran como parte del alcance, se proponen los recursos necesarios para lograrlos tanto a nivel de

formación como a nivel de construcción del prototipo, todo ello desglosado en las tareas e hitos necesarios dentro de una planificación temporal coherente con los conocimientos actuales y con la disponibilidad temporal.

1.4.1 Alcance.

Se enumeran a continuación los objetos que forman parte del alcance del proyecto:

- Contexto y justificación del proyecto
- Aproximación a las tecnologías necesarias.
- Diseño del sistema.
- Pruebas y simulaciones mediante prototipos.
- Redacción de la memoria del proyecto.
- Realización de la presentación en video del proyecto.

1.4.2 Hitos.

Los hitos del proyecto corresponden a las entregas parciales (PECs) y a la entrega final de la memoria y de la presentación. Así mismo, se considerarán también como hitos los diferentes borradores correspondientes a cada entregable. También, se añade como hito la fecha de defensa del proyecto, tabla 1.Por otro lado, estos hitos son la visión global de todo el trabajo, los cuales se desglosan en las diferentes tareas que se presentan en el diagrama de Gantt del siguiente apartado, que permite ver la EDT del proyecto.

Grupo	Hito	Fecha
PEC 1	Borrador PEC 1	01/03/2019
	Entrega PEC 1	**04/03/2019**
PEC 2	Borrador PEC 2	15/03/2019
	Entrega PEC 2	**18/03/2019**
PEC 3	Borrador PEC 3	15/05/2019
	Entrega PEC 3	**19/05/2019**
PEC 4	Borrador PEC 4	10/06/2019
	Entrega PEC 4	**13/06/2019**
PEC 5	Borrador PEC 5	21/06/2019
	Entrega Memoria y presentación	**23/06/2019**
Debate virtual.	**Respuestas debate virtual.**	**28/06/2019**

TABLA 1. HITOS

1.4.3 Calendario de trabajo.

El TFM tiene una carga de 12 créditos, de modo que se trata de repartir 300 horas (12 créditos x 25 horas/crédito) entre 15 semanas que hay desde el comienzo del proyecto hasta su entrega definitiva. Así, se puede considerar una carga semanal de 20 horas, por lo que si se establece una dedicación de dos horas diarias de lunes viernes y diez horas a lo largo del fin de semana, se estaría cumpliendo con la distribución temporal necesaria. En la tabla 2 se muestra la distribución de horas disponibles en función del número de días laborables y de fines de semana con la distribución planteada, donde se aprecia que se dispone de más tiempo que el supuestamente requerido.

Mes	Días de L a V	Horas	S y D	Horas	Total de horas mensuales
Marzo	21	42	10	50	**92**
Abril	22	44	8	40	**84**
Mayo	23	46	8	40	**86**
Junio	20	40	8	40	**80**
Total		172		170	**342**

TABLA 2. DISTRIBUCIÓN DE HORAS MENSUAL

1.4.4 Tareas y Diagrama de Gantt.

En este punto se procede a definir las tareas que formaran parte del trabajo necesario para alcanzar cada uno de los hitos. De este modo, se puede desglosar el proyecto en las siguentes tareas:

1. **PEC 1: Planificación (23/02/2019-04/03/2019).**
    - Concreción del TFG.
    - Definición de hitos.
    - Definición de tareas y diagrama de Gannt.
    - Elaboración del documento PEC1.
    - Revision y modificaciones.
    - Entrega del plan de proyecto.

2. **PEC 2. Estado del arte. (05/03/2019-18/03/2019).**
    - Revisión feedback y modificaciones.
    - Busqueda de información relativa a casos similares.
    - Estudio de las tecnologías necesarias en este proyecto.
    - Elaboración del documento PEC 2.
    - Revisión y modificaciones.
    - Entrega PEC 2.

3. **PEC 3. (19/03/2019 - 19/05/2019)**
    - Revisión feedback y modificaciones.
    - Monitorización de parámetros.
    - Automatización de procesos.
    - Simulaciones del sistema monitorización y auromatización..
    - Comunicaciones del sistema.
    - Prototipo experimental.
    - Pruebas del sistema completo.
    - Documento PEC3.
    - Revisión e imprevistos.
    - Entrega PEC3.

4. **PEC 4. Finalización del proyecto. (20/05/2017- 13/06/2017).**
    - Revisión feedback y modificaciones.
    - Mejoras.
    - Conclusiones y lieneas futuras de trabajo.
    - Colchon temporal: imprevistos, mejoras y cambios.
    - Documento PEC4.
    - Revisión e imprevistos.
    - Entrega PEC4.

5. **PEC 5. Entrega memoria y presentación. (14/06/2019-23/06/2019)**
    - Revisión feedback y modificaciones.
    - Finalizar la memoria.
    - Elaborar la presentación.
    - Revisión de memoria y presentación.
    - Entrega del proyecto

6. **Debate virtual (defensa del proyecto). (28/06/2018).**

Se ha tratado de ser escueto en la definición de las tareas para no alargar excesivamente el documento. Es obvio que todas estas tareas se subdividen en subtareas que no se han incluido en el diagrama de Gannt que puede verse en la figura 3, para mayor claridad. Por otro lado, hay que tener en cuenta que la planificación da lugar una presunción temporal donde la mayor carga de trabajo corresponde a la PEC 3, dado que es el marco temporal donde se realiza el diseño y pruebas del sistema planteado en este proyecto. Concluyendo, se trata de una planificación bastante ajustada a la carga lectiva, de modo que se deberían poder cumplir los objetivos básicos del proyecto.

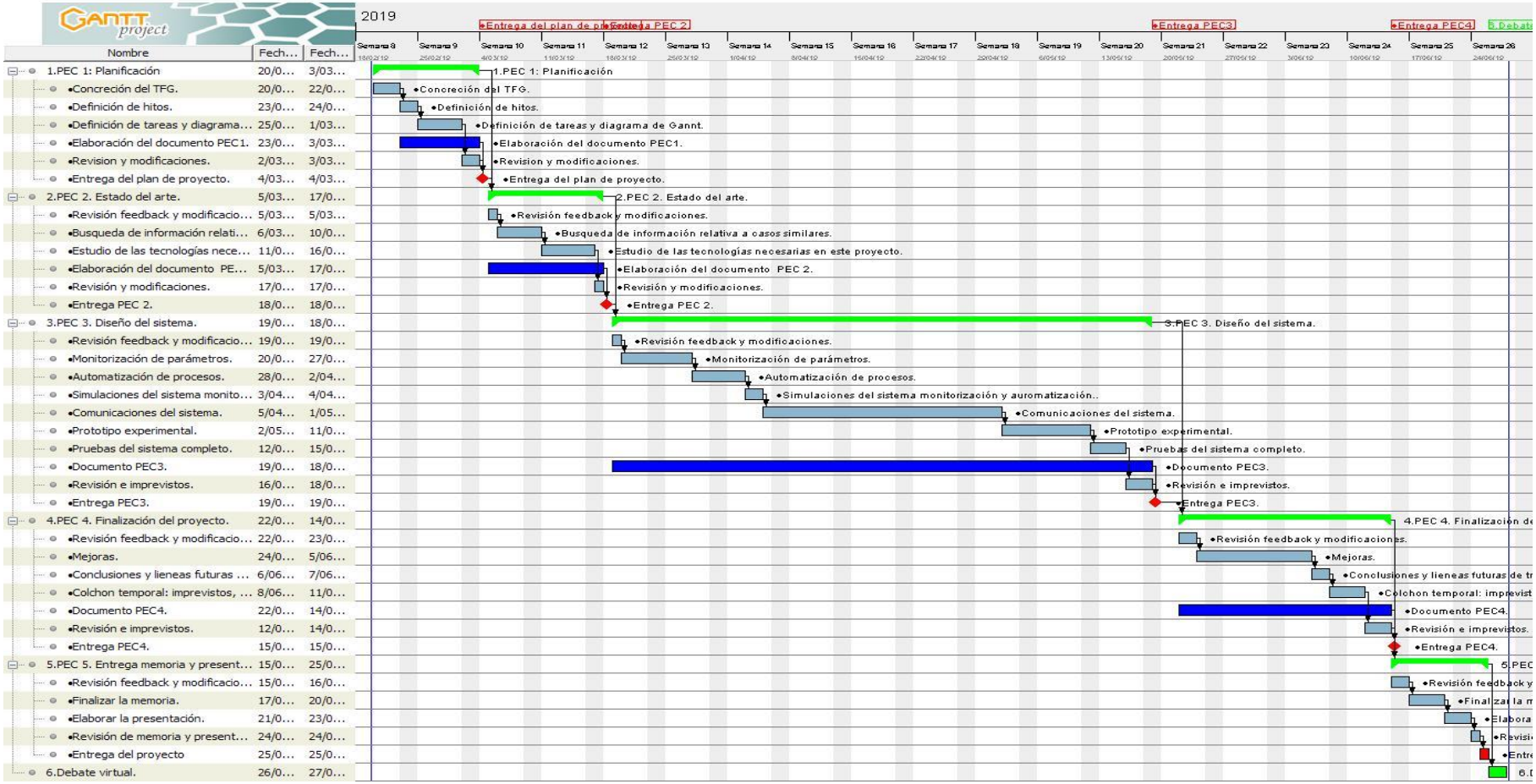


FIGURA 3. DIAGRAMA DE GANTT

1.4.5 Riesgos e incidencias.

En este punto es importante tratar de advertir sobre una serie de riesgos que pueden acontecer durante la elaboración del proyecto, con la finalidad de poder mitigar sus efectos en caso de que se materialicen. De este modo se exponen los que a priori pueden presentarse:

1. Desfases temporales respecto a lo planificado: **probabilidad media, impacto medio**. En este caso, se plantea como contramedida para evitarlo tratar de aprovechar al máximo los días festivos que existen a lo largo del proyecto. Es decir, tratar de ir siempre por delante de lo planificado en cuanto a la finalización de las diferentes tareas.
2. Problemas con el software y el hardware: **probabilidad media, impacto alto.** Como contingencia se empleará un segundo equipo informático para realizar las tareas que no puedan lograrse con el equipo principal. El primer equipo emplea Windows y el segundo Linux.
3. Pérdida de datos o documentación del proyecto: **probabilidad baja, impacto alto**. Para evitar pérdidas de información se realizarán copias diarias del trabajo realizado en una unidad de memoria externa y en la nube.

Es posible que surjan otros incidencias no contempladas, por ello es importante respetar el calendario de tareas y, si es posible, tratar de ir por delante del mismo, para aumentar el tiempo de disponible posibilitando la corrección de cualquier imprevisto o problema que pueda suceder.

1.4.6 Recursos.

Para la elaboración de este proyecto se requirieren una serie de recursos que permitan tanto elaborar la memoria como realizar las tareas de investigación y el diseño y prototipado del sistema. Entre ellos se pueden enumerar los siguientes:

- Pc o portátil con conexión a Internet.
- Software de edición, por ejemplo paquete MS Office.
- Disco duro externo o unidad USB para copias de seguridad.
- Tarjetas, sensores y actuadores (podrían servir LEDs) del entorno Arduino.
- IDE Arduino

- Módulos de comunicaciones LoRa.
- Fuente de alimentación regulable y/o pilas y baterías.
- Resistencias, condensadores, diodos y pequeños cables.

Partiendo de estos elementos y una vez conocido y estudiado tanto el entorno Arduino como LoRa, se puede definir con precisión el tipo de tarjetas o módulos que serán necesarios para elaborar el prototipo. Lo mismo sucede con los sensores específicos del sistema, ya que para simulaciones se pueden sustituir por cualquier otro tipo de entrada y una vez se tenga clara su disponibilidad se pueden adquirir y realizar el prototipo.

1.5 Breve sumario de productos obtenidos

Dado que el objetivo fundamental de este proyecto es el diseño del sistema y la construcción y pruebas de un prototipo para verificar su funcionalidad, se trata pues de obtener los siguientes productos:

- **Prototipo hardware** del sistema perfectamente operativo.
- **Software** que implemente las funciones elementales que demuestren el correcto funcionamiento del sistema.
- **Memoria** del proyecto, donde se expone detalladamente todo el proceso de diseño y su argumentación.
- **Video del emplazamiento de las pruebas realizadas,** para mostrar el contexto de aplicación de la solución diseñada mediante el prototipo construido.
- **Presentación** del proyecto en Power Point o similar, para exponer las ideas y recursos esenciales incluidos en el diseño.

Por otro lado, debido la extensión de los anexos, se incluyen todos ellos en un documento complementario a esta memoria, con la finalidad de incluir todo lo necesario para comprender que se ha hecho y como, pero fuera del documento principal.

1.6 Breve descripción de los otros capítulos de la memoria

Esta memoria se divide en 6 capítulos:

Capítulo 1: se contextualiza el problema y se propone la solución, se detalla todo el proceso de planificación especificando hitos y tareas necesarias junto con el calendario. Además, se presentan los riesgos y medidas preventivas, finalizando con la descripción de resultados obtenidos.

Capítulo 2: se realiza una aproximación al estado actual del ámbito del tema del proyecto: el agua potable y su control. También, se presentan ejemplos de elementos de control dentro de este ámbito así como trabajos académicos que abordan este campo.

Capítulo 3: se trata del capítulo principal del proyecto, donde se expone todo el proceso de diseño del sistema. Se parte de la exposición de las tecnologías que se emplean en el diseño de la solución (Arduino y LoRa), continuando con un pequeño análisis de los distintos tipos de componentes hardware empleados. Se proponen los dispositivos que se emplean y las pruebas necesarias para verificar su funcionamiento, tanto en monitorización como en comunicaciones. Finalmente, se aborda la aplicación de LoRaWAN en el proyecto empleando un gateway y The Things Network (TTN).

Capítulo 4: se trata el proceso de construcción del prototipo, sus evoluciones y las pruebas realizadas para verificar su funcionamiento, incluyendo la instalación del hardware en el depósito de agua para demostrar que realmente se puede implementar en un contexto real.

Capítulo 5: se expone una mejora del sistema que consiste en enviar mensajes SMS de alarma empleando la red GSM/GPRS. Para ello, se explica todo el proceso seguido en el diseño, la adaptación al prototipo construido en el capítulo 4 y las pruebas necesarias para validar su operatividad.

Capítulo 6: se proponen los resultados y las conclusiones obtenidas a la finalización del proyecto, así como las opciones futuras con las que se puede mejorar el diseño.

2. Estado del arte

En este capítulo se realiza la presentación y análisis del estado del arte en el ámbito relativo al proyecto, así como las tecnologías y técnicas necesarias para el diseño del sistema planteado. Básicamente, se expone la situación actual en el control del agua de consumo, técnicas, sistemas y proyectos similares y conceptos introductorios a Arduino y LoRa como ejes vertebradores del proyecto.

2.1 Contexto actual.

En el capítulo anterior ya se planteó la motivación y una aproximación a lo que se pretende realizar en este proyecto, pero en este punto se expone con más detalle la situación actual en el ámbito del control del agua para consumo humano, así como proyectos relacionados con este ámbito y otros ejemplos que emplean Arduino y/o LoRa.

2.1.1 Parámetros de control de calidad del agua potable.

En primer lugar, como ya se comentó, en [1] y [2] se pueden ver tanto la legislación que el agua potable debe cumplir como el estado de las distintas captaciones de agua en la mayoría de España. Por tanto, ahora se trata de conocer con mayor precisión como está este ámbito. Así, haciendo referencia a [1], en [4] se expone que la vigilancia de la calidad del agua para el abastecimiento a la población comienza en embalses, ríos y pozos, continúa durante su tratamiento en las estaciones de tratamiento de agua potable (ETAP) y a través de su paso por la red de distribución hasta que llega al consumidor. En todos estos puntos se recoge muestras de agua que, posteriormente, se analizarán en laboratorio. Con las técnicas adecuadas, los técnicos analizarán aquellos parámetros necesarios para conocer si el agua es apta para consumo humano. Por ejemplo, los parámetros a controlar para el grifo del consumidor son, al menos: olor, sabor, color, turbidez, conductividad, pH, amonio, bacterias coliformes, E. Coli, cobre, cromo, níquel, hierro, plomo, cloro libre residual y cloro combinado residual. La calidad del agua se determina comparando las características físicas y químicas de una muestra de

agua con unas directrices de calidad del agua o estándares. En el caso del agua potable, estas normas se establecen para asegurar el suministro de agua saludable para el consumo humano y, de este modo, proteger la salud de las personas. Estas normas se basan en unos niveles de toxicidad aceptables tanto para los personas como para los organismos acuáticos.

Por otro lado, los controles de calidad pueden realizarse in situ o bien tomando muestras que se llevan a un laboratorio para su estudio. En la figura 3 se muestran 2 kits para realizar mediciones en el lugar de la toma de muestras, que permiten medir parámetros como:

- Para la **temperatura**, ambos kits usan un termómetro.
- Para la **conductividad**, ambos kits usan un medidor específico.
- Para el **pH**, el kit Wagtech utiliza un medidor electrónico específico, mientras el kit Del Agua usa un medidor por colorimetría (comparador de cloro / pH)
- Ambos kits emplean un tubo de **turbidez** (también llamado nefelómetro).
- Para medir la cantidad de **Cloro residual**, ambos kits usan un medidor por colorimetría.
- Para los **Coliformes**, ambos kits tienen un sistema similar.

FIGURA 4. KITS PORTÁTILES DE MEDICIÓN. [5]

Además, en [5] se expone que la frecuencia de los análisis depende de muchos factores y en particular debe de ser adaptada a los recursos económicos y humanos

disponibles y se presentan en la tabla 3 los estándares de la Organización Mundial de la Salud (OMS).

Menos de 5.000 habitantes	1 muestra mensual
De 5.000 a 100.000 habitantes	1 muestra al mes por cada 5.000 habitantes
Más de 100.000 habitantes	20 muestras al mes más una muestra mensual por cada 10.000 habitantes

TABLA 3. FRECUENCIA DE MUESTREO SEGÚN LA OMS [5]

Finalmente y a modo de tema de estudio y controversia, hay que comentar que existe cierta polémica en cuanto al tratamiento de agua potable con lejía (hipoclorito sódico). Tradicionalmente, se ha empleado la lejía como desinfectante del agua y está perfectamente definida su dosificación [6] como se muestra en la tabla 4, pero hay estudios que demuestran que al emplear este producto se están generando compuestos potencialmente cancerígenos como los trihalometanos [7]. Por ello, en algunas provincias ya se está empezando a emplear como agente desinfectante el dióxido de cloro [8], que prácticamente elimina la presencia de los temidos trihalometanos.

VOLUMEN DE AGUA A TRATAR	CONCENTRACIÓN DE LA LEJÍA A UTILIZAR			
	20 gr. de cloro/litro	40 gr. de cloro/litro	80 gr. de cloro/litro	100 gr de cloro/litro
2 litros	4 gotas	2 gotas	1 gota	1 gota
20 litros	20 gotas	10 gotas	5 gotas	4 gotas
100 litros	5 ml.	2,5 ml.	25 gotas	20 gotas
1.000 litros	50 ml.	25 ml.	15 ml.	10 ml.

TABLA 4. DOSIFICACIÓN DE LEJÍA [6]

2.1.2 Sensores y sistemas de control del agua.

Como ya se ha comentado, el control típico de la calidad del agua se realiza mediante la toma de muestras y su posterior análisis en laboratorio o en algunos caso in situ. También, hay que decir que existen sistemas de monitorización de parámetros y automatización de procesos para corregir las desviaciones de estos parámetros a gran escala, pero se trata de redes destinadas al consumo en grandes ciudades [9], donde se emplean las TIC para gestionar todo el proceso. En el caso de este proyecto esto no sería viable debido a que la inversión sería desproporcionada si se tratara de implementar algo como lo que se presenta en la figura 4.

FIGURA 5. ESQUEMA DE MONITORIZACIÓN EN GRANDES REDES DE AGUA [9]

En primer lugar, si se parte de lo más básico, se pueden presentar determinados equipos y herramientas que permiten realizar mediciones de uno varios parámetros sin tener que recurrir a análisis en laboratorios. Así, existen desde pequeños equipos de medición ya integrados que pueden medir un parámetro determinado o varios y presentar la información en algún tipo de pantalla o sensores que solo pueden medir y deben conectarse a otros equipos electrónicos para poder emplear los datos medidos. Por ejemplo, en la figura 6 se muestra un medidor de Cloro y pH portátil.

FIGURA 6. SENSOR CLORO Y PH [41]

También, se encuentran en el mercado sensores y sondas que deben conectarse a otros equipos electrónicos para poder mostrar los datos o procesarlos. Estos son los que se emplean cuando se realizan diseños de sistemas más complejos, donde se persigue monitorizar mayor cantidad de parámetros y, posteriormente, emplear los datos para actuar sobre los mecanismos de corrección en caso necesario. Así, en la figura 7 se muestra un ejemplo que corresponde a un sensor para medir la turbidez del agua.

FIGURA 7. SENSOR TURBIDEZ DEL AGUA [42]

Finalmente, existen soluciones complejas que consisten en sistemas diseñados para medir una gran variedad de parámetros con la posibilidad de mostrarlos por pantalla y/o enviarlos a servidores web. En la figura 8 se pueden ver un ejemplo donde se muestra una solución que puede monitorizar diversos parámetros mediante las sondas que se pueden ver; pero no muestra los datos sino que emplea la conectividad móvil y/o ZigBee de largo alcance para subir los datos a un servidor.

FIGURA 8. SISTEMAS COMPLEJOS DE MONITORIZACION [43]

2.2 Trabajos relacionados.

Como en cualquier otro ámbito de estudio, la investigación y análisis de trabajos similares resulta imprescindible para entender que se está haciendo en la actualidad dentro del contexto del presente proyecto. Por ello, es necesario documentarse para tener mayor visión general sobre lo que se pretende realizar y lo que ya se ha hecho, de forma que se puedan implementar mejoras y/o justificar soluciones alternativas.

Entonces, si se consideran proyectos similares al que se plantea en este caso, en [10] se propone un trabajo académico de Grado realizado en 2018, que consiste en un sistema de adquisición y transferencia de variables remotas para monitorizar tanques de almacenamiento de agua potable, donde se emplea un PLC (Controlador Lógico Programable) al que se conecta un sensor de nivel y se comunica mediante radio con el destino. En cambio, en [11] un trabajo académico del año 2017 donde se recurre a Arduino y Ethernet para realizar el mismo cometido que en el caso anterior. En ambos casos se mide el nivel de los depósitos de agua y se transmite la información a distancia, pero se emplean tecnologías distintas. También, se puede ver en [12] otro trabajo académico del año 2012 que plantea un diseño de un sistema de control y automatización de una planta potabilizadora, pero en este caso se emplea un Controlador de Automatización Programable (PAC), sin envío de datos fuera de la planta.

Con todo, una vez comprobado que el sistema funciona parecería lógico recurrir a más sensores para poder medir otros parámetros del agua y esto es lo que empresas especializadas suelen realizar. Sin embargo, hay que tener presente que existen muchos tipos de sensores siendo necesario validar su fiabilidad y es aquí donde se puede ver como entran en juego iniciativas como el diseño de plataformas de validación de este tipo de sensores [13]. Aparte, conocer los parámetros de calidad del agua solo es útil para técnicos cualificados que sepan interpretar y tomar decisiones en base a estos datos. Por ello, para automatizar procesos de tratamiento de aguas se requiere que estos técnicos determinen los límites de dichos parámetros, de forma que

los automatismos entren en acción cuando la programación del sistema así lo requiera.

Por otro lado, considerando opciones de transmisión de datos de forma remota, ya se han visto en los ejemplos anteriores que cualquier tecnología puede ser empleada (Ethernet o radio), la diferencia está en el balanceo de coste / utilización. Por ejemplo, en [14] puede verse un trabajo académico de 2016 que propone la utilización de Arduino y un módulo GSM para la monitorización remota de los parámetros de calidad del agua, incluso permitiendo la interacción mediante SMS. Además, en este proyecto se enumeran ejemplos de proyectos que realizan funciones similares:

A. **VonRoll Hydrobox**, un producto comercial de la empresa suiza VonRoll Hydro, que dejó de comercializarse por su baja demanda [15]. Es un dispositivo comercial para el control de calidad del agua con sensores de pH, cloro, turbidez, temperatura, conductividad, y potencial de oxidación/reducción. Los datos medidos por estos sensores se pueden visualizar en la pantalla del dispositivo y son periódicamente enviados al cliente por un método de su elección entre SMS, email y fax. Además, los datos también están disponibles en una plataforma de la cual la empresa es propietaria. Para la transmisión de los datos la empresa ofrece el uso de GSM, GRPS y Wifi. Para su instalación y configuración es necesario la intervención de un técnico de la empresa y todos los cambios que se deseen realizar supondrán un coste extra para el cliente. Al ser un producto comercial desarrollado por una empresa privada, la documentación sobre su desarrollo no está disponible. Se estima que su coste no será inferior a 3 500€.
B. **Low-cost Autonomous Water Quality Monitoring System**, un proyecto desarrollado por la universidad de Melbourne, Australia [16]. Consiste en una red de sensores inalámbricos (WSN) diseñada por los investigadores de la universidad Sains de Malasia utiliza como nodos módulos Xbee que se comunican entre ellos gracias al estándar ZigBee. La información finalmente llega a un módulo GSM que lo envía al servidor central mediante un SMS. No hay referencia de su coste, pero comentan que es muy bajo.
C. **PublicLab: Proyecto Open Water**, una serie de desarrollos en el marco del proyecto Open Water de la comunidad OpenLab, de los cuales

destacan **RIFFLE** y **Water Quality Sensor** [17]. Es importante destacar que RIFFLE no se centra en el diseño de los distintos sensores, sino en que su dispositivo sea compatible con una gran variedad de estos. Entre las ventajas de RIFFLE frente a otros proyectos destacan el precio, su fácil uso y la idea de hágalo usted mismo, que permite a los usuarios replicar y adaptar el sistema a sus necesidades. Almacena las distintas medidas en un archivo de texto de una tarjeta SD, que el usuario a posteriori debe inspeccionar. Esto impide que la monitorización se realice en tiempo real e involucra demasiado al usuario en las tareas de monitoreo, aunque es viable la implementación de comunicaciones. El precio del RIFFLE es de aproximadamente 60€ sin contar el precio de los sensores o distintos módulos que se le quieran añadir.

D. **Mãe d'água -Water's mother**, un proyecto de Rede InfoAmazoniay de la iniciativa Open Water [18]. El dispositivo desarrollado está formado por un Arduino y sensores de pH, conductividad, potencial de oxidación/reducción y temperatura. Los datos recogidos por estos sensores se almacenan en una tarjeta SD. Además, el dispositivo está conectado a un módulo de comunicación 2G GSM que permite el envío de datos y alarmas por SMS y la transmisión de datos por GPRS a una página web para su visualización. Su precio, dependiendo del modelo, varía desde los 40$ a los 80$. A este precio, además de los gastos de envío, hay que sumarle el precio de una batería de litio y una antena, imprescindibles para su funcionamiento.

Aparte, comentar que existen infinidad de proyectos que tratan el tema de la monitorización, automatización y transmisión de datos a distancia, tanto para agua potable como para cualquier otro ámbito. En algunos solo se monitoriza y automatiza, en otros se actúa, en otros se monitoriza y se transmiten datos y no se actúa de forma programada, etc. Así, en [19] y [20] se muestran ejemplos de proyectos que emplean LoRa/LoRaWAN para conseguir sus objetivos de comunicaciones, pero no se automatiza ningún actuador. Para poder tener una visión de las posibles aplicaciones de la tecnología LoRa, basta con ver algunos ejemplos y proyectos que se han desarrollado. Así, ya se comentó que en [19] se propone un sistema de sensorización en un parque natural, donde se quieren conocer parámetros ambientales y, empleando

pasarelas, subir estos datos a servidores en Internet. También, en [20] se propone el diseño de un sistema de monitorización ciudadana empleando LoRa junto con wifi. Además, en [28] se presenta un diseño de la arquitectura para la monitorización remota de sensores (ambiental, posicionamiento GPS y contadores eléctricos) basado en LoRa y su volcado a servidores para el análisis de los datos. En [29] se propone evaluar y documentar las características de los módulos LoRa como dispositivo de comunicación de largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo, proponiéndose además posibles aplicaciones.

Complementando lo anterior, a nivel de proyectos o iniciativas empresariales se considerar algunos ejemplos de empresas de nueva creación o de otras ya existentes que han comenzado a incluir LoRa/LoRaWAN en sus productos. Por ejemplo, en [30] se ofrecen soluciones a medida basadas en LoRA, además de explicar al potencial cliente en qué consiste esta tecnología. En [31] se puede ver una empresa de distribución de sensores que ha comenzado a incluir LoRa en sus productos para dotarlos de mayor conectividad en entornos complicados o para reducir costes de despliegue. También, se pueden encontrar empresas que ofrecen toda la tecnología empaquetada en un kit para conectar cualquier sensor a internet empleando LoRa, como puede verse en [32], al igual que fabricantes de estos kits.

Finalmente, se propone en la tabla 5 una visión comparativa donde se pueden ver los distintos atributos presentes en algunos de los ejemplos dentro el ámbito del control de calidad de agua que se han propuesto. Esto debe permitir tener una visión global de lo que existe y de lo que se pretende conseguir en el presente proyecto. Por otro lado, mientras que en [10] y [11] solo se mide el nivel de los depósitos y se transmite la información al centro de control, en [12] se miden una cantidad considerable de parámetros y se actúa sobre los mismos mediante las técnicas necesarias. En cambio, en [14] se miden algunos parámetros y se transmite la información sin automatizar ningún proceso. Como puede verse, se trata de sistemas que tienen diferentes objetivos, por lo que emplean diferentes tecnologías y aunque pueden parecer similares, son diferentes y distintos de lo que se persigue en el sistema que se plantea en este proyecto.

Sistema	Monitorización	Automatización	Comunicaciones	Coste
VonRoll Hydrobox	pH, cloro, turbidez, temperatura, conductividad, y potencial de oxidación/reducción .	No	GSM, GRPS y Wifi	3500 euros
Low-cost Autonomous Water Quality Monitoring System	No se especifica	No	ZigBee y GSM	Muy bajo, no se especifica.
RIFFLE	No se especifica	No	No, inicialmente	60 $
Mãe d'água	pH, conductividad, potencial de oxidación/reducción y temperatura	No	GSM/GPRS	40-80 $
[10]. Bibliografía[1]	Nivel del tanque	No	Radioenlace	4848 $
[11]. Bibliografía[1]	Nivel del tanque	Si	Ethernet	Sin especificar
[12]. Bibliografía[1]	Presión, Tª, Nivel, Flujo, pH, turbiedad, etc.	Si	No	161.281 $[2]
[13]. Bibliografía[1]	pH, turbidez, Tª conductividad eléctrica	No	GSM/GPRS	Sin especificar
Este proyecto	Nivel del depósito y pH	Si, opcionalmente.	LoRa/LoRaWAN (opcional GSM)	Enlace punto a punto: <90 euros, si incluye LoRaWAN y GSM: 170 euros

TABLA 5. COMPARATIVA DE SISTEMAS

[1] Los títulos de los trabajos son demasiado largos para incluirlos en la tabla, por lo que se referencian con su orden de aparición en la bibliografía.

[2] El precio puede parecer desconcertante, dado que se trata de un proyecto completo para la modernización de una planta potabilizadora de agua, por lo que incluye bombas, tuberías y todo tipo de mecanismos eléctricos y electrónicos para su control, incluso diseño e instalación.

2.3 Resumen del capitulo

Hasta aquí han quedado expuestos los aspectos necesarios para entender el contexto del proyecto. Por un lado, se han presentado conceptos relacionados con los estándares de calidad del agua potable y la existencia de un amplio abanico de sensores y sistemas diseñados para su medición. Por otro lado, se han expuesto trabajos con un contexto similar al que se propone en este proyecto, de forma que pueda comprobarse la posibilidad de implementar mejoras que ayuden a optimizar procesos de esos sistemas. Aun así, se ha podido comprobar que en muchos casos los sistemas solo monitorizan y transmiten datos a un sistema de control ubicado en otro lugar, por ser proyectos mucho más complejos los que incluyen la automatización del tratamiento del agua. Además, en la tabla 5 se han resumido las características más interesantes de los trabajos similares que se han tomado como ejemplos de proyectos ya existentes, con la finalidad de comparar proyectos existentes con el que se plantea en este trabajo. Entonces, el presente trabajo persigue obtener un sistema totalmente escalable que permita la **monitorización y transmisión de datos empleando LoRa**, todo ello a un coste lo más bajo posible. Sin duda, existen soluciones muy complejas que ya se comercializan, al igual que existen proyectos o soluciones más sencillas y es a este último tipo a lo que se pretende llegar a la finalización del diseño del sistema. La parte de automatización, si bien es perfectamente realizable, puede que no se implemente en este trabajo por ser necesario el asesoramiento de técnicos en materia de calidad del agua, además de necesitar los permisos pertinentes.

3. Diseño del sistema

A lo largo de este capítulo se procede a desglosar el diseño del sistema, exponiendo las tecnologías y los dispositivos que se van a emplear tanto en monitorización como en comunicaciones. Por tanto, tras realizar una pequeña introducción de las tecnologías necesarias en el diseño, se procede empleando una metodología en cascada que vaya presentando a cada paso los subsistemas finalizados y completamente operativos.

3.1 Tecnologías necesarias

En el ámbito de un proyecto académico parece lógico recurrir a tecnologías libres como elementos para el diseño de cualquier sistema. Es por ello por lo que a continuación se presenta a modo introductorio algunos conceptos y características de las tecnologías que se van a emplear: Arduino y LoRa.

3.1.1 Entorno Arduino

Arduino consiste en una herramienta para facilitar el uso de la electrónica en proyectos multidisciplinares. Según [21], Arduino es una compañía de fuente abierta y hardware abierto así como un proyecto y comunidad internacional que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan detectar y controlar objetos del mundo real. Así, como se puede ver en la figura 9, el módulo Arduino es el corazón de cualquier sistema formado por sensores y actuadores, de tal forma que, una vez definido un conjunto de estos dispositivos requeridos en cualquier aplicación, se puede programar en la placa un código que permita automatizar cualquier proceso. Como puede verse en [22], gracias a su gran versatilidad y flexibilidad, se pueden aportar soluciones tecnológicas en cualquier ámbito, como puede ser: domótica, industria, robótica, agricultura, IoT (Internet de la cosas), etc.

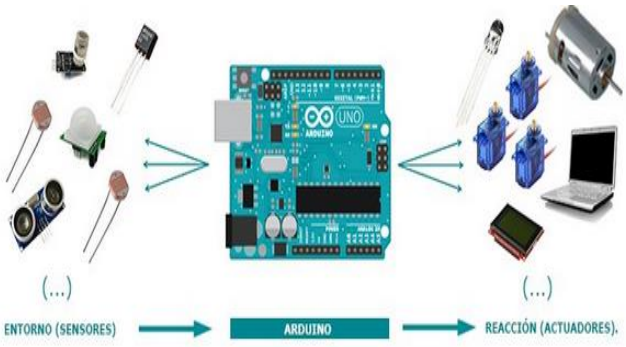


FIGURA 9. ENTORNO ARDUINO [39]

Hardware Arduino.

En este punto, se pretende poner de manifiesto la variedad de placas existentes en la actualidad, sin entrar en detalles técnicos sobre las mismas. Simplemente, mencionar que en función de los requisitos de la aplicación, será más interesante el empleo de una placa u otra. Si bien es cierto que tienen un comportamiento muy similar, existen diferencias relativas a su número de entradas y salidas, tamaño, precio, potencia del microcontrolador, comunicaciones, etc. Por tanto, cada aplicación puede emplear bastantes placas sin ningún problema; pero seguro que hay alguna que es óptima frente al resto, ya que no hay que olvidar el equilibrio entre solución y factor precio. En [23] se pueden ver todas las placas oficiales que existen actualmente. En cambio, también existen placas no oficiales que son muy similares, aunque no están fabricadas bajo la marca Arduino, por lo que llevan otros nombres. En la figura 10, se muestran algunas de las placas oficiales existentes.

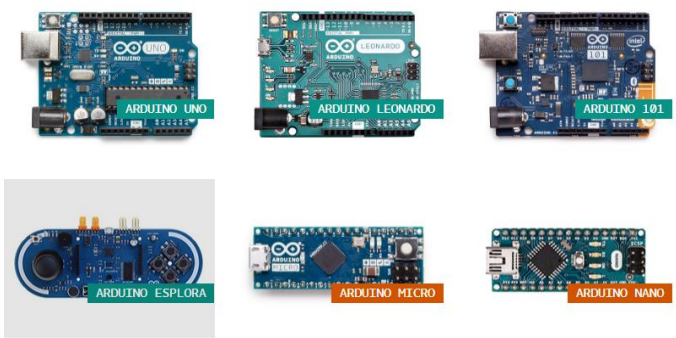


FIGURA 10. MÓDULOS ARDUINO [39]

IDE Arduino.

El entorno de programación de Arduino, del cual se muestra una imagen de su interfaz en la figura 11, emplea un lenguaje similar al lenguaje de programación C, donde se introducen las líneas de comandos que componen lo que se denomina sketch (programa de Arduino). Como puede observarse hay dos bloques principales:

- **void setup**: en este bloque se realiza la configuración del dispositivo y la asignación de pines (entradas y salidas analógicas y digitales).
- **void loop**: se corresponde con el bucle que contiene el programa principal, donde se incluyen todas las sentencias necesarias para obtener la funcionalidad deseada.

Una vez finalizado el programa, se compila y se puede cargar en el módulo Arduino mediante un cable USB.

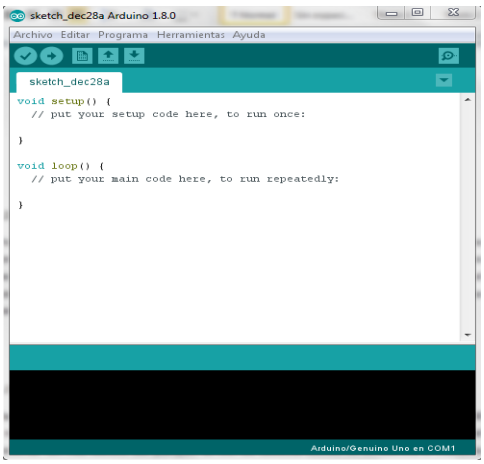


FIGURA 11. INTERFAZ DE PROGRAMACIÓN DE ARDUINO

Por otro lado, en [24] se puede ver que existen otras interfaces de programación que pueden ser empleadas, ya que el único requisito es que permitan la transmisión por puerto serie. De este modo, se pueden emplear otros lenguajes de programación para programar una placa Arduino. Una alternativa muy interesante es emplear Sublime Text que, con las adaptaciones necesarias, permite trabajar con Arduino al igual que el

IDE propio de Arduino, pero con ciertas ventajas que facilitan todo el proceso de codificación.

Aparte, es importante destacar que existen gran cantidad de **librerías creadas por usuarios** y que son de **código abierto**. Así, se pueden emplear en cualquier proyecto con la finalidad de **reutilizar código** y poder simplificar el trabajo de programación. De esta forma, se consigue reducir el número de líneas de código, ya que simplemente se requiere importar una librería y definir variables de este tipo (similar a la programación orientada a objetos), facilitando en gran medida las tareas de programación [22].

Comunidad Arduino.

Un factor del éxito de Arduino ha sido la comunidad que está apoyando este proyecto y que día a día publica nuevo contenido, divulga y responde a las dudas. En Internet hay disponible todo tipo de cursos, tutoriales, herramientas de consulta y proyectos que ayudan a que se pueda usar Arduino con facilidad. Además, en [25] se ve que Arduino playground es un wiki donde todos los usuarios de Arduino pueden contribuir, siendo el lugar donde publicar y compartir código, diagramas de circuitos, tutoriales, trucos, cursos y, sobre todo, el lugar donde buscar cuando existen dudas sobre un problema, una librería adecuada para un proyecto, siendo la base de datos de conocimiento por excelencia de Arduino.

3.1.2 LoRa

A continuación se procede a realizar una aproximación a la tecnología LoRa y como consecuencia al protocolo LoRaWAN, de forma que se comprenda de qué se trata y porque se aplica en este proyecto.

LoRa y LoRaWAN

En primer lugar, LoRa (Long Range) es la capa física o la modulación inalámbrica utilizada para crear enlaces de comunicación de largo alcance. Muchos sistemas inalámbricos heredados utilizan la modulación de cambio de frecuencia (FSK) como

capa física porque es una modulación muy eficiente para reducir la potencia de transmisión. LoRa se basa en la modulación CSS (modulación de espectro de chirrido), que mantiene las mismas características de baja potencia que la modulación FSK pero aumenta significativamente el alcance de comunicación. Es una técnica de modulación de espectro expandido derivada de la tecnología de espectro expandido de chirp (CSS). Los dispositivos LoRa de Semtech y la tecnología de radiofrecuencia inalámbrica (LoRa Technology) permiten crear una plataforma inalámbrica de largo alcance y bajo consumo que se ha convertido en la tecnología de facto para las redes de Internet de las cosas (IoT) en todo el mundo [26].

Por otro lado, también en [26], se puede ver que LoRaWAN es un protocolo de red de área amplia y de baja potencia (LPWAN), basado en la tecnología LoRa, que está diseñado para conectar a Internet de manera inalámbrica dispositivos con baterías en redes regionales, nacionales o globales. Por ello, el protocolo LoRaWAN aprovecha el espectro de radio sin licencia en la banda Industrial, Científica y Médica (ISM). La especificación define los parámetros de la capa física LoRa y el protocolo LoRaWAN y proporciona una interoperabilidad perfecta entre los dispositivos. Por tanto, se entiende que LoRa es la tecnología radio que opera en la capa física de la pila de protocolos, mientras que LoRaWAN se corresponde con los protocolos y la especificación del sistema de comunicaciones, como se muestra en la figura 12. Además, también se puede ver que existen cuatro frecuencias de transmisión (dependiendo de la zona geográfica) y tres clases de dispositivos con distintas características, que se comentarán posteriormente.

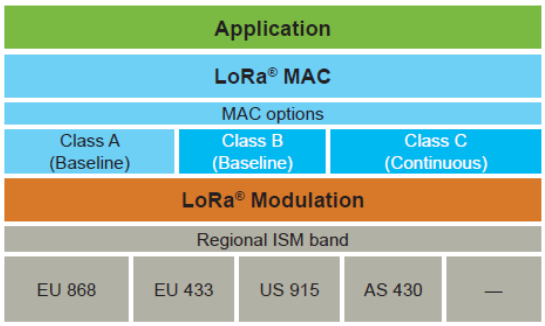


FIGURA 12. RELACIÓN LORA-LORAWAN [26]

En resumen, para entender la arquitectura LoRaWAN y su relación con LoRa, basta con ver la figura 13, donde se presenta el esquema general de despliegue de una red de sensores que empleen esta tecnología y protocolos para su conexión a Internet.

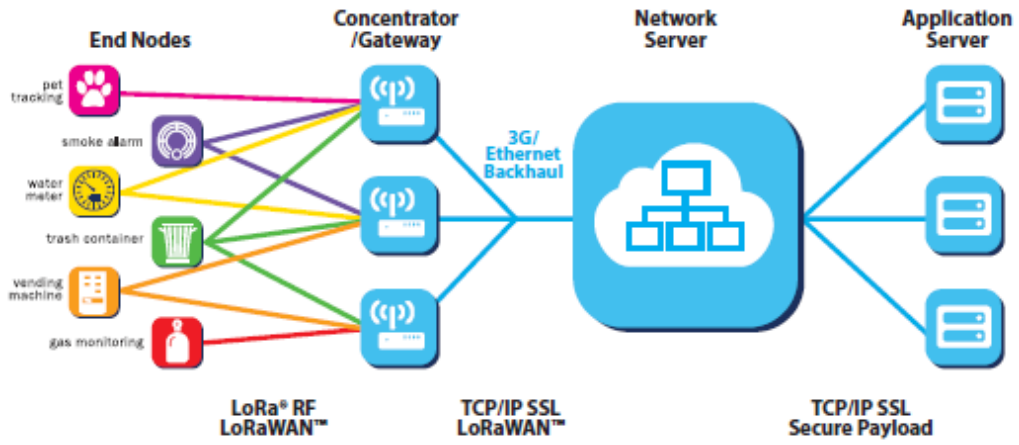


FIGURA 13. ARQUITECTURA LORAWAN [26]

Finalmente, hay que comentar que además de LoRa existen otras tecnologías LPWAN como son Sigfox e Ingenu, que presentan como mayor diferencia que son modelos de negocio propietarios, es decir estas empresas ofrecen todo lo necesario: venta de dispositivos, red de conexión y servicios [27]. En cambio, en LoRA es de Semtech, la empresa propietaria de la capa física (patente sobre los módulos de radio), mientras que la capa de acceso al medio está abierta y es desarrollada por LoRa Alliance, que es una agrupación de entidades que colaboran en el desarrollo de esta tecnología. Así, cualquier empresa interesada puede desarrollar soluciones LPWAN para que sus clientes tengan una infraestructura de red y servicios, por lo que es mucho más atractivo que Sigfox o Ingenu, donde todo está más encapsulado.

Especificaciones LoRa.

Sin tratar de profundizar en exceso en el conocimiento de LoRa, al menos si es necesario presentar algunas de sus características, para poder entender su funcionamiento. Así, partiendo de que LoRa trabaja en la banda ISM, en [27] se puede ver que las principales especificaciones de LoRa se presentan en la tabla 5.

	Europa	América del Norte
Banda de frecuencia	867-869 MHz	902-928 MHz
Canales	10	64 + 8 + 8
Canal banda ancha ascendente	125/250 kHz	125/500 kHz
Canal banda ancha descendente	125 kHz	500 kHz
TX encendido	+14 dBm	+20 dBm típ (+30 dBm permitidos)
TX desconectar	+14 dBm	+27 dBm
SF Up	7-12	7-10
Velocidad de datos	250 bps - 50 kbps	980 bps - 21.9 kbps
Link Budget Up	155 dB	154 dB
Link Budget Dn	155 dB	157 dB

TABLA 6. PARÁMETROS LORA EUROPA Y USA [27]

Entonces, es necesario explicar algunos conceptos de forma que se entiendan sus implicaciones en las comunicaciones LPWAN. Así, por un lado LoRa permite escoger entre 6 factores de ensanchamiento (SF) numerados de 7 a 12, donde cada uno define una relación entre potencia y tasa de transferencia. De este modo, a mayor SF mayor es la sensibilidad del receptor y, por consiguiente, mayor el alcance del enlace, pero la tasa de transferencia es menor y viceversa [26]. Por otro lado, es necesario saber que existen tres clases de dispositivos según las funcionalidades que soportan:

- **Clase A:** este tipo de clase ofrece el mayor ahorro de energía debido a que solo entra en modo escucha después de enviar datos hacia el Gateway, siendo la más eficiente cuando solo se quieren enviar datos.
- **Clase B**: permite la creación de ventanas de recepción sin transmisión previa, por ello consume más energía que la anterior, pero permite mejorar el envío de datos.
- **Clase C**: es la menos eficiente desde el punto de vista energético, debido a que está permanentemente en modo escucha y solo cambia a modo transmisión cuando tiene que transmitir. Es ideal para dispositivos que

necesitan recibir datos con mucha frecuencia (actuadores) presentando muy baja latencia.

Las tres clases pueden coexistir en la misma red y, además, todos los dispositivos deben cumplir con las funcionalidades de Clase A y opcionalmente cumplir las de Clase B y/o Clase C.

Para concluir, aunque la conexión punto a punto entre dos dispositivos LoRa es totalmente posible, dado el interés de esta tecnología en el ámbito IoT, en [26] se puede ver que una red LoRa está formada por tres tipos de dispositivos:

- Dispositivos finales: son los que se conectan los objetos (sensores) a la red LoRa, transmitiendo la información a las pasarelas.
- Pasarelas (Gateway): son las estaciones base LoRa que permiten pasar la información a la red de Internet, recibiendo los datos de múltiples dispositivos finales.
- Servidores: reciben la información de las pasarelas y gestionan y procesan esta información así como la red.

La red LoRa presenta una configuración en estrella, donde cada pasarela recibe de datos de múltiples dispositivos finales en un solo salto y estos datos son enviados a los servidores mediante Ethernet, telefonía móvil, fibra óptica o cualquier otro medio que soporte la pasarela. También, hay que hay que destacar una gran ventaja que consiste en que varias pasarelas pueden recibir datos del mismo dispositivo final, de forma que se puede proporcionar un servicio de posicionamiento basado en LoRa.

3.2 Monitorización y automatización.

En este apartado se detallarán todas las fases del diseño del sistema de monitorización y automatización tanto a nivel hardware como software. Así, se parte de la selección de los sensores adecuados y del módulo Arduino necesario, continuando con la descripción de la conexión de entradas y salidas en la placa Arduino (módulo de automatización) y la codificación necesaria del software, de modo que el sistema realice los procesos para los que está diseñado.

3.2.1 Sensores.

En este punto se describen los sensores que requiere el sistema para captar los datos que el proyecto necesita. Así, se proponen los sensores nivel y pH a emplear y que son compatibles con Arduino, de forma que las adaptaciones a realizar son inexistentes o mínimas.

Sensor nivel del depósito.

Dentro de la variedad de sensores eléctricos y mecánicos que pueden emplearse, se ha escogido el HC-SR04 [33], por su bajo precio (0,65 €) y su facilidad de uso con Arduino. Es un sensor de ultrasonidos para medir distancias y su funcionamiento se basa en el envío de un pulso de alta frecuencia, no audible por el ser humano. Este pulso rebota en los objetos cercanos y es reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia. En la figura 14 se puede ver el sensor y sus características.

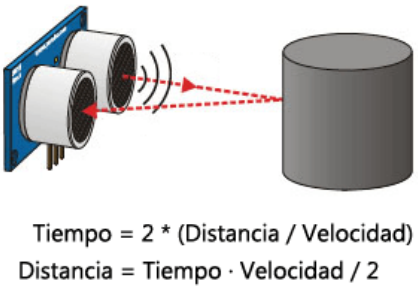
Características	
Alimentación	+5v DC
Frecuencia de trabajo	40 KHz
Consumo (suspendido)	< 2mA
Consumo (trabajando)	15mA
Ángulo efectivo	< 15º
Distancia	2cm a 400cm *
Resolución	0.3 cm

FIGURA 14. SENSOR HC-SR04

El sensor se basa simplemente en medir el tiempo entre el envío y la recepción de un pulso sonoro. Dado que la velocidad del sonido es 343 m/s en condiciones de temperatura 20 ºC, 50% de humedad y presión atmosférica medida a nivel del mar, se tiene que:

$$343\frac{m}{s} \cdot 100\frac{cm}{m} \cdot \frac{1}{1000000}\frac{s}{\mu s} = \frac{1}{29.2}\frac{cm}{\mu s}$$

Por tanto, se puede conocer la distancia a un objeto simplemente teniendo en cuenta el valor anterior, de forma que la distancia viene dada por la mitad del tiempo que ha tardado la señal en regresar al sensor. Es decir, si se ha medido 292 microsegundos, entonces la distancia es 5 cm. Para mejor comprensión se muestra esquema en la figura 15. Si las condiciones de presion temperatura y humedad ambiental son diferentes las medidas no son tan exactas, aunque el error puede despreciarse.



Tiempo = 2 * (Distancia / Velocidad)
Distancia = Tiempo · Velocidad / 2

FIGURA 15. ESQUEMA FUNCIONAMIENTO DEL HC-SR04

Con todo, hay que aclarar que la precisión del sensor no es del todo óptima (0.3cm), pero dada la aplicación donde se va a emplear es una precisión más que suficiente. Se puede considerar que un margen de error de 3 mm en este contexto no supone ningún inconveniente, pero quede claro que hay sensores más precisos a costa de presentar un precio más elevado.

Por otro lado, el esquema de montaje es el que se presente en la figura 16, donde se pueden ver las conexiones eléctricas requeridas. Simplemente, el sensor se alimenta con el pin Vcc = 5 V y GND conectado a tierra del Arduino. El pin Trigger es el que recibe la señal de disparo desde Arduino y el pin Echo es el que envía a Arduino el eco recibido desde el objeto. En base a esta señal rebotada es como se mide la distancia.

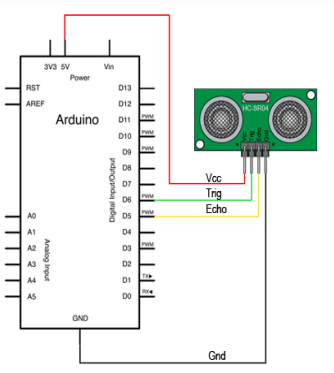


FIGURA 16. CONEXIÓN SENSOR DISTANCIA

En los anexos se incluye el código Arduino empleado para comprobar el correcto funcionamiento del sensor. Una vez montado el circuito y cargado el código en el microcontrolador, las mediciones se pueden ver en el monitor serie del IDE de Arduino, mostrándose las distancias en cms y sin decimales. Aun así, este código no es el adecuado para poder conseguir conocer el nivel de agua en el depósito, por lo que en la parte de diseño del prototipo se realizarán las modificaciones oportunas.

Sensor de pH.

En este caso, el sensor elegido es el que se puede ver en la figura 17. Se trata de un conjunto formado por una sonda y un circuito de adaptación, perfectamente diseñado para poder emplearse en el entorno Arduino. Además, su precio es muy bajo, por lo que para un proyecto académico como el actual es más que suficiente. Es obvio que existen sensores de mayor calidad (profesionales), pero no se considera oportuno adquirir alguno de ellos dados los precios excesivos. El precio de este conjunto de elementos es inferior a 25 euros, dependiendo del vendedor.

FIGURA 17. SENSOR DE PH [48]

Como puede verse está formado por dos elementos:

- Una sonda de medición de pH que se conecta con el circuito de adaptación mediante una conexión BNC.
- Un circuito de adaptación que proporciona una tensión de salida proporcional a la medición de pH y que permite introducir estos datos en el módulo Arduino.

Las principales características que se pueden obtener en la página del vendedor [34], son las que se presentan en la tabla 7.

Tensión de operación	**5 ± 0,2 V**
Corriente de operación	**5-10 mA**
Rango de medición	**0 - 14**
Temperatura de operación	**-10 a 50 ºC**
Tiempo de respuesta	**<= 5 s**
Potencia	**<= 0.5 w**

TABLA 7. CARACTERÍSTICAS SENSOR PH

En cuanto a su conexión con la placa Arduino, tal y como puede verse en la figura 18, es muy sencilla, dado que solo se requiere alimentar el sensor con 5 Vcc y GND y emplear un pin de entrada analógico para que Arduino reciba el valor de tensión que envía la sonda.

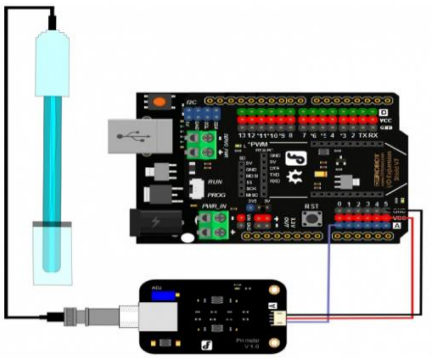


FIGURA 18. ESQUEMA CONEXIÓN SENSOR DE PH

Por otro lado, hay que tener en cuenta que las tensiones a la salida de la sonda, según lo especificado en el datasheet y que se muestran en la tabla 8, son simétricas. Es decir, las tensiones en la entrada del módulo Arduino deben ser adaptadas al rango 0-5V, por ello es necesario emplear el circuito de adaptación, que proporcionará ese rango de tensiones y no el rango simétrico que se ve en la tabla.

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

TABLA 8. TENSIONES EN LA SALIDA DE LA SONDA DE PH

Aparte, según el datasheet, se requiere la calibración de la sonda, para lo que es necesario emplear unas disoluciones con pH conocido, de forma que tras realizar varias medicones y dibujar la gráfica pH-tensión, se tiene la curva de calibración. Mediante esta curva es posible conocer la linealidad de la relación entre el pH y la tensión recibida, de tal forma que en el sketch de Arduino sea posible implementar esta relacion para que cuando se realicen mediciones estas sean reales. En los anexos se adjunta el sketch empleado para verificar el funcionamiento del sensor de pH y las intrucciones de calibración.

Finalmente, se han realizado varias mediciones empleando el modelo que se muestra en la figura 19. Una vez montado el circuito se emplean tres recipientes: el vaso tiene agua del grifo, la primera taza contiene vinagre y la segunda taza contiene de lejía. Las medidas de pH obtenidas son:

- Agua del grifo: pH = 7.3
- Vinagre: pH = 3.4
- Lejía: pH = 12.3

Como puede verse se pone de manifiesto que el sensor de pH detecta este parámetro de forma correcta, dependiendo de la disolución en la que se introduzca.

FIGURA 19. PUEBAS SENSOR DE PH

Si se busca información en Internet acerca del pH de algunas sustancias puede verse que estos valores son muy parecidos a los que se han obtenido, por lo que puede decirse que el sensor de pH es funcional. Además, se han realizado otras dos pruebas consistentes en añadir unas gotas de vinagre y de lejía al agua del grifo para ver como variaba el pH y se ha obtenido lo siguiente:

- Agua + lejía: pH=9.25
- Agua + vinagre: pH= 4.7

Con todo, **en este caso se trataba de comprobar que la sonda de pH funcionaba** y ha quedado demostrado que sin ser un dispositivo de una calidad propia de laboratorio, al menos cumple su función.

3.2.2 Módulo Arduino.

La placa escogida para realizar este proyecto es la versión UNO R3, la cual puede verse en la figura 20 junto con sus principales características. Esto es así, simplemente, por el hecho de que es la placa que se recomienda para principiantes

en el mundo de Arduino, siendo una placa muy versátil y económica (menos de 20 euros).

El cerebro de la placa es el microcontrolador ATmega 328, que es el encargado de ejecutar las instrucciones del sketch en función de las entradas, activando o desactivando las salidas de la placa, incluso enviando información a través del puerto serie.

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

FIGURA 20. ARDUINO UNO R3 [39]

Además, la placa cuenta con un puerto USB que permite la conexión con un PC, para realizar la carga del programa y el intercambio de información. De igual modo, se ve que existen pines que permiten la transmisión serial con cualquier dispositivo que admita este sistema de comunicación. También, se puede apreciar la existencia de un conector de alimentación que permite alimentar la placa desde cualquier fuente con tensión en el rango 7-12 Vcc, ya que la placa contiene un regulador de tensión a 5 Vcc, para su funcionamiento. Esto permite operar con la placa mediante una batería o fuente de alimentación independiente, posteriormente a la carga del programa y simulaciones.

Con todo, se podría emplear un circuito basado en el microcontrolador de Arduino. Como puede verse en la protoboard de la figura 21, el circuito solo consistiría en los elementos necesarios para que el microcontrolador pueda realizar sus operaciones. Una vez realizado el prototipo y verificado su funcionamiento, sería viable el diseño del layout y la fabricación de una PCB con las conexiones de entrada/salida necesarias, ocupando un espacio muy inferior al que requiere la placa Arduino.

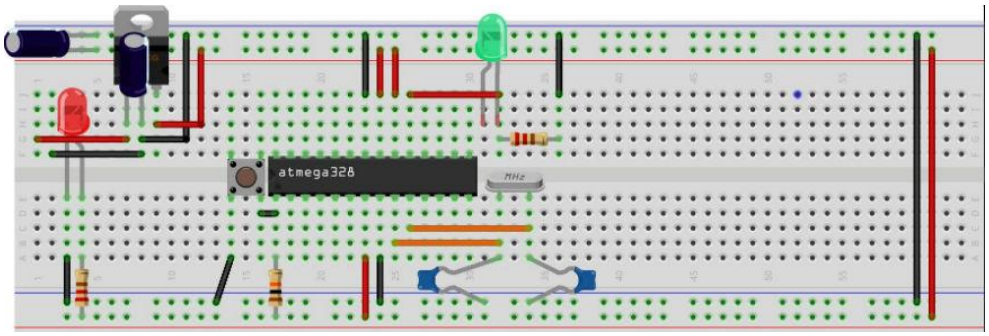


FIGURA 21. PROTOTIPO CON ATMEGA328 [39]

3.2.3 Sistema de monitorización y posible automatización.

En este punto bastaría con integrar los sensores junto con el módulo Arduino y los actuadores necesarios. Sin embargo, no se ha comentado nada sobre actuadores porque **inicialmente no se contempla actuar sobre ningún sistema**, por lo que simplemente se expone que es posible actuar sobre cualquier elemento, pero se requiere emplear relés y contactores. Arduino no tiene capacidad para activar directamente estos actuadores (su corriente de salida es muy baja). En la figura 22 se puede ver a la izquierda un módulo de 2 relés típicamente empleados en el entorno Arduino, mientras que a la derecha se presenta un contactor de 380 V. A modo de ejemplo, el relé podría ser activado por un pin digital de salida de la tarjeta Arduino (baja potencia) y este a su vez activar el contactor (alta potencia), permitiendo operar sobre actuadores de gran potencia.

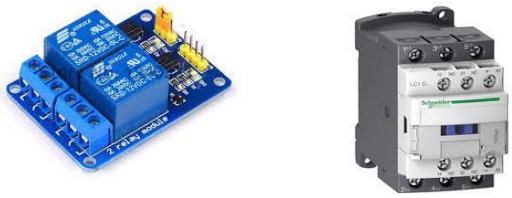


FIGURA 22. EJEMPLOS DE RELÉ Y CONTACTOR [39]

Por tanto, como puede verse en la figura 23, para la monitorización y automatización de procesos en el depósito de agua, el circuito es bastante sencillo. Solo es necesario

respetar las conexiones indicadas en los datasheets de los distintos periféricos y tener en cuenta las tensiones de alimentación y las tierras, que deben ser comunes a todos los elementos. Solo comentar, que dado que no hay un sensor de pH en el software con que se ha realizado el esquema, éste se ha sustituido por una fuente que proporciona una tensión variable en el rango previsto y que varía con una frecuencia determinada de antemano. También, se podría emplear una fuente de tensión fija y modificarla manualmente en cada simulación, pero sería más entretenido. Aun así, no se realiza ninguna simulación, porque esto solo pretende dar una idea del diseño (incluyendo posible actuación sobre bombas u otros mecanismos), el cual, en su momento, se materializará en un prototipo.

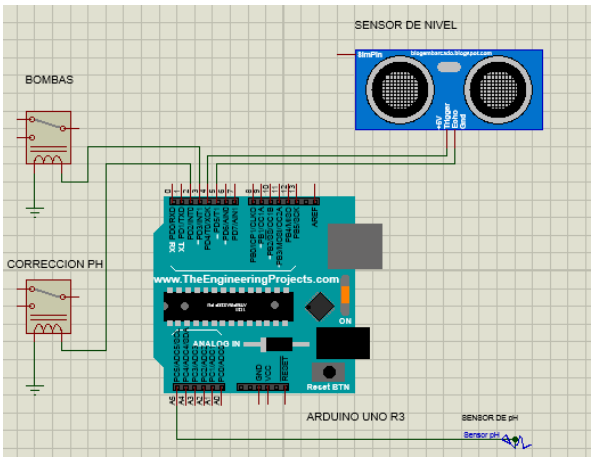


FIGURA 23. ESQUEMA SISTEMA DE MONITORIZACIÓN Y AUTOMATIZACIÓN

3.3 Comunicaciones LoRa.

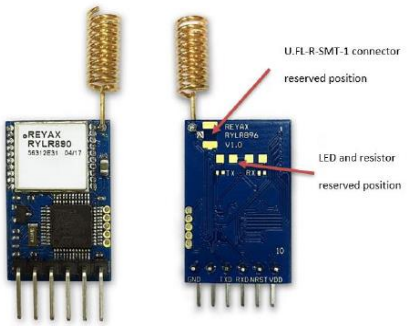
En este apartado se detalla todo el proceso para integrar un sistema de comunicaciones basado en la tecnología LoRa en el sistema de monitorización y automatización. Para ello, primero se requiere el análisis y comprensión de los módulos LoRa que se pueden emplear, así como comprobar su funcionalidad estableciendo un enlace radio y realizando algunas pruebas.

3.3.1. Módulos LoRa

A continuación, se enumeran una serie de módulos que incluyen una radio LoRa y que pueden ser usados como medio de comunicación del sistema. Entre ellos existen bastantes diferencias, debido a que el grado de complejidad es diferente y por tanto su funcionalidad así como su precio.

Módulo RYL869 de REYAX.

Se trata de una tarjeta muy básica que contiene una radio LoRa y los pines necesarios para conectarlo a cualquier dispositivo con capacidad de comunicaciones serie. Como puede verse en la figura 24, es un módulo muy sencillo y fácil de conectar cuyas características se presentan en la tabla 9.



Pin	Name	I/O	Condition
1	VDD	I	Power Supply
2	NRST	I	RESET(Active Low)100KΩ Internal pull up, Pull down at least 100ms
3	RXD	I	UART Data Input
4	TXD	O	UART Data Output
5	NC	-	
6	GND	-	Ground

FIGURA 24. MÓDULO RYL869 PINOUT [44]

Item	Min.	Typical	Max.	Unit	Condition
VDD Power Supply	2	3.3	3.6	V	VDD
RF Output Power Range	-4		15	dBm	
Filter insertion loss	1	2	3	dB	
RF Sensitivity	-148			dBm	
RF Input Level			10	dBm	
Frequency Range	862	868/915	1020	MHz	
Frequency Accuracy		±2		ppm	
Communication Range		4.5	15	KM	Depend on RF parameter
Transmit Current		43		mA	RFOP = +15 dBm
Receive Current		16.5		mA	AT+MODE=0
Sleep Current		0.5		uA	AT+MODE=1
Digital Input Level High	0.7*VDD		VDD	V	VIH
Digital Input Level Low	0		0.3*VDD	V	VIL
Digital Output Level High	0.9		VDD	V	VOH
Digital Output Level Low			0.1	V	VOL
Cycling (erase / write) EEPROM data memory		300		K	Cycles
Weight		7		g	
Operating Temperature	-40	25	+85	℃	

TABLA 9. CARACTERÍSTICAS DEL MÓDULO RYL869 [44]

Módulo E32-868T20D de EBYTE.

Se trata de un módulo bastante sencillo, pero un poco más completo que el anterior. Según su datasheet, dentro de la gama E32 de este fabricante, este módulo en concreto presenta una potencia de transmisión de 20 dBm (100mw) con un alcance aproximado de 3 Kms, en la banda de 868 MHz. Está un poco limitado comparado con el E32-868T30D, que presenta una potencia de 30 dBm (1000 mW) con alcance de hasta 8 Kms. Aun así, su precio es bastante inferior (aproximadamente 10 euros). En la figura 25 pueden verse vistas superior e inferior donde se aprecia el SoC y los pines de conexión. Además, en la tabla 10 se presenten sus características más importantes. Básicamente, está basado en el SX1276 de SEMTECH, al igual que la mayoría de módulos LoRa, y en este caso posee varios modos de funcionamiento en función del estado de los pines M0 y M1. Estos modos de operación son: normal, despertar, dormido, y ahorro energético, y el microcontrolador al que se conecta el módulo es el que decide en qué modo opera el módulo, mediante los pines de salida del micro.

FIGURA 25. MÓDULO E32-868T20D [45]

Frequency Band	862 - 893MHz	Default: 868MHz, channel:32
Supply voltage	2.1 - 5.5V DC	5V (recommended)
Communication voltage	Maximum 5.2V	-
Transmitting power	20dBm	Can be configured to 20, 17, 14, 10dBm
Air data rate	2.4kbps	Can be configured to 0.3, 1.2, 2.4, 4.8, 9.6, 19.2 kbps
Standby current	4.0uA	M1=1, M0=1 (Mode 3)
Transmitting current	120mA@20dBm	≥300mA
Receiving current	14mA	Mode 0 or Mode 1
Sensitivity	-138dBm/0.3kbps	Sensitivity has nothing to do with baud rate or timing

TABLA 10. CARACTERÍSTICAS DEL MODULO E32-868T201D [45]

Módulo Dragino Lora Shield.

Se trata de un escudo para Arduino con un factor de forma que permite encastrar directamente la tarjeta sobre una placa Arduino compatible, como puede verse en la figura 5. Es una tarjeta que cuenta con todo lo necesario para que la conexión con la placa Arduino sea muy simple y eficiente, permitiendo incluso que algunos de sus pines sean transparentes y permitan acceso al exterior a la tarjeta Arduino. Al igual que los anteriores, está basado en el chip SX1276 de SEMTECH, que es lo que proporciona la radio Lora, siendo sus características principales las siguientes:

- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -148 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation.
- 127 dB Dynamic Range RSSI.
- Built-in temperature sensor and low battery indicator.
- Compatible with 3.3v or 5v I/O Arduino Board.

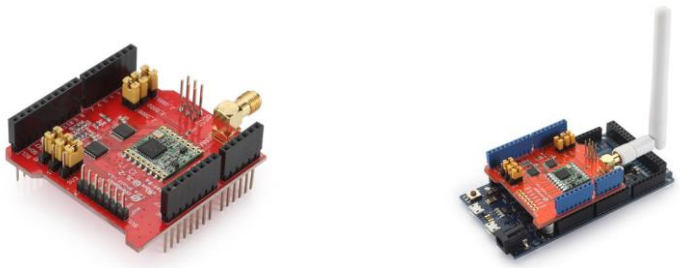


FIGURA 26. MÓDULO DRAGINO LORA SHIELD [46]

Módulo TTGO LoRA32 V2.1_1.6

Se ha escogido este módulo debido fundamentalmente a su bajo precio (aproximadamente 15 euros) y a que tiene una pantalla OLED que facilita verificar las comunicaciones. En la figura 27 se muestran 2 vistas del módulo, donde pueden verse todos los integrados, pines de entrada/salida y una la pantalla OLED de 0.96 pulgadas,

que facilita tener acceso a datos sin necesidad de emplear el monitor serie del IDE de Arduino. También, cuenta con un zócalo para tarjetas de memoria SD, así se puede almacenar información en esta o leer desde la misma.

FIGURA 27. TTGO LORA32 V2.1_1.6 [47]

Además, este módulo también presenta conectividad WiFi y Bluetooth, por lo que su incorporación a otras redes es viable, permitiendo escalar hacia sistemas más complejos. Sin duda, es un módulo muy adecuado para el ámbito IoT gracias a la radio LoRa que trabaja en las bandas 433/868/915 MHz, dependiendo de la región. El alcance viene determinado por el entorno (urbano o rural), de forma que al ajustar el spreading factor (SF) varía la sensibilidad. Sus características principales se presentan el al tabla 11 y el pinout en la figura 28.

Tensión e trabajo	**1.8 – 3.7 V**
Corriente de trabajo	10-14 mA
Frecuencias de trabajo	433/868/915 MHz
Sensibilidad	Depende del SF
Potencia de transmisión	20 dBm
Tasa de transmisión	0.018K ~ 37.5Kbps en LoRa
Tª de trabajo	-40 ºC a 85 ºC
Corriente en reposo	0.2uA dormido y 1.5uA ocupado

TABLA 11. CARACTERÍSTICAS MÓDULO TTGO LORA V2.1_1.6

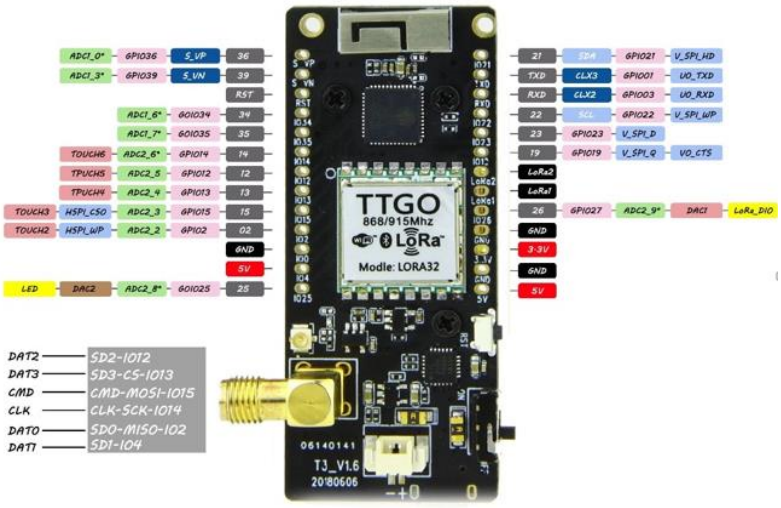


FIGURA 28. PINOUT DEL MÓDULO TTGO LORA32 V2.1_1.6 [47]

Hasta aquí, se ha realizado una aproximación a distintas opciones de hardware existentes dentro del ámbito de las comunicaciones basadas en LoRa. Se trataba de realizar una búsqueda de opciones existentes para poder determinar las existentes y cuál de ellas puede ser más interesante para implementar en el sistema. Por otra parte, es necesario aclarar que estos módulos se pueden emplear de dos formas: **enlaces punto a punto** o formando parte de una **red en estrella**. En caso de recurrir a una red en estrella, donde los nodos finales se conectan a un Gateway para poder subir información a un servidor en Internet, se estaría dentro del ámbito LoRaWAN.

Finalmente, en el presente proyecto, se pretende empleará dos módulos para establecer un enlace punto a punto, donde el primer módulo se encuentra integrado dentro del sistema de monitorización y automatización que se ubica en el depósito de agua y el segundo módulo estará en las dependencias municipales que el ayuntamiento indique. De este modo, será posible tener una lectura instantánea de los datos más actualizados que se han recibido y en caso necesario se podrá actuar antes de que se presente algún problema. Para ello, se propone el envío de los parámetros medidos dos veces al día: 08:00 y 18:00 horas, de forma que se puede ver si el depósito se ha recuperado durante la noche y como ha disminuido a media tarde, a la vez que conocer el pH.

3.3.2 Configuración y pruebas de enlace punto a punto.

En primer lugar, para poder realizar una serie de pruebas se recurre a los módulos TTGO LORA32 V2.1_1.6, ya que presentan la ventaja de la pantalla OLED, que permite la visualización de la información sin recurrir a elementos externos. Una vez adquiridas las tarjetas LoRa se requiere comprender su funcionamiento y probar su operatividad. Para ello, hay que recurrir al pinout de las placas entendiendo que función tiene cada pin y buscar las librerías necesarias para que el IDE Arduino pueda trabajar con este tipo de módulo. De este, modo es posible configurar el código necesario para probar que los módulos establecen un enlace punto a punto. Así, cargando en el módulo que hace de emisor el código necesario y en el que actúa como receptor el correspondiente, es posible establecer el enlace y enviar datos. Esta tarea no está carente de dificultad y tiempo, debido a la variedad de módulos LoRa existentes (diferencias en el pinout) así como a la diversidad de librerías. En los anexos se presentan ambos códigos.

Por otro lado, una vez realizados los sketchs de transmisión y recepción y cargados en los correspondiente módulos, se puede verificar su funcionamiento. Así, en la figura 29 se muestra que la prueba de enlace punto a punto es correcta. En esta figura pueden verse ambos módulos LoRa (433 MHz), el que se encuentra a la derecha actúa como emisor y el de lado izquierdo actúa como receptor. Como puede verse en las pantallas OLED, el emisor (está identificado como LoRaSender) envía un mensaje con una periodicidad que este caso ha sido fijada en cinco segundos y al final se ve el número de mensajes enviados. En cambio, el receptor (LoRa Receiver) informa sobre el mensaje recibido, el número de mensajes actual y sobre la potencia de la señal recibida expresada en dBm. También, se aprecia que ambas tarjetas están alimentadas con sendas baterías, de forma que no interviene para nada un PC. En esta simulación no se ha recurrido a ningún sensor, simplemente se trataba de ver si se establecía el enlace de comunicaciones, por ello solo se ha subido el código a cada tarjeta y se las ha alimentado mediante las baterías que se ven en la imagen.

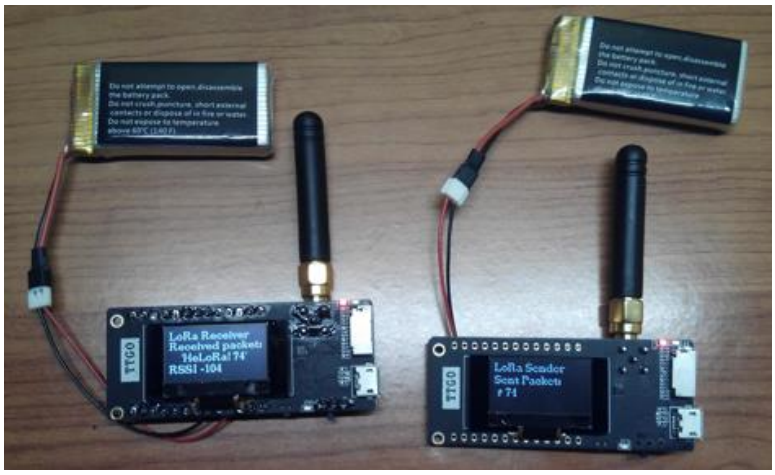


FIGURA 29. PRUEBAS ENLACE PUNTO A PUNTO

Aparte, se requiere tener claras algunas de las instrucciones que aparecen en el sketch, sobre todo las relativas a la configuración de la radio LoRa:

- La frecuencia de operación debe ser consistente con el módulo LoRa empleado, en este caso se puede escoger cualquier frecuencia en el rango 433-470 MHz.
- La frecuencia empleada debe estar permitida en la región donde se implementará el sistema, de modo que en el caso de este proyecto se pude emplear la banda de 433 o de 868 MHZ, pero no se puede emplear la de 915 MHz.
- El spreading factor (SF) necesario será aquel que permita el mayor alcance y la menor tasa de transferencia, en este caso el valor será 12. Si el SF es menor permite mayor tasa de transferencia pero menor alcance, debido a que presenta menos inmunidad frente a interferencias.
- La instrucción *"LoRa.setTxPower (17,PA_OUTPUT_PA_BOOST_PIN);"* es la que permite mayor potencia de transmisión, es decir la máxima potencia en condiciones normales de uso del amplificador antes de la antena.

A continuación, una vez claro todo lo anterior se requiere comprobar el alcance del enlace y si es necesario realizar las modificaciones oportunas para lograr que el enlace de comunicaciones sea operativo. Así, en la tabla 10 se muestran los valores

obtenidos (son aproximados) en una prueba real en un entorno al aire libre y sin obstáculos.

Distancia [m]	RSSI [dBm]
10	-34
50	-65
100	-92
200	-96
500	-105
1000	-118

TABLA 12. RELACIÓN DISTANCIA - POTENCIA RECIBIDA

También, se ha podido comprobar que el enlace es operativo situando emisor y receptor en dos edificios distintos separados aproximadamente 100m en línea recta, donde se observó una potencia recibida de -122 dBm. Con todo, en caso necesario, se puede optar por emplear una antena que proporcione mayor ganancia, de forma que el alcance del enlace pueda ser mayor, como puede verse en [36].

Seguridad del enlace.

Se requiere tener claras dos consideraciones acerca de las comunicaciones LoRa en materia de seguridad:

- Cualquier dispositivo que esté operando en la misma frecuencia que la que se está empleando en este caso (433 MHz) y que se encuentre dentro del rango del alcance puede recibir los datos que se están transmitiendo.
- LoRa no encripta la información, por ello si se requiere su encriptado se debe hacer antes de enviarlo al modem radio y una vez recibida en el receptor desencriptarla.

Estos aspectos no son críticos en el sistema debido a que los datos no suponen ningún activo de valor para nadie (no es susceptible de robo), excepto para los habitantes del municipio. Por ello, en principio no procede emplear ningún sistema de

seguridad que impida el acceso a los datos. En todo caso, podría tenerse en cuanta de cara a mejoras futuras a implementar en el sistema.

3.3.3 Pruebas con sensor DHT11 y TTGO.

En este punto y partiendo del anterior apartado, se propone verificar la capacidad de medir la temperatura y humedad ambiental mediante el ampliamente conocido DHT11 y enviar esta información empleando la tecnología LoRa. Para ello, se recurre a los módulos TTGO del anterior apartado, donde el emisor tendrá acoplado el sensor. Se ha realizado un prototipo y se ha comprobado que con la programación necesaria el emisor es capaz de medir la temperatura y humedad y transmitirla al receptor, figura 30. El emisor (a la izquierda) tiene conectado el sensor y emplea un resistencia de pull-up en su entrada de señal (por ello es necesaria la protoboard). El receptor muestra la señal de temperatura y humedad enviada por el emisor, así como la potencia de señal recibida. Como puede verse, los módulos de comunicaciones una vez programados simplemente se alimentan mediante sendas baterías y ya pueden realizar su cometido. En los anexos se incluyen tanto los sketchs de pruebas como más imágenes.

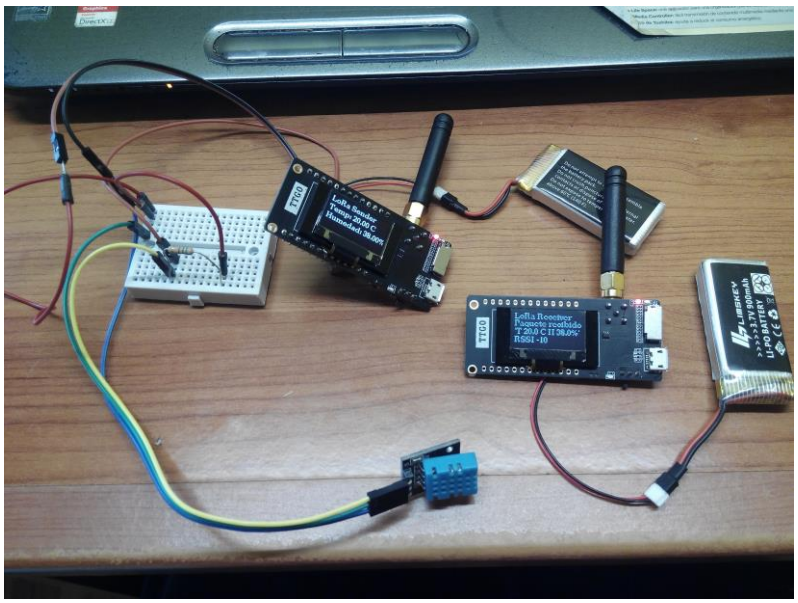


FIGURA 30. PRUEBAS CON TTGO Y DTH11

3.4. LoRaWAN y TTN

En este punto, parece interesante ir un poco más allá y añadir nuevas características al diseño planteado. Por ello, se propone incluir LoRaWAN en sistema, de forma que se posible enviar los datos a un servidor en Internet. Para ello, se realiza a continuación la exposición de todo lo necesario para su consecución, empleando The Things Network (TTN).

3.4.1 The Things Network.

The Things Network (TTN) es una iniciativa basada en la comunidad para establecer una red global de IoT dentro del contexto LoRaWAN. Actualmente cuenta con 7139 pasarelas LoRaWAN instaladas en más de 137 países con 68132 miembros [37]. Por tanto, se trata de un entorno muy adecuado para poder monitorizar el estado de sensores que emplean LoRaWAN desde la red de Internet, incluso proporcionando múltiples aplicaciones. Así es posible subir los datos de los sensores al servidor TTN y emplear servidores de aplicaciones para poder tratar estos datos como mejor convenga. Como puede verse en la figura 31, la cobertura de la red no es óptima, pero es indudable el continuo crecimiento de la misma, gracias a voluntarios que despliegan sus propios gateways y dan cobertura LoRaWAN.

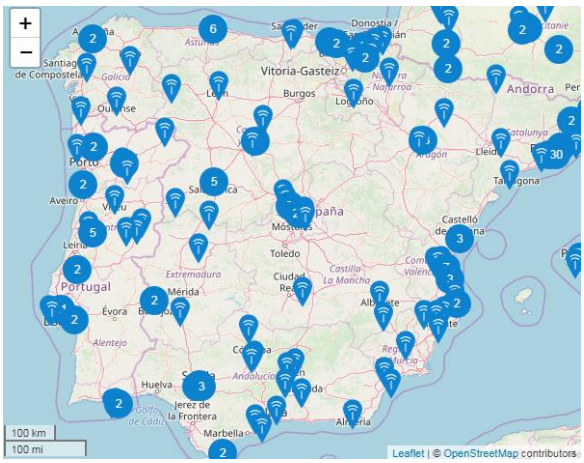


FIGURA 31.COBERTURA TTN EN ESPAÑA [49]

Arquitectura de red.

Como puede verse en la figura 32, la arquitectura necesaria para poder gestionar la información de los sensores desde Internet es la correspondiente a LoRaWAN. En esta arquitectura, las partes esenciales son:

- **Sensores:** nodos finales con capacidad de comunicación LoRa, se conectan con las pasarelas empleando la radio LoRa. Un nodo puede tener acceso a varios gateways.
- **Pasarelas:** reciben la información de los sensores mediante LoRa y la envían a Internet mediante conexión WiFi, Ethernet, 3G, etc. Cada pasarela determina una zona de cobertura LoRa a la que se conectan los nodos finales dentro que están dentro de esta zona.
- **Servidor de red**: es el elemento que recibe y almacena los datos y puede conectarse con servidores de aplicaciones para la gestión de los mismos. Este es el servidor TTN en nuestro caso.
- **Servidores de aplicaciones**: permiten implementar aplicaciones para trabajar con los datos recibidos

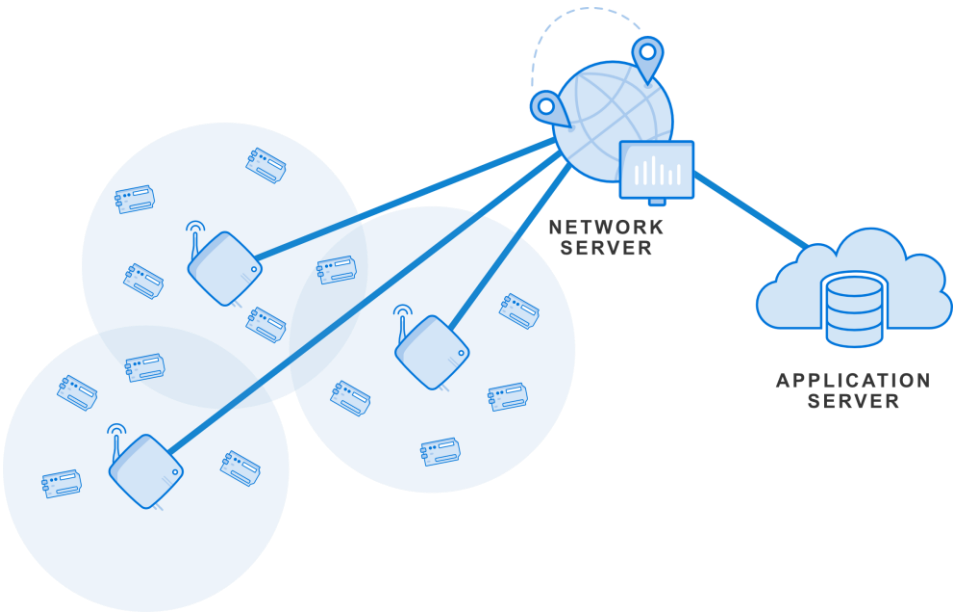


FIGURA 32. ARQUITECTURA DE RED LORAWAN [50]

3.4.2 TTN en este proyecto.

En primer lugar, parece interesante poder transmitir la información de los sensores a un servidor que proporcione acceso a los datos que se monitorizan en este proyecto. Esto permite no depender de la ubicación del receptor, como sucede en los enlaces punto a punto. Es decir, una vez que los datos están en un servidor son fácilmente accesibles desde cualquier dispositivo con acceso a Internet y desde cualquier lugar con cobertura, siempre y cuando se tenga autorización. Esto constituye una mejora importante respecto al enlace punto a punto, pero además proporciona herramientas para la gestión de los datos (servidores de aplicaciones) que en un enlace punto a punto serían muy complicadas.

Por un lado, como ya se ha comentado, se requiere de una pasarela para poder subir los datos de los sensores a Internet. En el caso del presente proyecto no hay ninguna disponible dentro del radio de alcance, de modo que es necesario adquirir y configurar una. Una pasarela es un router que incluye un concentrador LoRa que permite recibir paquetes LoRa. Estos paquetes serán convertidos a TCP/IP y enviados al servidor de red mediante una conexión del tipo que sea (Ethernet, WiFi, 3G/4G). Por ello, pensando en una alternativa sencilla, se puede emplear una gateway como el de la figura 33, siendo un dispositivo de un solo canal por lo que es bastante asequible (aproximadamente 60 euros). Existen pasarelas de 8 canales que permiten el acceso de un número muy superior de nodos finales, pero su precio es bastante elevado.

+ Managed by web GUI, SSH via LAN or WiFi
+ Support WiFi AP, Client or ad hoc (mesh) mode
+ Internet connection via LAN, WIFI, 3G or 4G.
+ Lora band 868MHz
+ Max range in Lora: 5 ~ 10 km

FIGURA 33. GATEWAY DRAGINO LG01 [51]

Por otro lado, se requiere registrar la pasarela dentro de TTN para poder emplearla. Primero es necesario configurar el gateway siguiendo las instrucciones de su datasheet y, después, siguiendo los pasos que se muestran en los anexos es sencillo de realizar el registro del dispositivo en TTN. Una vez registrado el dispositivo ya es posible enviar datos provenientes de cualquier dispositivo LoRa dentro de su radio de acción, pero es necesario que los nodos finales tengan el software adecuado y se registren en TTN. En este caso, además, se requiere registrar en TTN el nodo final que se emplea en este proyecto para que pueda enviar la información a través del gateway. En los anexos se incluye una pequeña explicación del proceso de registro y configuración de ambos dispositivos: gateway y nodo final así como la creación de la aplicación, dentro de la que se registra el nodo.

Finalmente, sería interesante estudiar las aplicaciones que se pueden emplear para manejar los datos, pero esto queda fuera del alcance del presente proyecto, pudiendo ser incluido como mejora futura del sistema. Además, la finalidad principal es la capacidad de trasmisión de la información y su visualización, no la gestión.

3.4.3 Pruebas con TTN.

Una vez configurado todo lo necesario para que el sistema sea integrado en TTN se procede a realizar una serie de pruebas para comprobar que todo es correcto y obtener algunas conclusiones. Se exponen a continuación algunos aspectos acerca de la aplicación, el dispositivo y el gateway creados:

1. Una vez se tiene cuenta de usuario en TTN, siguiendo los pasos necesarios se crea la aplicación en la consola de TTN, que como puede verse en la figura 34, en el caso del presente proyecto se identifica como “depositoaguagdi”.

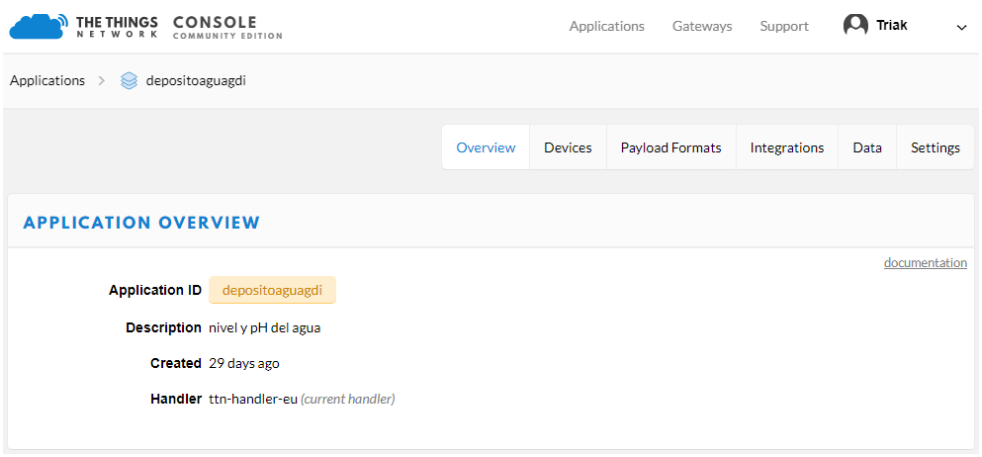


FIGURA 34. APLICACIÓN EN TTN

2. Dentro de esta aplicación se registra el dispositivo que enviará la información. En la figura 35 se ven los principales aspectos a tener en cuenta para la programación del software que gestionará el envío de los paquetes a TTN. Las claves que aparecen son necesarias para que la aplicación creada reconozca la identidad del dispositivo que envía la información y para poder encriptar dicha información. También, en este caso, como método de activación se ha empleado ABP (Activation By Personalization) en lugar de OTAA (Over The Air Activation), debido a que el gateway empleado es de un solo canal y presentaría muchos problemas si se empleara OTAA.

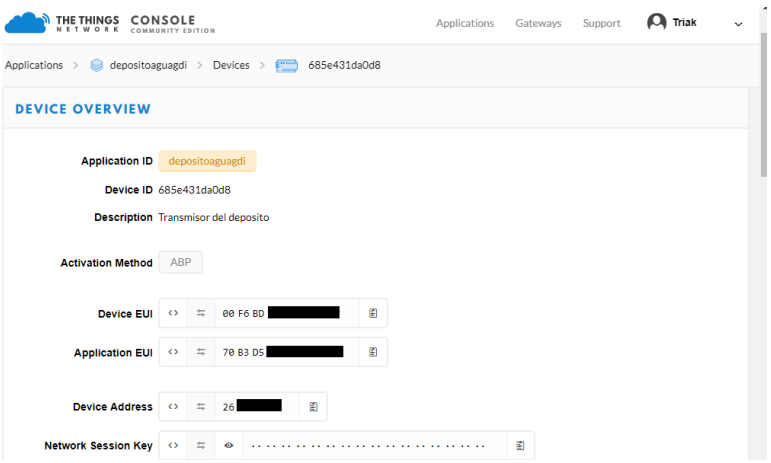


FIGURA 35. DISPOSITIVO FINAL EN TTN

3. Se registra el gateway en TTN tal como se explica en los anexos, que previamente se ha debido configurar y el resultado puede verse en la figura 36. En la imagen se ven los aspectos clave del registro del gateway, incluso puede verse como el nodo final ha estado enviando mensajes y éste los ha retransmitido a TTN (109 mensajes).

FIGURA 36. GATEWAY REGISTRADO EN TTN

4. Una vez que todo está registrado y configurado correctamente, al alimentar el nodo final y comenzar a transmitir, en la figura 37 puede verse como los datos de prueba llegan al servidor TTN. El sketch cargado en el nodo final implementa una cadena de texto (para pruebas), donde el payload corresponde con dicha cadena: *PRUEBAS TTGO!* Es decir, si se convierte a ASCII el mensaje en Hexadecimal que se ha recibido, es exactamente la cadena que se ha enviado. También, se ven datos relativos al gateway que ha intermediado en la transmisión, que obviamente es el que se ha implementado en este proyecto.

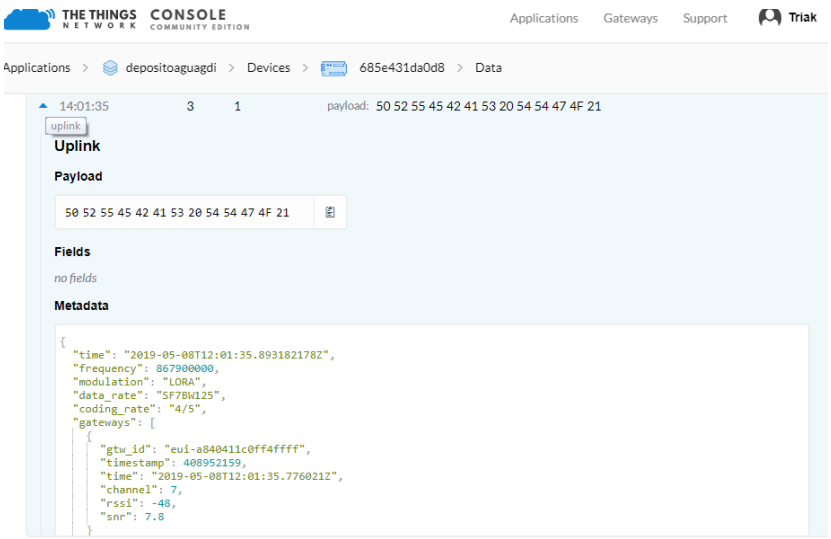


FIGURA 37. EJEMPLO DE PAQUETE RECIBIDO

5. Si se cambia la cadena de texto enviada por el nodo final, simulando una lectura real del sistema, se puede ver en la figura 38 una serie de paquetes correspondientes a esta simulación. El paquete superior, en su parte derecha muestra el contenido del payload decodificado y puede verse que todos os paquetes llevan el mismo payload, como es normal.

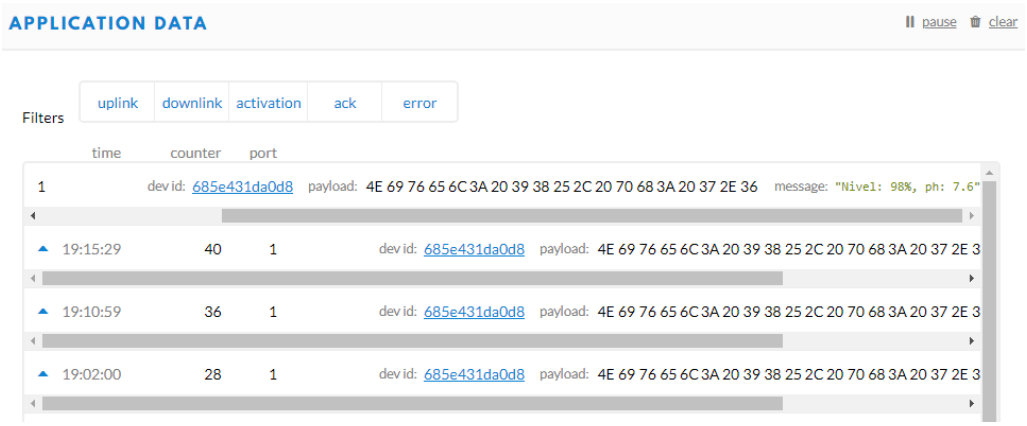


FIGURA 38. SIMULACION DE PAQUETES RECIBIDOS

Por tanto, hasta aquí se ha tratado de demostrar la viabilidad de emplear LoRaWAN y TTN como medio de mejora del sistema diseñado. Se ha podido comprobar que la

configuración de todos los elementos es correcta y las simulaciones realizadas muestran que es viable implementar esta mejora en el diseño del sistema.

Aparte, En la figura 39 se muestra el dispositivo final (TTGO) que contiene el sketch que envía la cadena de texto simulando los datos. De igual modo se muestra en la pantalla del ordenador el monitor serie del IDE de Arduino donde se aprecian los mensajes enviados y que se han podido ver en anteriores imágenes. Cada vez que el nodo final envía un paquete de datos a TTN espera la confirmación y cuando esta se produce muestra en la pantalla OLEC un mensaje “ACK Received” y se queda esperando el próximo envío de datos. También, presenta el número de paquetes enviados hasta el momento. Todos estos aspectos simplemente se han configurado en el sketch para realizar pruebas y poder comprobar la conectividad entre el dispositivo final y TTN.

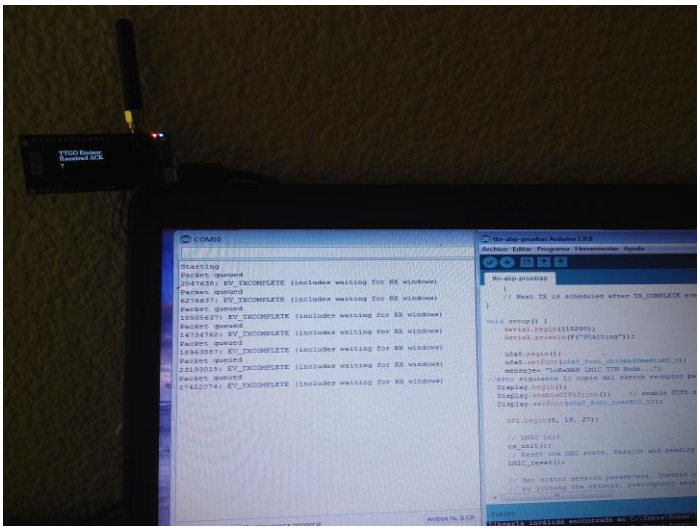


FIGURA 39. PRUEBAS CON TTGO Y TTN

Finalmente, hay que decir que se ha podido comprobar que TTN sólo muestra datos cuando se abre la web de la aplicación, de forma que los datos que va recibiendo son los que se pueden visualizar, siendo imposible tener acceso a datos anteriores. Por lo que se ha podido averiguar, en [38] se expone que se requiere una integración de almacenaje, de forma que TTN presta un servicio de almacenamiento de datos gratuito durante 7 días. Así, mediante consultas SQL se puede acceder a los datos

almacenados, como puede verse en las figuras 40 y 41, donde se presentan tanto la integración de almacenaje como una consulta.

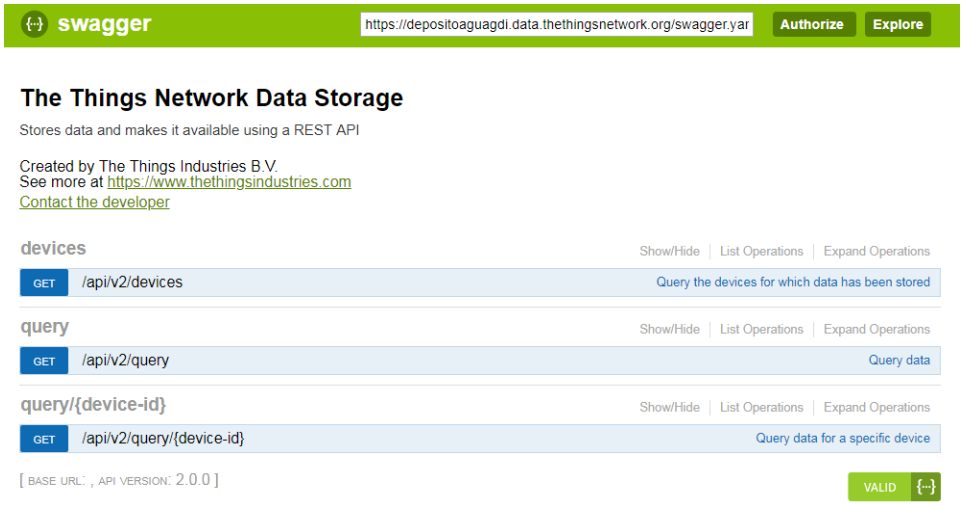


FIGURA 40. TTN - INTEGRACION DE ALMACENAJE DE DATOS

FIGURA 41. TTN - CONSULTA A LA BASE DE DATOS

3.5 Resumen del capítulo

A lo largo de este capítulo se ha puesto en escena el conjunto de tecnologías hardware y software que se emplean en el diseño del sistema y construcción del prototipo. Por un lado, se ha realizado una pequeña introducción a Arduino y a la tecnología LoRa como ejes vertebradores del proyecto y se han enumerado algunas de sus características. Por otro lado, se ha expuesto como es el sistema de monitorización y posible automatización en este proyecto y sus elementos. También, se ha analizado el sistema de comunicaciones basado en LoRa, donde se han expuesto algunos modelos de módulos a existentes y se han realizado algunas pruebas para verificar la operatividad de las comunicaciones y alcance. Incluso, se ha experimentado con un módulo TTGO LoRa32 para medir temperatura y humedad mediante el sensor DHT11 y transmitir esta información, resultando que la experiencia ha sido positiva (se puede medir y se pueden transmitir las medidas). También, se ha estudiado y propuesto incluir LoRAWAN y TTN en el sistema, lo que permite una integración completa con el paradigma IoT. Para ello, ha sido necesario un complejo estudio del contexto, posibilidades y requisitos hardware y software, dando lugar a que se ha impuesto la necesidad de adquirir y configurar un gateway que permita a este sistema subir los datos a Internet. Con todo, pese a las dificultades, ha sido posible configurar todos los elementos necesarios (tanto dentro de TTN como en los equipos) para simular el envío de datos desde el dispositivo final al servidor TTN, a través del gateway que se ha adquirido y configurado. Finalmente, ya se comentó que la automatización de procesos, aunque es viable y se ha propuesto, no se ha realizado por ser bastante complejo. Esto es debido a los permisos necesarios y a la necesidad de colaboración con personal técnico cualificado en materia de calidad y seguridad del agua, aun así es posible escalar el sistema y en un futuro introducirlo como mejora.

4. Prototipo y pruebas

En este capítulo se expone el proceso de instalación y pruebas del prototipo en la ubicación final, basado en lo expuesto en el capítulo anterior, de forma que se pueda obtener un sistema completamente funcional y demostrar su operatividad. Esta fase puede presentarse de forma incremental, pasando de un prototipo básico hacia algo más complejo y elaborado.

4.1 Ubicación.

La idea fundamental de este proyecto era el diseño del sistema y su posterior verificación mediante la construcción del prototipo y su instalación en el depósito municipal de suministro de agua. Por tanto, es momento de especificar el emplazamiento del prototipo, de forma que quede más claro el lugar concreto donde se ha instalado.

En primer lugar, el depósito de agua se encuentra situado en el municipio de Gumiel de Izán (Burgos), en una zona elevada conocida como El castillo, cuya cota de nivel es superior a todas las viviendas, figura 42. De esta forma, el agua llega a las mismas por su propio peso, de igual modo el agua llega al depósito por gravedad, desde dos manantiales a varios kilómetros de distancia, gracias a que dichos manantiales se encuentran también a mayor altura que el depósito: todo es por diferencia de nivel.

FIGURA 42. DEPÓSITO DE AGUA MUNICIPAL

Por otro lado, la capacidad del depósito es de aproximadamente 62 metros cúbicos, con una dimensiones de 6x4x3 (largo x ancho x alto). Actualmente, el sistema que regula la cantidad de agua que hay en el mismo consiste en sendas válvulas que se abren o cierran el paso de agua al depósito en función de una boya. Es decir, cuando la boya indica que el nivel ha disminuido aproximadamente un 20% ordena a las válvulas que deben abrirse y cuando alcanza el nivel de llenado se cierran. Además, existe un rebosadero para evitar que el depósito se desborde, en caso de malfuncionamiento del sistema. Todo ello puede verse en la figura 43.

FIGURA 43. VISTA INTERIOR DEL DEPÓSITO.

4.2 Prototipo inicial.

Para empezar, se parte de un sistema basado en la tarjeta Arduino UNO como sistema de monitorización de parámetros y automatización de procesos, junto con los respectivos sensores. Mediante la comunicación serie los datos se envían al emisor LoRa, que inicialmente consiste en el módulo TTGO anteriormente explicado, el cual transmite la información al receptor LoRa (ídem módulo anterior) ubicado en las dependencias municipales que se sugieran por parte del ayuntamiento.

Por un lado, una vez realizado el montaje del prototipo, cada uno de los dispositivos requiere su propio sketch de funcionamiento, los cuales se incluyen en los anexos, de forma que sea posible realizar las funciones correspondientes a cada subsistema. En

estas condiciones, el montaje queda como se muestra en la figura 44, donde se muestran todos los elementos del prototipo diseñado. Para mejor comprensión se enumeran los componentes:

1. **Fuente de alimentación regulada**: para alimentar el sistema de monitorización y automatización (tarjeta Arduino UNO). Se puede alimentar el sistema mediante una batería de 9 V por el mismo conector que se ha empleado en la imagen.
2. **Sensor de nivel**: ya quedó explicado en el apartado correspondiente.
3. **Sensor de pH:** ídem anterior.
4. **TTGO emisor:** recibe la información mediante comunicación serie desde la tarjeta Arduino, muestra los datos en el display OLED y los envía al TTGO Receptor empleando la radio LoRA incorporada.
5. **TTGO Receptor:** muestra la información recibida empleando LoRa en el display.

Hay que añadir, que la parte de actuadores no se incluye, simplemente se han incluido unos leds que realizan la función de testigos de funcionamiento, para simular dichos actuadores.

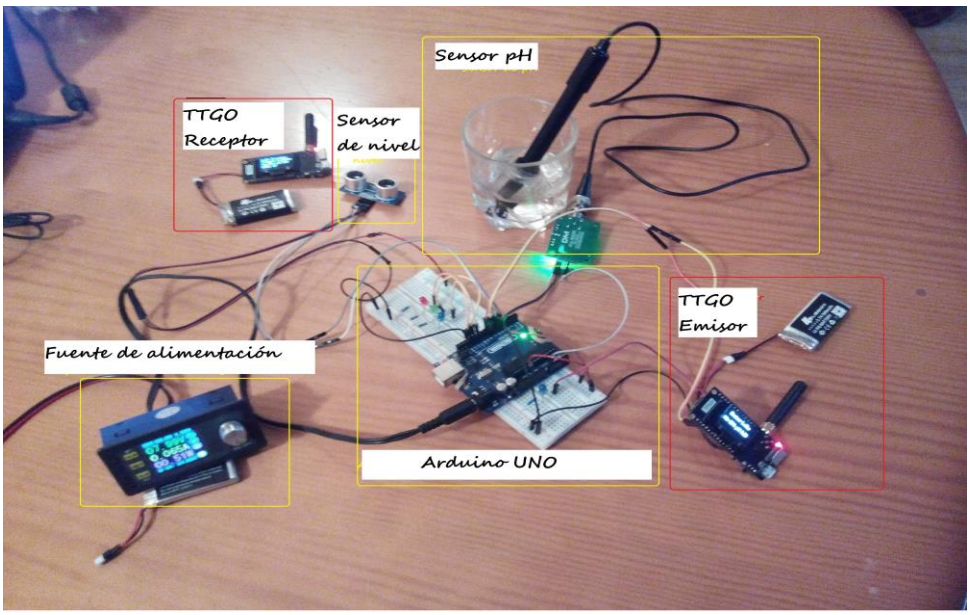


FIGURA 44. PROTOTIPO INICIAL

Por otro lado, para verificar el funcionamiento del prototipo y poder validar que los sketchs programados funcionan bien, se han realizado algunas pruebas simulando variaciones del nivel del depósito y del pH del agua. Para ello, simplemente se recurre a modificar la distancia que mide el sensor correspondiente y modificar el contenido de un recipiente (en este caso un vaso). Pese a que ya se demostró en el capítulo anterior que el sistema de monitorización funcionaba, ahora lo interesante es comprobar que la información se trasmite por LoRa entre los nodos que forman el enlace. De este modo en las figuras 45, 46 y 47 pueden verse estas simulaciones. Entonces, en la figura 45 se muestra la información que el TTGO ha recibido del sistema de monitorización y que es enviada al TTGO Receptor. En este caso, se ve que el nivel medido del depósito es del 59% y que el pH tiene un valor de tres (el sensor estaba en un vaso de agua con una gotas de vinagre). En la figura 46 se presenta un ejemplo de información recibida por el TTGO Receptor, al igual que en la figura 47. La diferencia es que en las simulaciones se han empleado distintos niveles y soluciones con distinto pH, donde lo importante era demostrar que los parámetros medidos eran transmitidos desde el subsistema de monitorización al receptor.

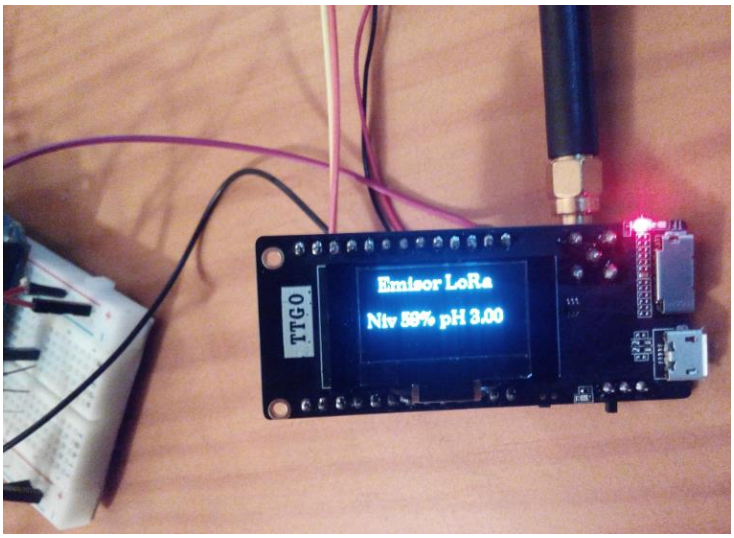


FIGURA 45. PRUEBAS TTGO EMISOR

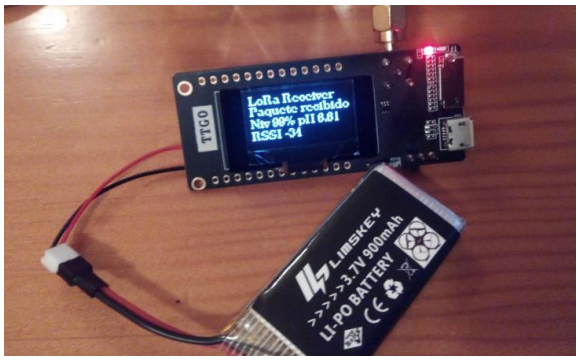


FIGURA 46. PRUEBA_UNO TTGO RECEPTOR

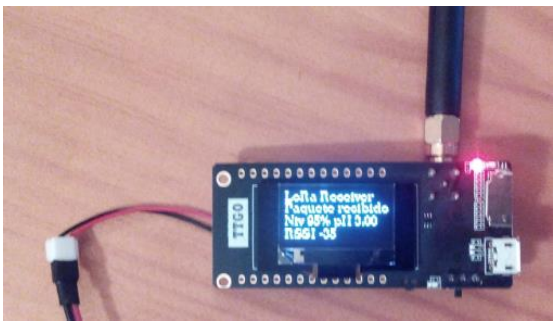


FIGURA 47. PRUEBA_DOS TTGO RECEPTOR.

Por tanto, mediante las pruebas realizadas se ha podido comprobar que el prototipo inicial que se ha construido cumple con sus objetivos: **medir los parámetros que se plantearon inicialmente y transmitir esta información a distancia empleando radio LoRa**. Aunque no se han incluido aquí mas imágenes por no extender este apartado, se puede decir es es bastante obvio que las pruebas han sido satisfactorias, a falta de realizar pruebas de largo alcance (centenas de metros a kilómetros).

Finalmente, hay que aclarar que en este diseño las comunicaciones LoRa se establecen en la frecuencia central de 433 MHz, que no es lo ideal siendo más interesante emplear 868 MHz, porque las pasarelas en Europa suelen emplear esta frecuencia. Es decir, si se pretende enviar información a Internet a través de una pasarela, en este diseño se debería construir o adquirir una que opere a 433 MHz.

4.3 Prototipo definitivo

Partiendo de lo explicado en el anterior apartado y aplicando pequeños cambios se tiene lo que puede considerarse como prototipo definitivo, que es el que se ha instalado en el depósito. Esto consiste en emplear también el módulo TTGO que trabaja a la frecuencia central de 868 MHz, ya que es la banda más empleada en Europa. Así, se añade capacidad de entrar a formar parte de la red LoRaWAN desplegada en España. De este modo, el prototipo definitivo es casi idéntico al que se ha presentado en la figura 44, solo que se ha añadido un módulo TTGO a 868 MHz, con un sketch adecuado para que pueda subir los datos a TTN a través del gateway que se ha adquirido. Por tanto, **el prototipo instalado en el depósito**, que puede verse en la imágenes de la figura 48, consta de los siguientes elementos: **tarjeta Arduino, TTGO** a 433 Mhz (enlace punto a punto), **TTGO a 868 MHz** (datos al gateway para TNN), **sensor de nivel y sensor de pH**. El sensor de pH se encuentra encastrado en un elemento que permite la flotabilidad del conjunto para poder estar siempre en contacto con el agua. El sensor de nivel está sujeto a la pared superior del depósito de agua orientado hacia su superficie. Ambos sensores se han cableado hasta la caja que contiene el resto de módulos y componente, En la caja esta la tarjeta arduino que recibe la información de los sensores, un regulador que alimenta a 5 Vdc ambos sensores y dos emisores TTGO LORA 32 a 868 y 433 MHz. Además, el conjunto se alimenta mediante un regulador variable configurado a 9 Vdc.

FIGURA 48. PROTOTIPO EN EL DEPÓSITO

Por otra parte, se incluye un receptor TTGO 433 MHz para recibir los datos del enlace punto a punto, que se ubica en dependencias municipales, al alcance del responsable de gestión de aguas del municipio. De este modo es inmediato conocer la última lectura de niveles que se ha realizado.

Finalmente, para subir los datos a TTN, se emplea el gateway que se ha adquirido para este proyecto y que está configurado a la frecuencia de 868,1 MHz, al igual que el TTGO emisor que hay en el depósito. En ambos casos se han cargado los sketchs oportunos para permitir la transmisión de información del TTGO emisor al gateway y desde este al servidor TTN. Todo esto se incluye en los anexos, siendo los más importantes la correcta configuración del gateway así como las claves necesarias en el sketch del TTGO emisor. Es imprescindible que TTN sea capaz de identificar la aplicación y el nodo final que está enviando datos y, para ello, al tratarse de una conexión ABP se requieren estos tres parámetros:

- Device Address
- Network Session Key
- App Session Key

Para mayor claridad, en la figura 49 se presenta la arquitectura que permite ver todos los componentes que forman parte del diseño y del prototipo construido.

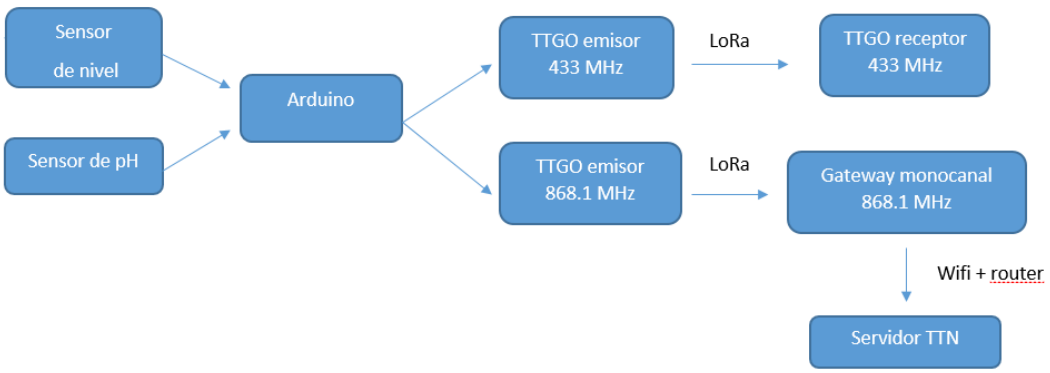


FIGURA 49. ARQUITECTURA DEL PROTOTIPO

4.4 Pruebas.

Una vez instalado en el depósito el prototipo y configurado todo lo necesario, se han realizado las pruebas para comprobar la transmisión de los datos tanto en el enlace punto a punto como a TTN. El objetivo principal ha sido comprobar que el sistema era capaz de medir los parámetros y enviar dichas mediciones empleando LoRa y LoRaWAN.

Por un lado, en el enlace punto a punto se ha comprobado que el enlace es operativo y se reciben la información procedente del depósito. Esto, además permite comprobar que todo funciona correctamente sin tener que acceder a Internet, incluso programando la transmisión con mayor frecuencia es posible tener acceso a mediciones más actualizadas.

Por otro lado, para verificar que TTN recibe los datos, se inicia sesión en el servidor y en la aplicación que se ha creado y se puede comprobar si se reciben los datos. Así, en las figuras 50 y 51 se muestran los datos que recibe el gateway y los datos que recibe la aplicación que se ha creado, provenientes del emisor TTGO a 868.1 MHz.

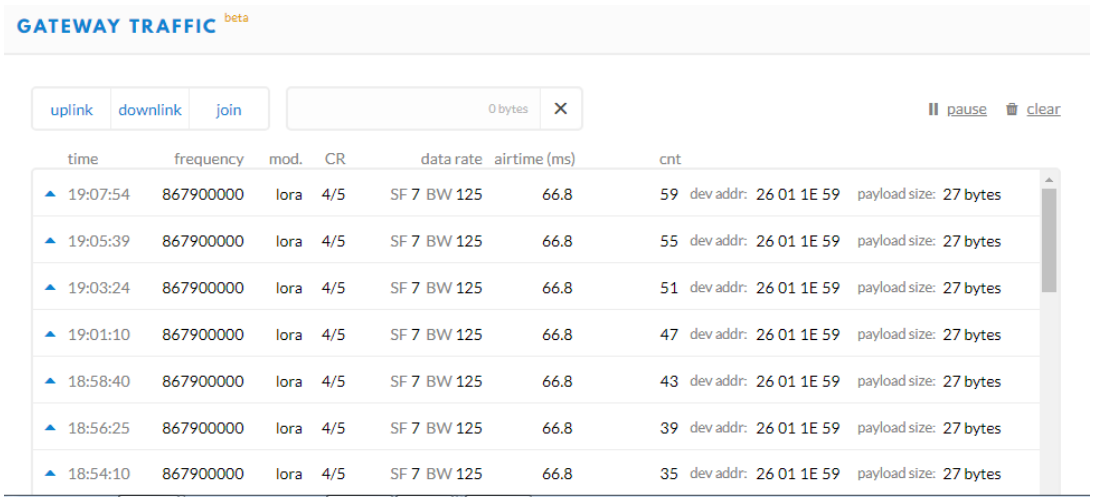


FIGURA 50. TTN - TRÁFICO EN EL GATEWAY

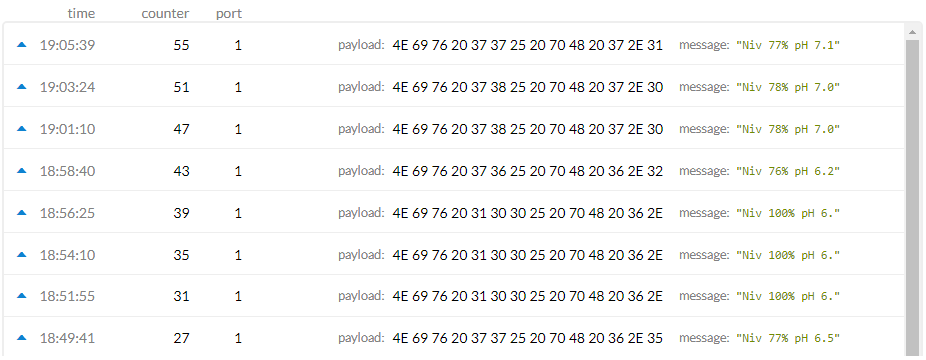


FIGURA 51. TTN - DATOS RECIBIDOS POR LA APLICACIÓN

Finalmente, se ha comprobado que los paquetes se reciben, pese a que la distancia entre el gateway y el emisor es aproximadamente 400 m en línea recta. Además, están condicionantes como que el gateway está en el interior de una vivienda, el emisor está dentro de un depósito de hormigón y el gateway está en una cara del cerro mientras que el emisor está en la contraria, constituyendo todo ello condiciones muy adversas de comunicación. Aun así, como puede verse en la figura 52 la fuerza de la señal está dentro del límite, siendo esta de -121 dBm.

FIGURA 52. TTN - PAYLOAD Y CARACTERÍSTICAS DE UN PAQUETE

4.5 Comentarios sobre los resultados de las pruebas

A continuación, pese a que las pruebas realizadas con el prototipo muestran que se han logrado los objetivos que se plantearon, se exponen algunos resultados y conclusiones que se han observado:

1. Se ha detectado cierto error en los datos medidos, sobre todo en el nivel de pH. Si bien, las pruebas iniciales que se realizaron para configurar dicho sensor parecían buenas, el hecho de que ahora el sensor este flotando en un medio que se mueve constantemente (oscilaciones del agua del depósito) puede estar causando estos errores.

2. Se ha podido comprobar que TTN no recibe todos los paquetes que envía el emisor. Esto parece deberse a que el emisor en la configuración de las librerías que se incluyen tiene configurado que se transmite en tres frecuencias: 868.1, 868.3 y 868.5 MHz, iterativamente, mientras que el gateway al ser de un canal y estar configurado a 868.1 MHz, solo recibe la trasmisión a esta frecuencia, por ello, solo es visible un paquete de cada tres en TTN. Esto se ha podido comprobar empleando el monitor serie del IDE de Arduino, donde observando todo el proceso del sketch y mediante la instrucción ”*Serial.println(LMIC.freq);*” es posible conocer la frecuencia de transmisión de cada paquete en LoRa.

3. También, se ha observado que TTN conoce el número de paquetes que el emisor ha enviado, por lo que es posible que exista un contador en la capa TCP que se actualiza con cada mensaje que realmente se recibe y esto es lo que permite saber el número de paquetes perdidos. Es decir, si el gateway solo recibe un paquete de cada tres, ¿cómo es posible que TTN realmente sepa que el número exacto de paquetes enviados?

4. TTN presenta un comportamiento no esperado, dado que solo muestra datos recibidos a partir del momento que se inicia sesión y se abre la web del dispositivo o del gateway. Por ello, no se tiene acceso a datos anteriores, siendo esto algo que no es muy práctico. Además, se visto que la conexión es inestable y cada poco tiempo (pocas horas) tiende a bloquearse, de forma que

al refrescar la página se pierden los datos que hasta ese momento se habían recibido.

Existen ciertos aspectos que se pueden mejorar, quizás cambiando de servidor o profundizando en el conocimiento del funcionamiento de TTN, pero queda fuera del alcance de este trabajo. Es posible que uno de los factores clave sea el hecho de no disponer de un gateway multicanal para la correcta recepción de todos los paquetes, pero creo que TTN seguiría comportándose de igual forma. Con todo, si las librerías que se incluyen en el sketch tuvieran configurada una única frecuencia de transmisión que coincidiera con la programada en el gateway, la recepción en TTN seria perfecta; pero estas librerías no se idearon para pasarelas de un único canal.

4.6 Presupuesto.

En la tabla 13 se presenta el presupuesto del prototipo construido, sin incluir tiempo de diseño del sistema, construcción e instalación del prototipo. Como puede verse, el **presupuesto material** sin contar pequeño material, asciende a **172 euros**, que para ser un pequeño proyecto académico presenta un coste considerable, incluso habiendo recurrido a productos que quizás no sean de la mejor calidad. Los costes de diseño, prototipado y pruebas son bastante elevados, debido al número de horas empleadas.

Concepto	***P. unitario [euros]***	***Cantidad***	***Total [euros]***
Arduino UNO	8	1	8
Sensor nivel	7	1	7
Sensor pH	15	1	15
TTGO 433 MHz	25	2	50
TTGO 868 MHz	27	1	27
Dragino LG01	65	1	65

TABLA 13. PRESUPUESTO

4.7 Resumen del capítulo.

En este capítulo, partiendo de la especificación de la ubicación donde se realizan las pruebas, se ha mostrado el prototipo del sistema diseñado junto con las pruebas realizadas y algunos comentarios sobre los resultados obtenidos. Por un lado, se ha construido un primer prototipo que emplea 433 MHz en el enlace LoRa punto a punto y se ha comprobado mediante pruebas que cumple con lo esperado. Por otro lado, se ha realizado una pequeña descripción del prototipo definitivo que se ha construido, de forma que se puedan emplear simultáneamente las bandas de 433 y 868 MHz y así poder acceder a TTN mediante LoRaWAN, manteniendo simultáneamente el enlace punto a punto. Pese a los condicionantes, se ha podido comprobar que el sistema funciona y el prototipo construido aunque es mejorable logra los objetivos planteados inicialmente. Para acabar, además del presupuesto material, se han expuesto algunas conclusiones que se han obtenido tras la realización de las pruebas, comentando aspectos relacionados con el sensor de pH, la perdida de paquetes en el enlace LoRaWAN y el comportamiento inesperado de TTN.

5. Mejora del sistema: alarmas SMS

En este capítulo se plantea lo necesario para dotar al sistema de la posibilidad de enviar alarmas mediante SMS empleando la red GSM/GPRS. Se ha considerado que se trata de una forma adecuada de acceder a información instantánea sobre un acontecimiento inesperado sin tener que estar monitorizando el sistema de forma continua.

5.1 Contexto y requisitos

Para empezar, el hecho de poder conocer de forma instantánea que se producido una situación anormal es muy interesante de cara a poder realizar las medidas correctoras con la mayor prontitud. En este caso, independientemente de la causa, si se detecta un descenso del nivel del agua por debajo de un límite prefijado o una desviación del pH fuera de los límites establecidos, es muy importante conocerlo con la mayor prontitud. Por tanto, dotar al sistema de la posibilidad de enviar una alarma en una situación de este tipo parece ser algo muy necesario. Para ello, partiendo de lo realizado en el TFG [39] y adaptándolo al caso actual, es factible implementar esta mejora, permitiendo al responsable del agua municipal estar informado inmediatamente de que algo no marcha bien, reduciendo los tiempos de intervención enormemente.

Por un lado, para implementar esta idea en el sistema se requiere la Tarjeta SIM900 GSM/GPRS compatible con Arduino, que puede verse en la figura 53 y que será la encargada de permitir al sistema enviar un SMS de alarma. Se trata de una tarjeta o módulo de expansión perfectamente compatible con cualquier tarjeta del universo Arduino, que emplea la red de telefonía móvil GSM/GPRS, por lo que requiere una tarjeta SIM de cualquiera de las operadoras con cobertura en la zona de aplicación del sistema.

FIGURA 53. TARJETA SIM900 GSM/GPRS [39]

Por otro lado, no se incluye más información sobre sus características y su funcionamiento, que puede ser consultado en los anexos, pero si hay que comentar que su conexión es muy sencilla. Únicamente se requiere emplear un puerto serie de Arduino y conectarlo con el correspondiente de la SIM 900, además de alimentarla y conectar las masas en común. Obviamente, es necesario incluir el código necesario para que se pueda emplear en el sistema, ya que la tarjeta de comunicaciones debe enviar un mensaje cuando sea necesario y, para ello, primero tiene que ser configurada y reconocida por Arduino. En este punto se debe aclarar que debe estar incluido en el código el número de móvil receptor de los mensajes, para que el responsable del servicio de aguas reciba el mensaje y no otra persona.

Finalmente, hay que decir que las pruebas realizadas para comprobar su funcionamiento también se incluyen en el anexo correspondiente, donde se realiza una explicación detallada de todo el proceso.

5.2 Implementación en el sistema y pruebas

En primer lugar, una vez que se ha comprobado que la tarjeta funciona y es capaz de enviar y recibir mensajes SMS, el siguiente paso es implementar esta mejora en el sistema. En la figura 54 se presenta el esquema hardware actualizado del prototipo. Entonces, en la figura 55 se puede ver que se ha conectado Arduino al módulo GSM empleando un puerto serie definido por software utilizando los pines dos y tres de Arduino. Así, es posible mantener los pines siete y ocho como puerto serie definido por software para comunicar Arduino con los dos transmisores LoRa que se ven en la

imagen. Tanto Arduino como la tarjeta GSM se alimentan a través de la fuente de tensión regulable que se aprecia en la imagen, siendo un sistema totalmente operativo sin necesidad de conectarlo a un ordenador. Solo se conecta para cargar el software o para monitorizar que ocurre mediante un emulador de puerto serie.

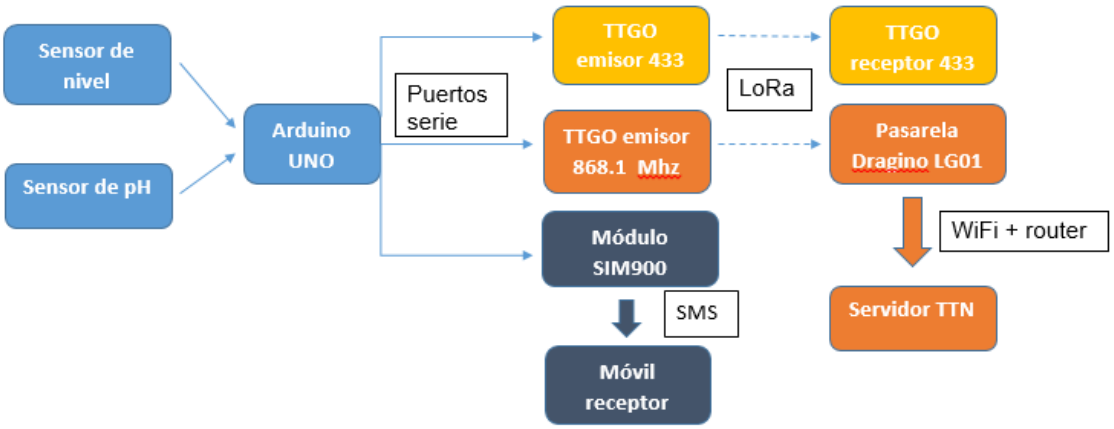


FIGURA 54.ESQUEMA HARDWARE CON GSM

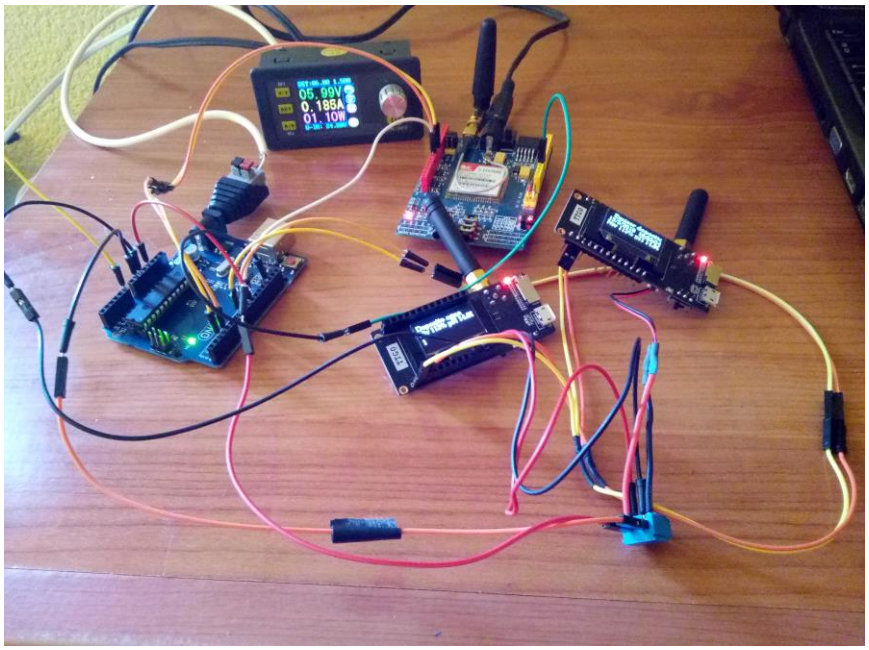


FIGURA 54. PROTOTIPO CON TARJETA GSM INCLUIDA

Después, tras definir todos los parámetros necesarios en el sketch, como por ejemplo el número de móvil que recibirá los mensajes de alarma, si se realizan algunas simulaciones básicas forzando la trasmisión, se obtiene lo que se muestra en la figura 54. De este modo, puede verse que el teléfono móvil destinatario de los mensajes de alarma recibe este tipo de mensajes cuando se da la situación que dispara la alarma. En este caso, se ha condicionado el envío del mensaje a que el sensor de nivel presentara una lectura inferior al 80%. Con todo, los parámetros límite que disparan las alarmas deben configurarse en el sketch de acuerdo a lo sugerido por los técnicos de aguas.

FIGURA 55. EJEMPLO DE SMS RECIBIDOS

Para terminar, hay que decir que no se han incluido los sensores en la figura 55, para mayor claridad; pero partiendo de lo que se presenta en la figura 44 solo se requiere conectar la tarjeta GSM como se ha comentado y cargar el nuevo sketch, que es una actualización del que se propuso en el capítulo de prototipo y pruebas. Además, es de destacar que esta mejora no supone un sobrecoste elevado, dado que el precio actual de este tipo de tarjetas es inferior a 20 euros, como puede verse en [40], por lo que presenta un balance coste/solución muy bueno.

5.3 Pruebas con el prototipo en el depósito

El prototipo que se probado en el depósito es que se ha presentado en la figura 54, junto con los sensores propuestos y que se ven en las figuras del capítulo 4, por ello no procede volver a presentar imágenes del mismo. Entonces, una vez que se encuentra funcionando, se ha forzado que el nivel del depósito disminuya, manipulando la apertura de las válvulas que abren la entrada de agua al depósito. Así se consigue una disminución lenta pero constante del nivel del agua, que puede verse en la figura 57, donde se ve como TTN muestra este descenso de nivel.

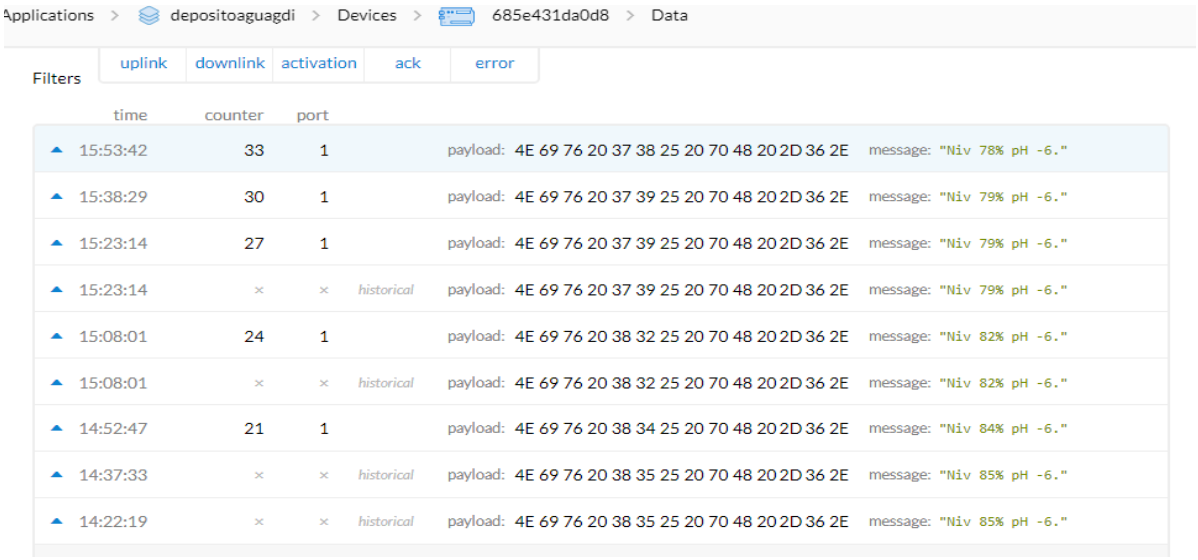


FIGURA 56. TTN - VACIADO DEL DEPÓSITO

Dado que en el sketch se ha programado el envío de SMS de alarma cuando el nivel disminuya por debajo del 75 % y por debajo del 50 %, se presentan en la figura 58 los mensajes recibidos por el móvil destinatario de dichas alarmas. Como puede verse se han recibido algunos mensajes en diversos momentos indicando las alarmas comentadas, siendo los periodos de envío acordes con los tiempos que se mostraron en TTN. Es decir, la velocidad de vaciado y los tiempos en los que se envían las alarmas son bastante coherentes. Se realizaron algunos *reset* del sistema para poder forzar medidas instantáneas, ya que los periodos de medición sino serían un poco largos.

FIGURA 57. SMS DE ALARMA RECIBIDOS

Una vez verificado que el sistema de alarmas funciona, se anula la intervención realizada para que se vaciara el depósito y este comienza a llenarse. Como puede verse en la figura 59, los datos recibidos por TTN muestran el proceso de llenado del depósito, que es bastante más rápido que el vaciado por consumo normal.

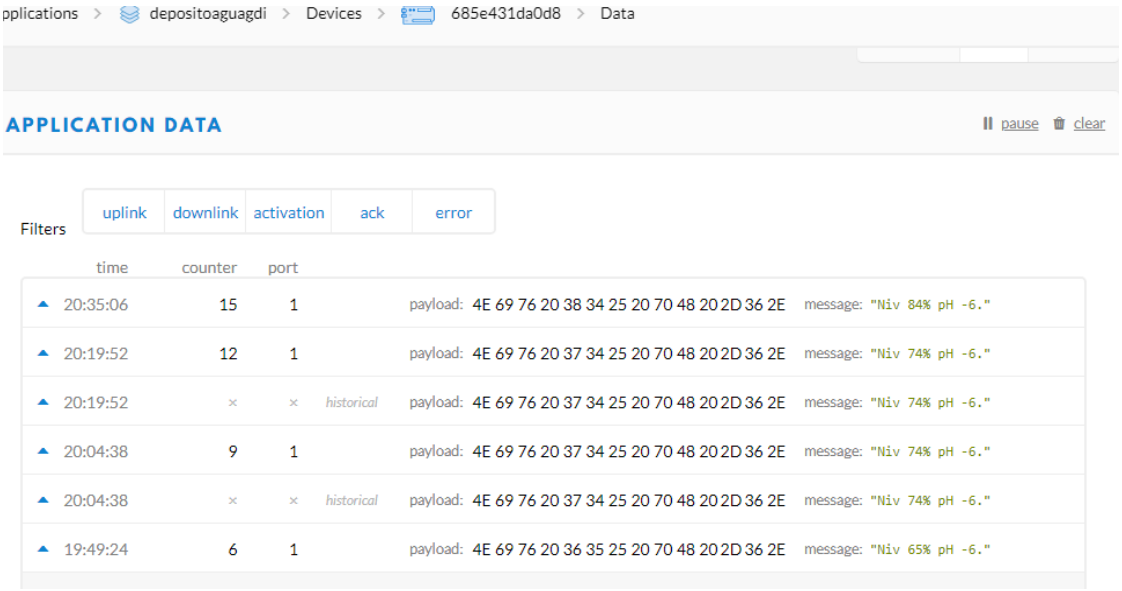


FIGURA 58. TTN - LLENADO DEL DEPÓSITO

5.4 Resumen del capítulo.

En este capítulo se ha propuesto y desarrollado una mejora del sistema que puede aportar una solución ante situaciones inesperadas que alteren los parámetros que este sistema monitoriza. Así, empleando la red GSM, se ha diseñado y probado que sea posible enviar un mensaje SMS de alarma para estar informado a la mayor brevedad, permitiendo una rápida intervención. Para ello, se ha presentado el módulo de comunicaciones y se ha explicado su integración el prototipo del capítulo cuatro, realizando las pruebas oportunas para verificar que realmente se envían y reciben los SMS de alarma, junto con las demostraciones de que TTN muestra los datos del proceso de vaciado y llenado (medidas del nivel de agua) Con todo, la mayor parte de los detalles no se han incluido en el capítulo, sino en los anexos, para evitar alargar la extensión de este documento. Por ello, para mayor información acerca de la tarjeta y diversas pruebas, es preferible recurrir al documento de anexos.

6. Conclusiones y líneas futuras

Una vez finalizado el presente trabajo, es momento de realizar una reflexión y poder obtener una serie de conclusiones que permitan asentar conocimientos y procedimientos. De este modo, se puede exponer lo siguiente:

1. Se ha logrado alcanzar el **objetivo fundamental** del proyecto: diseño de un **sistema de monitorización y transmisión de datos empleando LoRa**, dejando claro que el sistema cumple con las expectativas requeridas. Aun así, la parte de automatización se ha descartado, debido a que no se ha intervenido sobre los mecanismos que regulan la entrada de agua al depósito ni sobre la corrección del pH, aunque su implementación futura en el sistema es totalmente factible.

2. Se ha demostrado con todo tipo de simulaciones, capturas de pantalla y fotografías que **el proceso de diseño es correcto y el prototipo es funcional**. Esto conlleva una extensión de la memoria superior a lo necesario, dado que se ha tratado de detallar todos los pasos seguidos, por tratarse de un proyecto académico. Por ello, gran parte del código que se ha empleado se adjunta como anexos, junto a otras explicaciones.

3. **Se han alcanzado los hitos** según lo planteado en la planificación, sin surgir imprevistos con gran impacto. Incluso, ha sido posible ir más allá de lo inicialmente planteado, siendo posible evolucionar de un enlace LoRa punto a punto a subir los datos a TTN y a la mejora de los mensajes de alarma mediante la tarjeta GSM900. En este punto hay que comentar que la parte de comunicaciones ha requerido un esfuerzo mayor al esperado, sobre todo LoRaWAN.

4. Se ha alcanzado un **conocimiento inicial de software y hardware** bastante variado, que de no ser por la realización del presente proyecto no habría sido posible. Esta es una de las conclusiones más importantes que se pueden transmitir, ya que implica conocer nuevas herramientas de diseño y prototipado, para obtener cualquier tipo de solución electrónica **que incluya comunicaciones LoRa** y/o mensajes SMS.

5. Se ha conseguido **poner en práctica competencias comunicativas y la redacción de textos técnicos**, sin duda componentes muy importantes en el ámbito de la Ingeniería. Además, se ha experimentado con la elaboración de presentaciones y la edición de video.

6. Se ha manifestado la exigencia de una **buena planificación inicial**, de forma que esta sea consecuente con la dificultad del mismo, con la finalidad de evitar desvíos temporales importantes por falta de previsión. Esto, junto a un **esfuerzo superior al previsto**, ha permitido ir adelantado sobre los plazos previstos, disponiendo de tiempo para aumentar las posibilidades de comunicaciones del sistema.

7. Idealmente, se debería **emplear un gateway multicanal**, pero su precio era excesivo y no era el objetivo principal de este proyecto subir los datos a TTN. Esto ha ocasionado que haya sido necesario adquirir y configurar un gateway de un solo canal, con el problema de que las librerías que se emplean en los nodos finales configuran como mínimo tres frecuencias de transmisión cíclicas. Este hecho ha sido muy difícil de averiguar y corregir, al menos para mí, y da lugar a que en TTN solo se recibe un paquete de cada tres que envía el nodo.

9. La automatización de los **procesos de corrección de pH y apertura de válvulas de entrada de agua no se ha realizado**, debido a que, pese a ser una de las ideas iniciales y siendo el diseño más o menos asumible, la **implementación y pruebas sería muy complicada** por intervenir de forma activa sobre una instalación de dominio público con implicaciones sobre la salud de las personas. Además, tratándose de un proyecto perteneciente al área de Sistemas de comunicación, se ha dado prioridad a las comunicaciones del sistema: LoRa/LoRaWAN y GSM. Aun así, como ya se ha comentado, es posible intervenir sobre estos sistemas pero se requiere colaboración con técnicos especialistas en materia de calidad y control de agua.

Aparte de lo anterior, es necesario comentar que ha sido un proceso muy enriquecedor y muy entretenido. Esto me ha permitido aprender y practicar con el entorno Arduino, GSM y sobre todo con LoRa, algo inicialmente desconocido para mí y que presenta un potencial enorme. Poder construir los circuitos, desarrollar el código y

realizar las simulaciones es lo que hace que todo este proceso haya sido ameno; pero también hay que decir que la búsqueda de información y aprendizaje es muy exigente. Con todo, considero que realizar este proyecto ha sido una experiencia muy gratificante.

Por otro lado, en cuanto a **ampliaciones o futuras líneas de trabajo** se pueden enumerar las siguientes:

1. **Incluir más sensores** que permitan medir mayor variedad de parámetros de calidad del agua.

2. Incluir la parte de **automatización de mecanismos correctores**, siempre y cuando se cuente con la autorización pertinente y el asesoramiento de técnicos competentes en materia de calidad del agua.

3. **Mejorar** todo el contexto **LoRaWAN,** de forma que sea posible tener un servidor que presente más funcionalidades que las que se han conseguido, como por ejemplo elaborar estadísticas, gráficas, informes, etc.

4. **Diseñar un prototipo de circuito impreso** que incluya solo los elementos necesarios, todos ellos en una única tarjeta con los conectores necesarios: microcontrolador y sus accesorios para funcionar, chip LoRa, fuente de alimentación y conectores para sensores y actuadores.

En resumen, pese a que se pueden considerar diversas mejoras para optimizar el sistema, se considera más que suficiente todo lo desarrollado, porque se han puesto de manifiesto técnicas de programación, diseño y simulaciones con software, prototipado y simulaciones reales. Aun así, no es descartable implementar alguna de estas mejoras como reto personal en el futuro.

Rubén Adrián de la Cámara En Gumiel de Izán a 12/06/2019.

Glosario

ABP Activation by Personalization, en TTN

FSK Modulación por desplazamiento de frecuencia.

GSM Global System for Mobile

GPRS General Packet Radio Service

IoT Internet de las cosas.

ISM Espectro de radio sin licencia en la banda Industrial, Científica y Médica

LoRa Long Range, tecnología de modulación radio propiedad de Semtech.

LoRaWAN Red de área extendida basada en LoRa, protocolos y especificaciones.

LPWAN Red de área extendida de baja potencia.

OTAA Over-The-Air Activation, en TTN

pH Potencial de Hidrógeno.

RSSI Indicador de la fuerza de la señal recibida, en dBm.

SF Spreading factor, número de bits por símbolo.

SoC System on Chip

ZigBee Estándar de comunicaciones 802.15.4.

Bibliografía

- [1]. Real Decreto 140/2003, de 7 de febrero, por el que se establecen los criterios sanitarios de la calidad del agua de consumo humano. [Sede web]. [Acceso: 25/02/2019]. Disponible en: https://www.boe.es/eli/es/rd/2003/02/07/140/con
- [2]. SINAC. Sistema de Información Nacional de Aguas de Consumo. [Sede web]. [Acceso: 25/02/2019]. Disponible en: http://sinac.msssi.es/CiudadanoWeb/ciudadano/informacionAbastecimientoActionEntrada.do
- [3]. Web del ayuntamiento de Gumiel de Izan. [Acceso: 01/03/2019]. Disponible en: http://www.gumieldeizan.es/
- [4]. Parámetros de control del agua potable. [Sede web]. [Acceso: 05/03/19]. Disponible en: https://www.iagua.es/blogs/beatriz-pradillo/parametros-control-agua-potable
- [5]. GUÍA BÁSICA de control de calidad de agua. [PDF]. [Acceso: 05/03/19]. Disponible en: https://www.ongawa.org/wp-content/uploads/2015/09/Agua-CAS-revisar2.pdf
- [6]. USO DE LEJÍA PARA DESINFECTAR EL AGUA DE CONSUMO DOMÉSTICO. [PDF]. [Acceso: 05/03/19]. Disponible en: https://www.navarra.es/NR/rdonlyres/505A533D-EFA8-4517-990F-900A277C102C/146271/09DocUSODELEJIAdesinfeccinaguaconsumo.pdf
- [7]. Los trihalometanos y la cloración del agua. [Sede web]. [Acceso: 05/03/19]. Disponible en: http://hispagua.cedex.es/node/61343
- [8]. Mejora en la calidad del agua suministrada en los municipios de la provincia de Córdoba mediante el empleo de dióxido de cloro. Sede web]. [Acceso: 05/03/19]. Disponible en: https://www.tecnoaqua.es/articulos/20170919/mejora-calidad-agua-suministrada-municipios-cordoba-empleo-dioxido-cloro
- [9]. Monitorización del Agua Potable. [Sede web]. [Acceso: 07/03/19]. Disponible en: https://www.s-can.at/es/aplicaciones/agua-potable

- [10]. Salazar Vilañez M.D. (2018). Unidad de adquisición y transferencia de variables remotas para el monitoreo en tanques de agua potable. (Grado). Disponible en: http://repositorio.utn.edu.ec/bitstream/123456789/8044/1/04%20MEC%20221%20TRABAJO%20DE%20GRADO.pdf
- [11]. Huayata Sucasaca, J.C y Suaña Humpire, E.W. (2017). Diseño e implementación de un sistema scada para el control del nivel de agua para uso domótico mediante redes industriales. (Tesis). Disponible en: http://repositorio.unap.edu.pe/bitstream/handle/UNAP/4185/Huayta_Sucasaca_Jose_Carlos_Suaña_Humpire_Elmer_Wilson.pdf?sequence=1
- [12]. Estrada Luna, V y Gutierrez Vázquez V. F. (2012). Control y monitoreo de una potabilizadora de agua por medio de una red Controlnet. (Tesis). Disponible en: https://tesis.ipn.mx/bitstream/handle/123456789/10512/61.pdf?sequence=1
- [13]. Paraira Faus, M; Martín Alonso, J; González Banco, S y Puigdomènech Serra, C. (2016). Creación de una plataforma para la validación de sensores on line para el control de la calidad del agua. Disponible en: https://www.tecnoaqua.es/media/uploads/noticias/documentos/articulo-tecnico-creacion-plataforma-validacion-sensores-on-line-control-calidad-agua-tecnoaqua-es.pdf
- [14]. Martínez Ferrer, J. (2016). Adaptación e implementación de un módulo de comunicación GSM con un servidor para la gestión de datos y alarmas integrado a un sistema de monitorización de calidad de agua. Disponible en: http://oa.upm.es/44668/1/PFC_JULIA_MARTINEZ_FERRER_2016.pdf
- [15]. HYDROPORT Acceso al Internet del agua. [Sede web]. [Acceso: 08/03/19]. Disponible en: https://www.vonroll-hydro.ch/es/hydroport.html
- [16]. Design of low-cost autonomous water quality monitoring system. [Sede web]. [Acceso: 08/03/19]. Disponible en: https://ieeexplore.ieee.org/abstract/document/6637139/authors#authors
- [17]. The Open Water Project. [Sede web]. [Acceso: 08/03/2019]. Disponible en: https://publiclab.org/wiki/open-water#The+Open+Water+Project
- [18]. MÃE D'ÁGUA. [Sede web]. [Acceso: 08/03/2019]. Disponible en: https://rede.infoamazonia.org/mae-dagua/

- [19]. Córdoba Peñalver, E.J. (2017). Análisis y diseño de una red de sensores en un parque natural (Grado). Disponible en: http://hdl.handle.net/10609/59925
- [20]. Delgado Bajo, A. (2019). Sistema de monitorización de flujo ciudadano mediante el uso de WiFi y LoRaWAN (Master). Disponible en: http://hdl.handle.net/10609/89725
- [21]. Arduino. [Sede web]. [Acceso: 09/03/19]. Disponible en: https://es.wikipedia.org/wiki/Arduino
- [22]. Entornos de Aplicación Arduino. [Sede web]. [Acceso: 09/03/19]. Disponible en: https://aprendiendoarduino.wordpress.com/2016/06/26/entornos-de-aplicacion-arduino/
- [23]. Arduino Products. [Sede web]. [Acceso: 09/03/2019]. Disponible en: https://www.arduino.cc/en/main/products
- [24]. Libraries. [Sede web]. [Acceso: 09/03/19]. Disponible en: https://www.arduino.cc/en/Reference/Libraries
- [25]. Arduino en Español - Wiki Comunitaria. [Sede web]. [Acceso: 09/03/19]. Disponible en: https://playground.arduino.cc/Es/Es
- [26]. What is LoRa? [Sede web]. [Acceso: 10/03/2019]. Disponible en: https://www.semtech.com/lora/what-is-lora
- [27]. Ordoñez Monfort, I. (2017). Estudio de la arquitectura y el nivel de desarrollo de la red LoRaWAN y de los dispositivos LoRa. Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/64365/6/iordonezTFM0617memòria.pdf
- [28]. De Prado Liceaga, M. (2017). Diseño y Despliegue de arquitectura para la recogida y presentación de medidas de sensores IoT (Master). Disponible en: https://addi.ehu.es/bitstream/handle/10810/23420/Memoria.pdf?sequence=1
- [29]. Rodriguez Munca, J. D. (2016). Dispositivo LoRa de comunicación al largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo. (Master). Disponible en : http://oa.upm.es/44890/1/TFM_JOSE_DANIEL_RODRIGUEZ_MUNCA.pdf
- [30]. LoRaWAN. [Sede web]. [Acceso: 12/03/19]. Disponible en: https://lorawan.es/

- [31]. APLICACIONES SENSORES LORAWAN. [Sede web]. [Acceso: 12/03/19]. Disponible en: https://www.catsensors.com/es/lorawan/aplicaciones-sensores-lorawan
- [32]. IOT STARTER KIT CON TECNOLOGÍA LORA. [Sede web]. [Acceso: 12/03/19]. Disponible en: https://www.matrix.es/product/wireless-iot/multiconnect-conduit-iot-starter-kit
- [33]. MEDIR DISTANCIA CON ARDUINO Y SENSOR DE ULTRASONIDOS HC-SR04. [Sede web]. [Acceso: 12/03/19]. Disponible en: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/
- [34]. Disponible en: https://es.aliexpress.com/item/1Set-Liquid-PH-Value-Detection-Regulator-Sensor-Module-Monitoring-Control-Meter-Tester-BNC-PH-Electrode-Probe/32803578131.html
- [35]. ESP32, EL "HERMANO MAYOR" DEL ESP8266 CON WIFI Y BLUETOOTH [Sede web]. [Acceso: 25/03/19]. Disponible en: https://www.luisllamas.es/esp32/
- [36]. ESP32 LoRa: You Can Reach Up to 6.5 Km! [Sede web]. [Acceso: 25/03/19]. Disponible en: https://www.instructables.com/id/ESP32-LoRa-You-Can-Reach-Up-to-65-Km/
- [37]. Building a global open LoRaWAN™ network. [Sede web]. [Acceso: 20/04/2019]. Disponible en: https://www.thethingsnetwork.org
- [38]. How to: Storage integration. [Acceso: 14/05/2019]. Disponible en: https://www.youtube.com/watch?v=kVf8GmCbOuE&list=PLM8eOeiKY7JVwrBYRHxsf9p0VM_dVapXl&index=3
- [39]. Adrian de la Camara, R. Arduino + módulo GSM/GPRS: monitorización, automatización y gestión remota en un viñedo [2017]. Disponible en : http://hdl.handle.net/10609/65345
- [40]. SIM900 GPRS/GSM Shield Development Board Quad-Band Module with Antenna. [Acceso: 28/05/209]. Disponible en: https://www.ebay.es/i/152602473101?chn=ps
- [41]. SENSOR DE CLORO Y PH. [Acceso: 28/05/2019]. Disponible en: https://lualtec.es/sensor-de-cloro-y-ph.html?gclid=EAIaIQobChMIr_WDxdTI4gIVbRHTCh1EAwIDEAQYCSABEgIvy_D_BwE

- [42]. SENSOR PARA MEDIR LA TURBIDEZ DEL AGUA. [Acceso: 28/05/2019]. Disponible en: https://lualtec.es/sensor-para-medir-la-turbidez-del-agua.html?gclid=EAIaIQobChMIh8etldXI4gIVggvTCh3Zcw8pEAQYAiABEgKsPPD_BwE
- [43]. Libelium revoluciona el mercado de la calidad del agua con su nueva plataforma de monitorización Smart Water Xtreme. [Acceso: 28/05/2019]. Disponible en: http://www.interempresas.net/Agua/Articulos/231670-Libelium-revoluciona-mercado-calidad-agua-plataforma-monitorizacion-Smart-Water-Xtreme.html
- [44]. REYAX RYLR896 Lora module SX1276 UART 868MHz 915MHz Antenna AT command FCC NCC. [Acceso: 28/05/2019]. Disponible en: https://www.ebay.com/itm/REYAX-RYLR896-Lora-module-SX1276-UART-868MHz-915MHz-Antenna-AT-command-FCC-NCC-/181562403752
- [45]. 32-868T20D User Manual. [Acceso: 28/05/2019]. Disponible en: http://www.ebyte.com/en/downpdf.aspx?id=132
- [46]. Lora Shield. [Acceso: 28/05/2019]. Disponible en: http://wiki.dragino.com/index.php?title=Lora_Shield
- [47]. TTGO LoRa32 V2.1_1.6 Version 433/868/915Mhz ESP32 LoRa OLED 0.96 Inch SD Card Bluetooth WIFI Wireless Module ESP-32 SMA. [Acceso: 28/05/2019]. Disponible en: https://es.aliexpress.com/item/32872078587.html?spm=a2g0o.detail.1000016.1.6e406ee1zwaQwP&isOrigTitle=true
- [48]. Módulo de sensor de detección del valor del pH en líquidos y sonda del electrodo del PH para Arduino. [Acceso: 28/05/2019]. Disponible en: https://es.aliexpress.com/item/32803578131.html?src=google&albslr=227527957&albch=shopping&acnt=494-037-6276&isdl=y&slnk=&plac=&mtctp=&albbt=Google_7_shopping&aff_platform=google&aff_short_key=UneMJZVf&albagn=888888&albcp=1633820309&albag=63890294393&trgt=743612850714&crea=es32803578131&netw=u&device=c&gclid=EAIaIQobChMIkZy4iO_I4gIViTLTCh0h8AHzEAYYAiABEgLnU_D_BwE&gclsrc=aw.ds
- [49]. THE THINGS NETWORK. [Acceso: 28/05/2019]. Disponible en: https://www.thethingsnetwork.org/country/spain/

- [50]. Network Architecture. [Acceso: 28/05/2019]. Disponible en: https://www.thethingsnetwork.org/docs/network/architecture.html
- [51]. LG01-S IoT Gateway featuring LoRa® technology. [Acceso: 28/05/2019]. Disponible en: https://www.dragino.com/products/lora/item/119-lg01-s.html

Anexos

1. Código Arduino para comprobar el sensor de nivel.

```
const int EchoPin = 5;
const int TriggerPin = 6;

void setup() {
     Serial.begin(9600);
     pinMode(TriggerPin, OUTPUT);
     pinMode(EchoPin, INPUT);
}

void loop() {
   int cm = ping(TriggerPin, EchoPin);
   Serial.print("Distancia: ");
   Serial.println(cm);
   delay(1000);
}

int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;
   digitalWrite(TriggerPin, LOW);
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);
   duration = pulseIn(EchoPin, HIGH);    //tiempo entre pulsos, en
microsegundos
   distanceCm = duration * 10 / 292/ 2;    // obtener distancia en cm
     return distanceCm;
}
```

2. Código Arduino para comprobar el funcionamiento del sensor de pH.

```
#define SensorPin 0              //pH meter Analog output to Arduino
Analog Input 0
unsigned long int avgValue;   //Store the average value of the sensor
feedback
float b;
int buf[10],temp;

void setup()
{
   pinMode(13,OUTPUT);
   Serial.begin(9600);
   Serial.println("Ready");    //Test the serial monitor
}
void loop()
{
  for(int i=0;i<10;i++)       //Get 10 sample value from the sensor for
smooth the value
   {
   buf[i]=analogRead(SensorPin);
   delay(10);
   }
  for(int i=0;i<9;i++)        //sort the analog from small to large
   {
     for(int j=i+1;j<10;j++)
     {
     if(buf[i]>buf[j])
       {
   temp=buf[i];
   buf[i]=buf[j];
   buf[j]=temp;
       }
     }
```

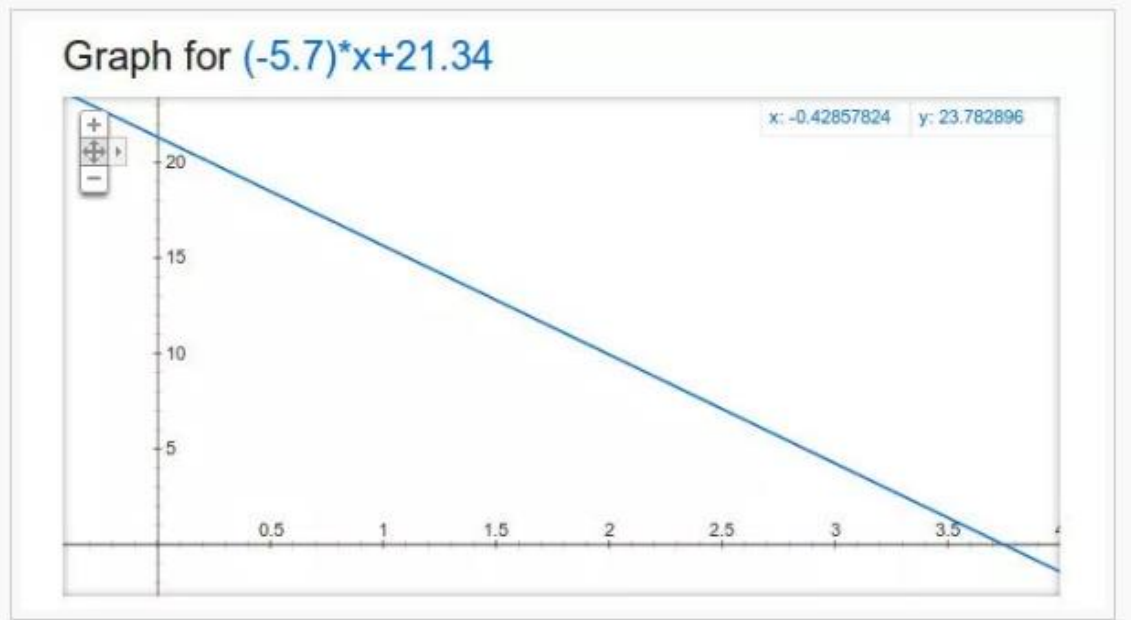
```
  }
  avgValue=0;
  for(int i=2;i<8;i++)      //take the average value of 6 center sample
    avgValue+=buf[i];
  float phValue=(float)avgValue*5.0/1024/6; //convert the analog into
millivolt
  phValue=3.5*phValue;          //convert the millivolt into pH value
  Serial.print("    pH:");
   Serial.print(phValue,2);
   Serial.println(" ");
   digitalWrite(13, HIGH);
   delay(800);
   digitalWrite(13, LOW);

}
```

3. Instrucciones para calibrar el sensor de pH.

Como se puede observar, en el circuito hay dos potenciómetros: el que está más cerca al conector BNC de la sonda es el que regula el offset y el otro es el del límite de pH.

- **Offset:** el rango de media de la sonda oscila entre valores negativos y positivos. El 0 representa un pH de 7.0. Para poder utilizarlo con Arduino este circuito añade un valor de offset al valor medido por la sonda, de esta forma el ADC solo tendrá que tomar muestras de valores positivos de tensión. Por lo tanto forzaremos un pH de 7.0 desconectando la sonda del circuito y cortocircuitando la parte interna del conector BNC con la exterior. Con un multímetro medimos el valor del pin Po y ajustamos el potenciómetro para que sea 2.5V.
- **Límite de pH**: Este potenciómetro es para establecer una valor del límite del circuito sensor de pH que hace que el LED rojo se encienda y la señal de pin Do se ponga en ON.

Además hay que calcular la conversión del voltaje que dará el sensor de pH, para lo que se necesitan dos valores de referencia de pH y medir el voltaje que devuelve el sensor en el pin Po. Lo más recomendable es utilizar una solución de calibración en sobre, también las hay en líquido pero es más fácil conservar las de sobre. Estas soluciones se venden en diferentes valores aunque los más comunes son pH 4.01, pH 6.86 y pH 9.18.



*— Gráfica de la ecuación de conversión entre voltaje y pH medido. y= -5.70 * x + 21.34*

FIGURA 59. CURVA DE CALIBRACION DEL SENSOR DE PH

Utilizando los sobres con pH 4.01 y pH 6.86 obtenemos los voltajes en el pin Po 3.04V y 2.54V respectivamente. El sensor es lineal por lo que tomando dos puntos podemos deducir la ecuación para convertir el voltaje medido a pH. La fórmula general sería y=mx+b, por lo que se tiene que calcular m y b, ya que x sería el voltaje e y el pH. El resultado es y=-5.70x+21.34.

4. Código para comprobar el enlace LoRa punto a punto (emisor).

```
  #include <SPI.h>
  #include <LoRa.h>
  #include <U8g2lib.h>
  #include <U8x8lib.h>

  #define OFF 0
  #define ON 1

#define LORA_SCK 5        // GPIO5 - SX1276 SCK
#define LORA_MISO 19     // GPIO19 - SX1276 MISO
#define LORA_MOSI 27    // GPIO27 -  SX1276 MOSI
#define LORA_CS 18     // GPIO18 -   SX1276 CS
#define LORA_RST 14   // GPIO14 -    SX1276 RST
#define LORA_IRQ 26  // GPIO26 -     SX1276 IRQ (interrupt request)

#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16

#include <SPI.h>
#include <LoRa.h>
#include <U8g2lib.h>

#define OFF 0
#define ON 1

  // SPI LoRa Radio
#define LORA_SCK 5        // GPIO5 - SX1276 SCK
#define LORA_MISO 19     // GPIO19 - SX1276 MISO
#define LORA_MOSI 27    // GPIO27 -  SX1276 MOSI
#define LORA_CS 18     // GPIO18 -   SX1276 CS
#define LORA_RST 14   // GPIO14 -    SX1276 RST
#define LORA_IRQ 26  // GPIO26 -     SX1276 IRQ (interrupt request)
```

```
  // I2C OLED Display works with SSD1306 driver
  #define OLED_SDA 21
  #define OLED_SCL 22
  #define OLED_RST 16

/* Pick One. Hardware I2C does NOT work! This article helped: https://robotzero.one/heltec-wifi-kit-32/
  TTGo boards similar to Heltec boards, LED_BUILTIN = 2 instead of pin 25
  Some OLED displays don't handle ACK correctly so SW I2C works better. Thank you Olikraus!
  TTGo OLED has pin 16 reset unlike other OLED displays
*/

// UNCOMMENT one of the constructor lines below
//U8X8_SSD1306_128X64_NONAME_SW_I2C Display(/* clock=*/ OLED_SCL, /* data=*/ OLED_SDA, /* reset=*/ OLED_RST); // Unbuffered, basic graphics, software I2C
//U8G2_SSD1306_128X64_NONAME_1_SW_I2C Display(U8G2_R0, /* clock=*/ OLED_SCL, /* data=*/ OLED_SDA, /* reset=*/ OLED_RST); // Page buffer, SW I2C
U8G2_SSD1306_128X64_NONAME_F_SW_I2C Display(U8G2_R0, /* clock=*/ OLED_SCL, /* data=*/ OLED_SDA, /* reset=*/ OLED_RST); // Full framebuffer, SW I2C
  //const int blueLED = LED_BUILTIN;
  int counter = 0;

void setup() {
  Serial.begin(115200);
  while (!Serial);

  Serial.println("LoRa Sender");
  Display.begin();
  Display.enableUTF8Print();    // enable UTF8 support for the Arduino print() function
```

```
  Display.setFont(u8g2_font_ncenB10_tr);
  // Very important for SPI pin configuration!
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);
  // Very important for LoRa Radio pin configuration!
  LoRa.setPins(LORA_CS, LORA_RST, LORA_IRQ);
  pinMode(25, OUTPUT); // For LED feedback
  if (!LoRa.begin(433E6)) {
   Serial.println("Starting LoRa failed!");
   while (1);
   }
   LoRa.setSpreadingFactor(12);
   LoRa.setTxPower(14, PA_OUTPUT_RFO_PIN);
   }
void loop() {
   Serial.print("Sending packet: ");
   Serial.println(counter);

   digitalWrite(25, ON);
   // send packet
   LoRa.beginPacket();
   LoRa.print("HeLoRa! ");
   LoRa.print(counter);
   LoRa.endPacket();
   digitalWrite(25, OFF);

   // Display Info
   Display.clearBuffer();
   Display.setCursor(0,12); Display.print("LoRa Sender");
   Display.setCursor(0,30); Display.print("Sent Packet:");
   Display.setCursor(0,48); Display.print(" # " + (String)counter);
   Display.sendBuffer();
  counter++;
  delay(5000);
}
```

5. Código para comprobar el enlace LoRa punto a punto (receptor).

```
#include <SPI.h>
#include <LoRa.h>
#include <U8g2lib.h>

#define OFF 0
#define ON 1

#define LORA_SCK 5        // GPIO5 - SX1276 SCK
#define LORA_MISO 19      // GPIO19 - SX1276 MISO
#define LORA_MOSI 27      // GPIO27 - SX1276 MOSI
#define LORA_CS 18        // GPIO18 - SX1276 CS
#define LORA_RST 14       // GPIO14 - SX1276 RST
#define LORA_IRQ 26       // GPIO26 - SX1276 IRQ (interrupt request)

#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16

U8G2_SSD1306_128X64_NONAME_F_SW_I2C Display(U8G2_R0, /* clock=*/ OLED_SCL, /* data=*/ OLED_SDA, /* reset=*/ OLED_RST); // Full framebuffer, SW I2C

  String rssi = "";
  String packet = "";

void setup() {

  Serial.begin(115200);
  while (!Serial);

 Serial.println("LoRa Receiver");

  Display.begin();
 Display.enableUTF8Print();
Display.setFont(u8g2_font_ncenB10_tr);
```

```
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);

  LoRa.setPins(LORA_CS, LORA_RST, LORA_IRQ);
  pinMode(25, OUTPUT); // For LED feedback
  if (!LoRa.begin(433E6)) {
   Serial.println("Starting LoRa failed!");
   while (1);
  }
  LoRa.setSpreadingFactor(12);
}

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");
  digitalWrite(25, ON);
  packet = "";
  while (LoRa.available()) {
    packet += (char)LoRa.read();    }
    rssi = LoRa.packetRssi();
  // Display Info
  Display.clearBuffer();
  Display.setCursor(0,12); Display.print("LoRa Receiver");
  Display.setCursor(0,26); Display.print("Received packet:");
  Display.setCursor(0,42); Display.print("    '" + packet + "'");
  Display.setCursor(0,58); Display.print("RSSI " + rssi);
  Display.sendBuffer();
    digitalWrite(25, OFF); // Turn blue LED off
    Serial.println(packet + "' with RSSI " + rssi);
  }
}
```

6. Sketch monitorización y automatización (tarjeta arduino).

Este sketch se emplea para comprobar que los sensores funcionan y se pueden tomar mediadas de nivel y pH, asi como pasar la información a un emisor LoRa (TTGO) por un puerto serie definido por software.

```
#include <SoftwareSerial.h>
SoftwareSerial TTGO(7, 8); // Comunicacion serie con el TTGO emisor
String mensaje=""; // para pasar parámetros al TTGO por puerto serie


// para el sensor de nivel
const int EchoPin = 5;  // pin 5
const int TriggerPin = 6; // pin 6
int separacion = 0; // (cms)distancia del sensor al nivel  lleno del
deposito

// para el sensor de pH
const int analogInPin = A0; // pin A0
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;

void setup() {
   TTGO.begin(19200);
   delay(100);
   Serial.begin(19200);
   Serial.println(F("Configurando el sistema (setup)........."));
   pinMode(TriggerPin, OUTPUT);
   pinMode(EchoPin, INPUT);
   pinMode(pulso, OUTPUT); // pin 4, acitvacion modulo GSM
   Serial.println("Iniciando...");
   SIM900.begin(19200); //
   Serial.println(F("Configuracion completada......"));
   delay(1000);
}
```

```
//  Función que calcula el nivel del depósito
int nivel(){
   int distAgua = ping(TriggerPin, EchoPin);
   long cotaCero = 400 + separacion; // deposito lleno + separacion al
sensor
   int nivel = (cotaCero - distAgua)*100/cotaCero;
   return nivel;

  }

//Función que realiza el disparo del sensor de distancia
 int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;
   digitalWrite(TriggerPin, LOW);   //para generar un pulso limpio
ponemos a LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);   //generamos Trigger (disparo) de
10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);

   duration = pulseIn(EchoPin, HIGH);   //medimos el tiempo entre
pulsos, en microsegundos

   distanceCm = duration * 10 / 292/ 2;//convertimos a distancia, en
cm

   Serial.println("Distancia: " + distanceCm);
   return distanceCm;
}

// funcion para medir el pH
  float medirPH(){
    for(int i=0;i<10;i++)
 {
  buf[i]=analogRead(analogInPin);
  delay(10);
```

```
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
  {
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
  }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;
 float phValue = -5.70 * pHVol + 21.76;
 return phValue;
 }

 //Funcion para enviar la informacion al TTGO
 void sendToTTGO(int nivel, float ph){
   TTGO.print("Niv ");
   TTGO.print(nivel);
   TTGO.print("%");
   TTGO.print(" pH ");
   TTGO.println(ph);
  }

void loop() {
   Serial.println("Sensores midiendo...");
   int niv = nivel();
   float pH = medirPH();

   mensaje= (char)niv  +(char)niv;
   Serial.print("Nivel del depósito: ");
   Serial.print(niv);
```

```
   Serial.println(" %");
   Serial.print("ph del agua: ");
   Serial.println(pH);

   //Enviar la info al TTGO emisor
    sendToTTGO(niv, pH);

    //TTGO.print(mensaje ); // mensaje a enviar
    Serial.println("Mensaje enviado).......");
    delay(1000);

   delay(300000); // reposo entre mediciones
}
```

7. Sketch del TTGO emisor en enlace punto a punto.

```
#include <SPI.h>
#include <LoRa.h>          // https://github.com/sandeepmistry/arduino-LoRa
#include <U8g2lib.h>   // https://github.com/olikraus/U8g2_Arduino

#define OFF 0   // For LED
#define ON 1

//// SPI LoRa Radio
#define LORA_SCK 5        // GPIO5 - SX1276 SCK
#define LORA_MISO 19     // GPIO19 - SX1276 MISO
#define LORA_MOSI 27    // GPIO27 - SX1276 MOSI
#define LORA_CS 18     // GPIO18 - SX1276 CS
#define LORA_RST 14   // GPIO14 - SX1276 RST
#define LORA_IRQ 26  // GPIO26 - SX1276 IRQ (interrupt request)

// I2C OLED Display works with SSD1306 driver
#define OLED_SDA 21
#define OLED_SCL 22
```

```
#define OLED_RST 16


U8G2_SSD1306_128X64_NONAME_F_SW_I2C Display(U8G2_R0, /* clock=*/
OLED_SCL, /* data=*/ OLED_SDA, /* reset=*/ OLED_RST); // Full
framebuffer, SW I2C

String mensaje=" Esperando..."; // Para recibir datos del puerto serie
String texto;
char inchar=0;

void setup() {
  pinMode(25, OUTPUT); // pin 25, datos seri recibidos
  Serial.begin(19200);
  Serial.println("Iniciando...");
  delay(1000);
  Display.begin();
  Display.enableUTF8Print();    // enable UTF8 support for the Arduino
print() function
  Display.setFont(u8g2_font_ncenB10_tr);

  // Very important for SPI pin configuration!
 SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);

  // Very important for LoRa Radio pin configuration!
  LoRa.setPins(LORA_CS, LORA_RST, LORA_IRQ);

  if (!LoRa.begin(433E6)) {
  Serial.println("Starting LoRa failed!");
  while (1);
  }

    LoRa.setSpreadingFactor(12);
    LoRa.setTxPower(17,PA_OUTPUT_PA_BOOST_PIN );
}


void loop() {
```

```
  digitalWrite(25, LOW);
delay(500);

texto= mensaje;
  if (Serial.available() >0){
          mensaje="";
          digitalWrite(25, HIGH);

       while(Serial.available() >0)// Se lee el texto del SMS entrante
y se almacena en un string
       {
       inchar=Serial.read();
       if(inchar=='/') {break;}// Si llega caracter vacio se sale
       mensaje= mensaje+inchar; //String que almacena el texto recibido
despues de la '@'

      }
  }
  if(mensaje != " "){
 texto= mensaje;
  }
// para enviar el mensaje al receptor
  LoRa.beginPacket();
  LoRa.print(mensaje);
  LoRa.endPacket();

 Display.clearBuffer();
    Display.setCursor(10,12); Display.print("Emisor LoRa ");
    Display.setCursor(0,42); Display.print(texto);
    Display.sendBuffer();

}
```

8. Sketch del TTGO receptor en enlace punto a punto.

```
#include <SPI.h>
#include <LoRa.h>          // https://github.com/sandeepmistry/arduino-LoRa
#include <U8g2lib.h>    // https://github.com/olikraus/U8g2_Arduino
// #include <U8x8lib.h>

#define OFF 0    // For LED
#define ON 1

// SPI LoRa Radio
#define LORA_SCK 5         // GPIO5 - SX1276 SCK
#define LORA_MISO 19      // GPIO19 - SX1276 MISO
#define LORA_MOSI 27     // GPIO27 - SX1276 MOSI
#define LORA_CS 18      // GPIO18 - SX1276 CS
#define LORA_RST 14    // GPIO14 - SX1276 RST
#define LORA_IRQ 26   // GPIO26 - SX1276 IRQ (interrupt request)

// I2C OLED Display works with SSD1306 driver
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16

U8G2_SSD1306_128X64_NONAME_F_SW_I2C    Display(U8G2_R0,    /*    clock=*/ OLED_SCL,   /*   data=*/   OLED_SDA,   /*   reset=*/   OLED_RST);   //   Full framebuffer, SW I2C

String rssi = ""; // niveel de señal recibida
String packet = "";

void setup() {
  Serial.begin(115200);
  while (!Serial);

  Serial.println("LoRa Receiver");
```

```
  Display.begin();
  Display.enableUTF8Print();     // enable UTF8 support for the Arduino
print() function
  Display.setFont(u8g2_font_ncenB10_tr);

  // Very important for SPI pin configuration!
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);

  // Very important for LoRa Radio pin configuration!
  LoRa.setPins(LORA_CS, LORA_RST, LORA_IRQ);

  pinMode(25, OUTPUT); // For LED feedback

  if (!LoRa.begin(433E6)) { // frecuencia de operacion (433-470 MHz)
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  // The larger the spreading factor the greater the range but slower
data rate
  // Send and receive radios need to be set the same
  LoRa.setSpreadingFactor(12);  // ranges from 6-12, default 7 see API
docs
}

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet ");
    digitalWrite(25, ON);  // Turn blue LED on

    // read packet
    packet = "";                    // Clear packet
    while (LoRa.available()) {
      packet += (char)LoRa.read(); // Assemble new packet
    }
    rssi = LoRa.packetRssi();
```

```
    // Display Info
    Display.clearBuffer();
    Display.setCursor(0,12); Display.print("LoRa Receiver");

    Display.setCursor(0,26); Display.print("Paquete recibido N:");
    Display.setCursor(0,42); Display.print(packet );
    //Display.setCursor(0,42); Display.print( packet);
    Display.setCursor(0,58); Display.print("RSSI " + rssi);
    Display.sendBuffer();

    digitalWrite(25, OFF); // Turn blue LED off

    Serial.println(packet + " with RSSI " + rssi);
  }
}
```

9. TTN. Registro de nodos y pasarelas.

Se exponen a continuación todos los pasos para registrar un dispositivo final y un gateway en TTN.

9.1 Registro de un nodo final.

1. Se crea una cuenta de usuario en TTN, siguiendo los pasos habituales.
2. Desde *Console* de la cuenta creada se puede ir a *Application*s o *Gateways* , bien para registrar un nodo y crear aplicaciones o para añadir una nueva pasarela a la red.

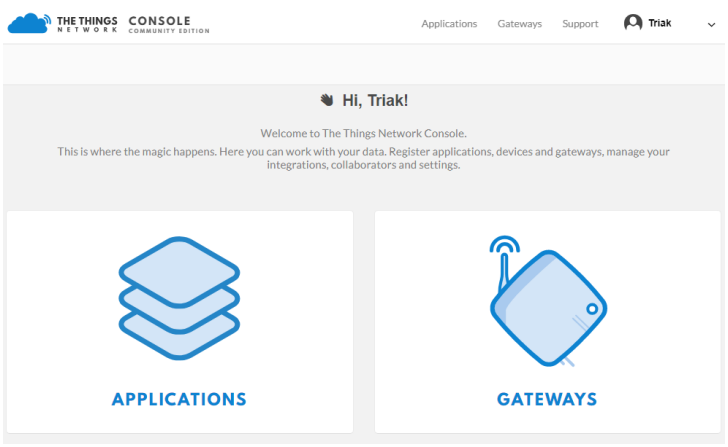


FIGURA 60. CONSOLA EN TTN

3. Dentro de *Applications* hay que rellenar algunos campos y otros se autocomplementan si se desea. Estos campos son:
   Application ID. Es el identificador de la aplicación que estamos creando en la red. Se trata de un conjunto de letras minúsculas y números (todo ello sin espacios) que identifica la aplicación. Por ejemplo: 123pepe.
   Description. Es un pequeño texto descriptivo de la aplicación. Por ejemplo: contador de personas. Con estos dos campos es suficiente,dado que los otros campos se autocomplementan una vez se hace click sobre “Add application”.

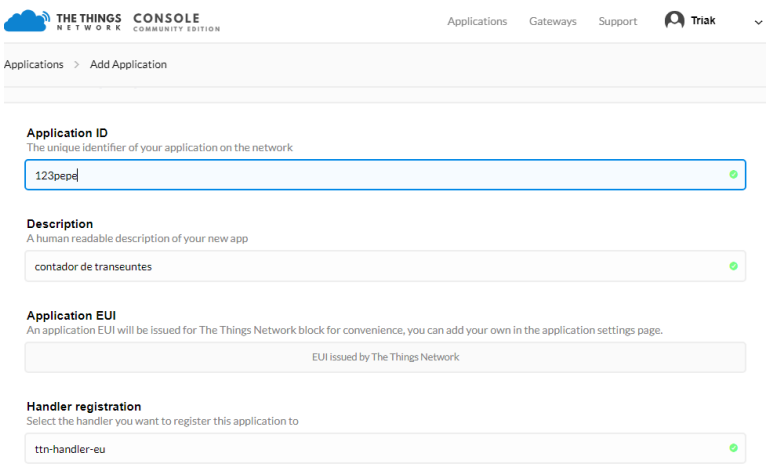


FIGURA 61. REGISTRO DE UNA APLICACIÓN EN TTN

4. Una vez añadida la aplicación hay que registrar el dispositivo que se va a emplear, de modo que se hace click sobre *register device* y se accede la configuración del registro del nodo final.

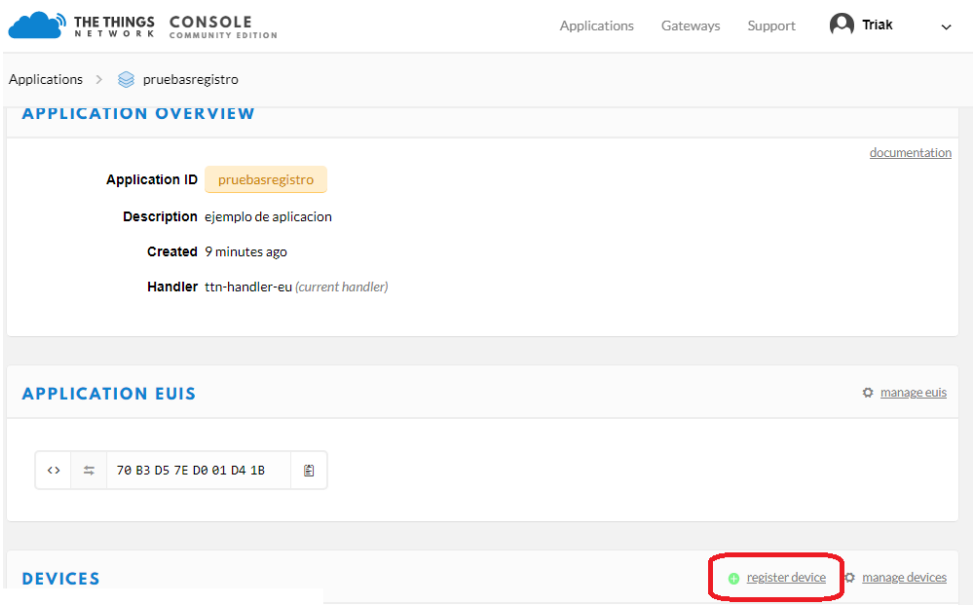


FIGURA 62. VISTA PREVIA DE LA APLICACIÓN

5. Dentro de la nueva pestaña se rellena el campo DeviceID, que es un conjunto alfanumérico que identifica el dispositivo inequívocamente y es único de cada dispositivo. Para obtenerlo se requiere cargar un pequeño sketch de Arduino en el dispositivo, que ejecutarse muestra en el monitor serie del IDE dicho identificador. El resto de campos se generan de modo automático. Se clica sobre *Register* y ya está.

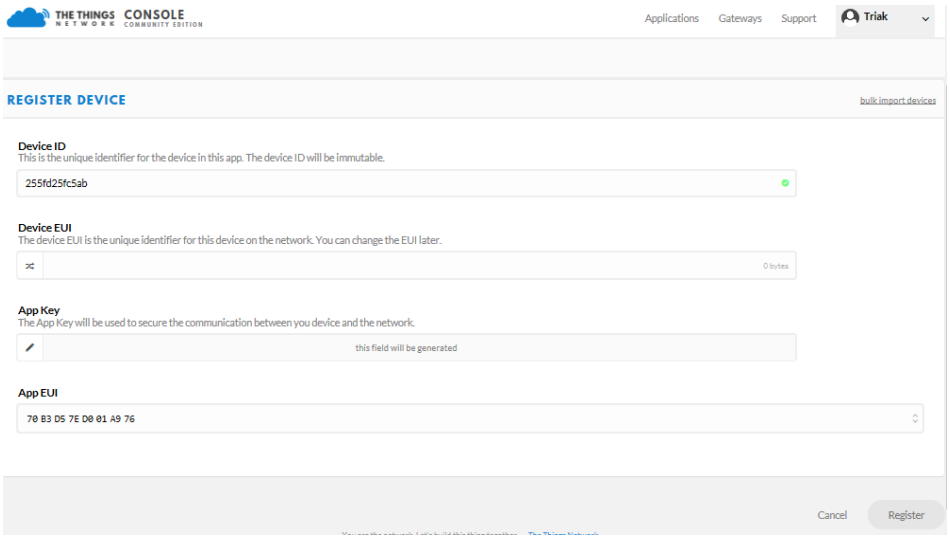


FIGURA 63. REGISTRO DE UN DISPOSITIVO

6. En este punto ya está el proceso finalizado y es posible enviar datos a TTN que serán visualizados mediante la aplicación. Ahora bien, para que el dispositivo puede conectarse con la aplicación y enviar datos se requiere que en su configuración incluya las claves de acceso: AppKey y AppEUI, para una conexión OTAA o Device Address, Network Session Key y App Session Key para una conexión ABP, que se deben incluir en el sketch de forma que el dispositivo se identifique ante la red.

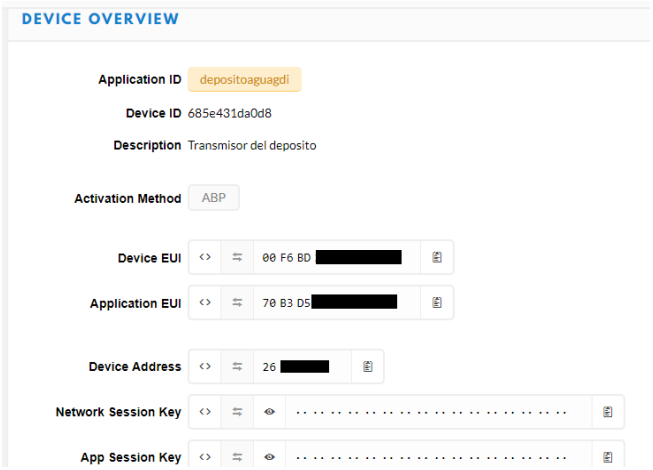


FIGURA 64. VISTA PREVIA DEL DISPOSITIVO REGISTRADO

9.2 Registro de una pasarela.

1. Desde *Gateways*, se clica en *register gateway*.
2. Se rellenan los campos correspondientes, donde básicamente hay que indicar el identificador del gatewa, una pequeña descripción, la frecuencia de trabajo (868 MHz), la localización aproximada, el router TTN correspondiente a la zona y si la antena es de interior o exterior. Se hace click en *Register gateway* y ya está finalizado el proceso.

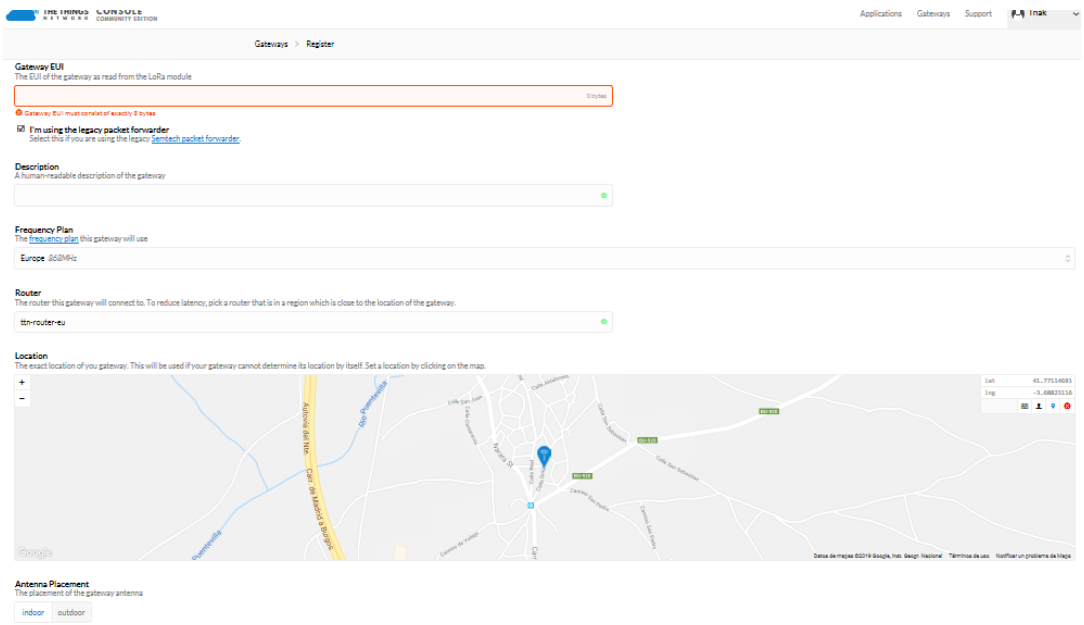


FIGURA 65. REGISTRO DE UN GATEWAY

3. Una vez finalizado el proceso se pueden ver las características del registro y empezar a emplearlo como pasarela. Además, es posible editar ciertos parámetros del dispositivo y que sean visibles o no.

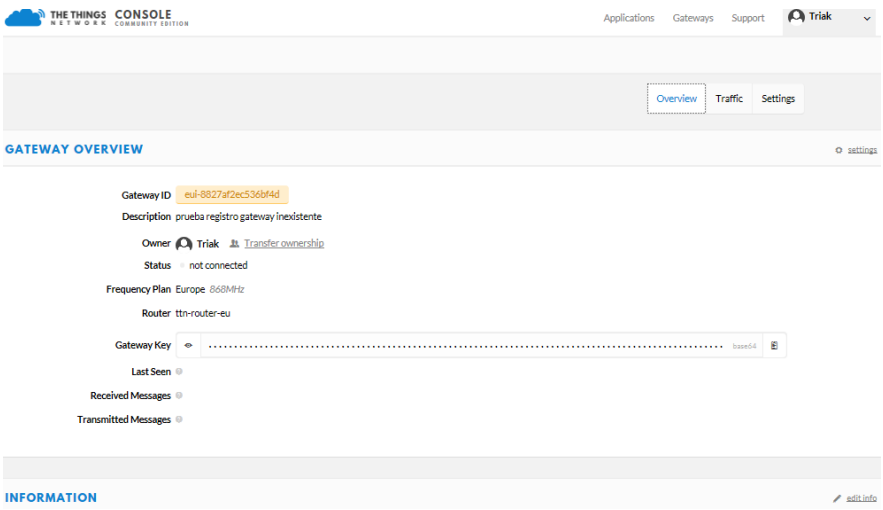


FIGURA 66. VISTA PREVIA DEL GATEWAY

10. Configuración del gateway Dragino LG01-P.

El gateway viene configurado de fábrica como WiFi AP, por lo que se puede acceder a el mediante un pc, abriendo un navegador web a través de la dirección IP 10.130.1.1. Una vez se ha accedido pide la contraseña que viene de fábrica por defecto: “dragino”. A partir de esto ya se tiene acceso a todos los parámetros configurables del gateway. Primero, se configura el método de conexión a Internet que empleará el gateway, mediante el menú Network -> Internet Access.

FIGURA 67. DRAGINO GAETWAY CONFIGURACION ACCESO A INTERNET

Se rellenan los campos siguientes con la información que se empleó en el registro en TTN, mediante el menú Sensor -> LoRa/LoraWAN.

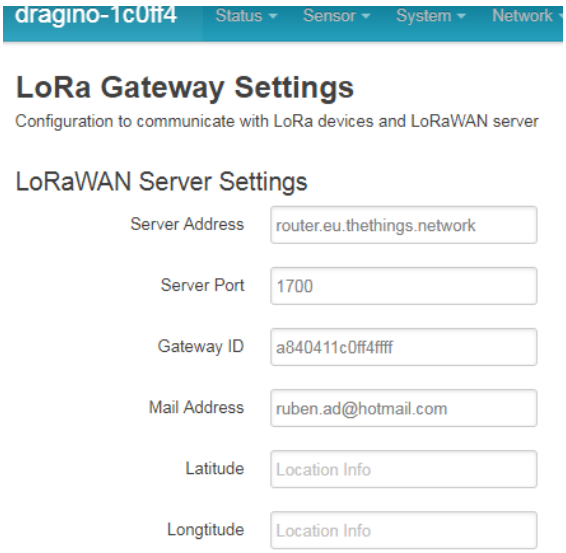


FIGURA 68. DRAGINO GATEWAY CONFIGURACIÓN LORAWAN

También, hay que configurar los parámetros de radio del gateway, dentro del mismo menú anterior.

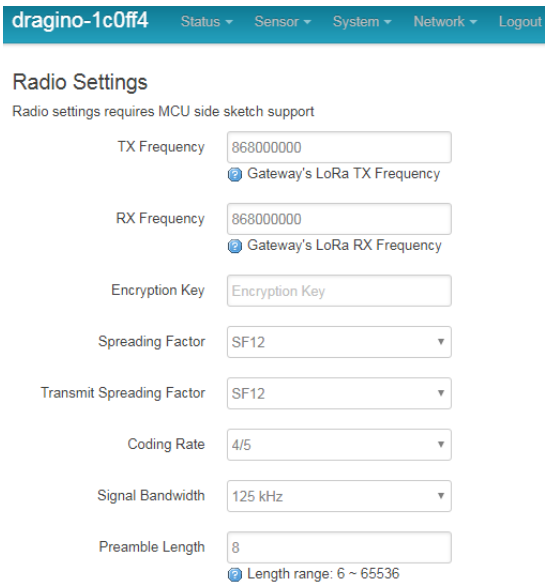


FIGURA 69. DRAGINO GATEWAY CONFIGURACIÓN LORA

Se selecciona el tipo de servidor IoT.

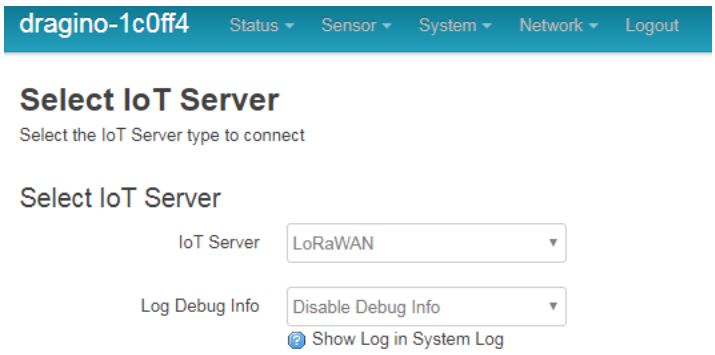


FIGURA 70. DRAGINO GATEWAY SERVER IOT

Existen varias versiones de sketchs que permiten el reenvío de los paquetes LoRa al servidor, así que una vez que se ha escogido uno de ellos se puede cargar en el gateway empleando el IDE de Arduino. Es decir, se abre el sketch en Arduino, se selecciona la tarjeta (Dragino Yun + UNO or LG01) y el puerto correspondiente al gateway y se sube el sketch. Así el gateway queda listo para reenviar los paquetes al servidor.

11. Sketch para Arduino en el prototipo.

Este es el sketch que se emplea en el prototipo del capítulo cuatro para realizar las pruebas correspondientes en el depósito.

```
#include <SoftwareSerial.h>
SoftwareSerial TTGO(7, 8); // Comunicacion serie con el TTGO emisor

const int sensores = 12; // testigo sensores midiendo
```

```
char inchar=0; // lee los caracteres que envia el modulo de
comunicaciones
String mensaje=""; // Para recibir datos del puerto serie

// para el sensor de nivel
const int EchoPin = 5;  // pin 5
const int TriggerPin = 6; // pin 6
int separacion = 45  ; // (cms)distancia del sensor al nivel lleno del
deposito

// para el sensor de pH
const int analogInPin = A0; // pin A0
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;

void setup() {
   TTGO.begin(115200);
   delay(100);
   Serial.begin(115200);
   Serial.println(F("Configurando el sistema (setup)........."));
   pinMode(bombas, OUTPUT);
   pinMode(sensores, OUTPUT);
   pinMode(TriggerPin, OUTPUT);
   pinMode(EchoPin, INPUT);

   Serial.println("Iniciando...");

   delay(1000);
}

//  Función que calcula el nivel del depósito
int nivel(){
   int distAgua = ping(TriggerPin, EchoPin);
   long cotaCero = 300 + separacion; // deposito lleno + separacion al
sensor
   int nivel = (cotaCero - distAgua)*100/300;
```

```
   return nivel;

  }

//Función que realiza el disparo del sensor de distancia
 int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;

   digitalWrite(TriggerPin, LOW);    //para generar un pulso limpio
ponemos a LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);   //generamos Trigger (disparo) de
10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);

   duration = pulseIn(EchoPin, HIGH);    //medimos el tiempo entre
pulsos, en microsegundos

   distanceCm = duration * 10 / 292/ 2;    //convertimos a distancia,
en cm

   Serial.print("Distancia: ");
   Serial.println(distanceCm);
   return distanceCm;
}

// funcion para medir el pH
  float medirPH(){
    for(int i=0;i<10;i++)
 {
  buf[i]=analogRead(analogInPin);
  delay(10);
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
  {
```

```
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
  }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;
 float phValue = -5.70 * pHVol + 21.76;
 return phValue;
    }

 //Funcion para enviar la informacion al TTGO emisor por puerto serie
 void sendToTTGO(int nivel, float ph){
   TTGO.print("Niv ");
   TTGO.print(nivel);
   TTGO.print("%");
   TTGO.print(" pH ");
   TTGO.println(ph);
  }

void loop() {
  digitalWrite(sensores, HIGH); // tyestigo de sensores leyendo
   Serial.println("Sensores midiendo...");
   int niv = nivel();
   float pH = medirPH();

   mensaje= (char)niv  +(char)niv;
   delay (5000);
   digitalWrite(sensores, LOW);// testigo sensores apagado
   Serial.print("Nivel del depósito: ");
   Serial.print(niv);
   Serial.println(" %");
   Serial.print("ph del agua: ");
```

```
   Serial.println(pH);

   //Enviar la info al TTGO emisor
   sendToTTGO(niv, pH);

    Serial.println("Mensaje enviado........");
    delay(300000); //5 minutos entre mediciones

}

//fin del sketch *********************************
```

13. Sketch en el TTGO emisor 868 MHz del prototipo (para TTN).

Este sketch corresponde al programa que se carga en el TTGO emisor del prototipo que debe subir a TTN los datos de nivel y pH que recibe del módulo Arduino.

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <U8x8lib.h>
#include <U8g2lib.h>
#define BUILTIN_LED 25

// I2C OLED Display works with SSD1306 driver
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16

// the OLED used
  U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 15, /* data=*/ 4,
/* reset=*/ 16);
  U8G2_SSD1306_128X64_NONAME_F_SW_I2C  Display(U8G2_R0,  /*  clock=*/
OLED_SCL,  /*  data=*/  OLED_SDA,  /*  reset=*/  OLED_RST);  //  Full
framebuffer, SW I2C
```

```
// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the early
prototype TTN
// network.

// Estos datos son del  TTGO 78D5421DA0D8
static const PROGMEM u1_t NWKSKEY[16] = { 0xBD, 0x43, 0xF2, 0x36,
0x93, 0x97, 0xA8, 0x92, 0x52, 0x89, 0x90, 0xE3, 0xDE, 0xED, 0xC5, 0xEA
};
// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the early
prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = {    0xA8, 0x9C, 0x6A, 0x80,
0x9D, 0x87, 0xDF, 0x20, 0x55, 0xE0, 0x9E, 0x75, 0xC7, 0x18, 0xBB, 0x89
};

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x26011E59 ; // direccion del Device

// These callbacks are only used in over-the-air activation, so they
are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will
complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "LoRa Deposito";
static osjob_t sendjob;
const unsigned int MAX_INPUT = 50;
static char input_line [15];

// para enviar la info recibida desde el puerto serie
String mensaje =" Pruebassss";
char inchar=0;
```

```
int cont=0;

// Schedule TX every this many seconds (might become longer due to
duty
// cycle limitations).
const unsigned TX_INTERVAL = 300; // este es el ultimo cargado

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 18,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 14,
  .dio = {26, 33, 32},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    u8x8.setCursor(0, 5);
    u8x8.printf("TIME %lu", os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
             mensaje= "EV_SCAN_TIMEOUT";
             delay(1000);
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            mensaje= "EV_BEACON_FOUND";
            delay(1000);
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            mensaje= "EV_BEACON_MISSED";
            delay(1000);
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
```

```
            mensaje= "EV_BEACON_TRACKED";
            delay(1000);
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            mensaje= "EV_JOINING";
            delay(1000);
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            mensaje= "EV_JOINED";
            // Disable link check validation (automatically enabled
            // during join, but not supported by TTN at this time).
             LMIC_setLinkCheckMode(0);
             delay(1000);
            break;
        case EV_RFU1:
            Serial.println(F("EV_RFU1"));
            mensaje= "EV_RFUI";
            delay(1000);
            break;
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            mensaje= "EV_JOIN_FAILED";
            delay(1000);
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            mensaje= "EV_REJOIN_FAILED";
            delay(1000);
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));
            mensaje= "EV_TXCOMPLETE";
            delay(1000);;

            digitalWrite(BUILTIN_LED, LOW);
```

```
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
              mensaje= "Received ACK";
              delay(1000);
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              u8x8.drawString(0, 6, "RX ");
              Serial.println(LMIC.dataLen);

              mensaje= "%i bytes", LMIC.dataLen;
              Serial.println(F(" bytes of payload"));

            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
           mensaje="EV_LOST_TSYNC";
           delay(1000);
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
           mensaje= "EV_RESET";
           delay(1000);
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            mensaje= "EV_RXCOMPLETE";
            delay(1000);
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            mensaje= "EV_LINK_DEAD";
            delay(1000);
            break;
```

```
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            mensaje= "EV_LINK_ALIVE";
            delay(1000);
            break;
         default:
            Serial.println(F("Unknown event"));
            u8x8.setCursor(0, 7);
          mensaje="UNKNOWN EVENT %d", ev;
          delay(1000);
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
        mensaje= "OP_TXRXPEND, not sent";
    } else {
        // Prepare upstream data transmission at the next possible
time.
        //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);

        LMIC_setTxData2(1, (uint8_t *)input_line,sizeof(input_line)-1,
0);
        Serial.println(F("Packet queued"));
        mensaje= "PACKET QUEUED";
          // contador de paquetes enviados
            cont= cont+1;
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));
```

```
    u8x8.begin();
    u8x8.setFont(u8x8_font_chroma48medium8_r);
    mensaje= "LoRaWAN LMiC TTN Node...";
//esto siguiente lo copio del sketch receptor para inicializar el display OLED
  Display.begin();
  Display.enableUTF8Print();    // enable UTF8 support for the Arduino print() function
  Display.setFont(u8g2_font_ncenB10_tr);

    SPI.begin(5, 19, 27);

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();

    // Set static session parameters. Instead of dynamically establishing a session
    // by joining the network, precomputed session parameters are be provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only copied to RAM
    // once. Copy them to a temporary buffer here, LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif
```

```
    #if defined(CFG_eu868)
    // Set up the channels used by the Things Network, which
corresponds
    // to the defaults of most gateways. Without this, only three base
    // channels from the LoRaWAN specification are used, which
certainly
    // works, so it is good for debugging, but can overload those
    // frequencies, so be sure to configure the full frequency range
of
    // your network here (unless your network autoconfigures them).
    // Setting up channels should happen after LMIC_setSession, as
that
    // configures the minimal channel set.
    // NA-US channels 0-71 are configured automatically

  // LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//     LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
BAND_MILLI);      // g2-band
    // TTN defines an additional channel at 869.525Mhz using SF9 for
class B
    // devices' ping slots. LMIC does not have an easy way to define
set this
```

```
    // frequency and support for class B is spotty and untested, so
this
    // frequency is not configured here.
    #elif defined(CFG_us915)
    // NA-US channels 0-71 are configured automatically
    // but only one group of 8 should (a subband) should be active
    // TTN recommends the second sub band, 1 in a zero based count.
    //                      https://github.com/TheThingsNetwork/gateway-
conf/blob/master/US-global_conf.json
    LMIC_selectSubBand(0);
    #endif

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink (note: txpow seems
to be ignored by the library)
    LMIC_setDrTxpow(DR_SF7,14);
    // Start job
    do_send(&sendjob);
}

// funcion para mostrar el mensaje por la pantalla oled del TTGO
void oled(String text){
  Display.clearBuffer();
    Display.setCursor(0,12); Display.print("Deposito agua");
    Display.setCursor(0,26);
    Display.print(text);
    Display.setCursor(0,46);
    Display.print(cont);
    Display.sendBuffer();
  }

// funcion para leer los datos del puerto serie y menetrlos en mydata
  void processIncomingByte (const byte inByte)
```

```
  {
   static unsigned int input_pos = 0;
  switch (inByte)
    {
    case '\n':   // end of text
      input_line [input_pos] = 0;  // terminating null byte

      // reset buffer for next time
      input_pos = 0;
      break;
    case '\r':   // discard carriage return
      break;
    default:
      // keep adding if not full ... allow for terminating null byte
      if (input_pos < (MAX_INPUT - 1))
        input_line [input_pos++] = inByte;
        mensaje= (char*)input_line;
        break;
         }  // end of switch
  } // end of processIncomingByte

void loop() {
if (Serial.available() >0){
          mensaje="";
          digitalWrite(25, HIGH);

        while(Serial.available()  >0)//  Se  leen  datos  entrantes  y  se
almacenan en un string
         {
           processIncomingByte (Serial.read ());
          }
         if(mensaje != " "){
          Serial.print("Lectura recibida: ");
          Serial.println(mensaje);
          oled(mensaje);
          }
        }
    os_runloop_once();
```

```
}
// Fin del sketch ***********************************
```

14. Mensajes con tarjeta SIM900 GSM/GPRS.

Se trata de una tarjeta o módulo de expansión perfectamente compatible con cualquier tarjeta del universo Arduino, incluso es posible encastrarla sobre la tarjeta Arduino UNO R3, debido a que los conectores que posee están alineados con esta finalidad. En la figura 28 se describen los elementos principales.

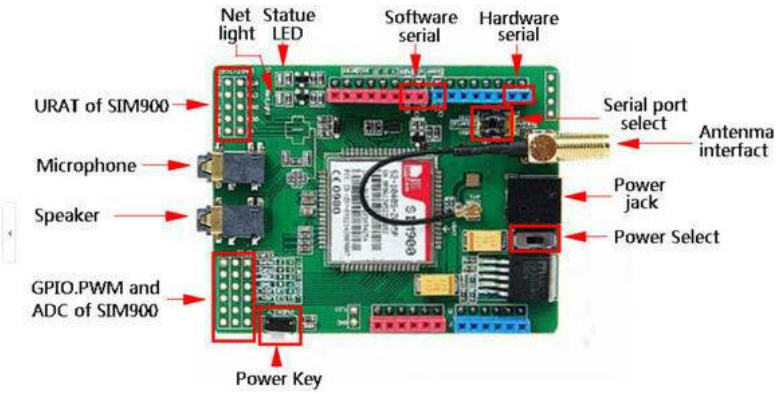


FIGURA 71. TARJETA SIM 900 GSM/GPRS

14.1 Hardware

Los elementos principales que conforman esta tarjeta son:

a) Chip SIM900: en su datasheet se pueden encontrar todas las características y funcionalidades. En la figura 29 se muestra el pin out del chip.

b) Antena: tiene un zócalo para conectar una antena mediante un terminal para cable coaxial, directamente al chip.

c) Zócalo tarjeta SIM: en la parte inferior de la tarjeta se encuentra el zócalo donde se debe insertar la tarjeta SIM desbloqueada. También en ese mismo lado se encuentra el alojamiento para una pila de botón de 3 V, que permite mantener la información cuando la tarjeta no tiene alimentación.

d) Conector de alimentación tipo Jack: para alimentar la tarjeta con 5 VDC. Normalmente, la intensidad operativa es de 450 mA, pero pueden haber ráfagas en las que se requiera hasta 2 A, por lo que la fuente de alimentación debe ser consecuente con este hecho.
e) UART: comunicaciones serie asíncronas por hardware mediante los pines D0 y D8 o mediante software por los pines D7 y D8. Esta característica se configura mediante unos jumpers a tal efecto.
f) Pines de entrada y salida: permiten la conexión con Arduino y presentan características similares aunque no coinciden todos. Hay pines analógicos, digitales y PWM.
g) Conectores para micrófono y auricular: conectores de tipo minijack para emplearlos en llamadas telefónicas.
h) Leds indicadores: indican el estado de los elementos de la tarjeta. En la tabla 5 se muestra la información que ofrecen.
Obviamente la tarjeta consta de toda una serie de componentes electrónicos (condensadores, resistencias, reguladores...) para permitir la correcta interacción entre los distintos elementos, pero no se entrará más en detalle en este aspecto.

14.2 Software.

Para programar las funcionalidades de la tarjeta se requiere emplear comandos AT y pese a que la lista de estos comandos es bastante extensa, en este caso se emplean solo unos pocos. Hay que aclarar que son los comandos AT y para ello, se puede decir que son lo que se conoce como el conjunto de comandos Hayes: un lenguaje desarrollado por la compañía Hayes Communications que prácticamente se convirtió en estándar abierto de comandos para configurar y parametrizar módems. Los caracteres «AT», que preceden a todos los comandos, significan «Atención», e hicieron que se conociera también a este conjunto de comandos como comandos AT.

Por un lado, aunque la lista de comandos AT es muy extensa, los comandos principales que se emplean son los siguientes:
AT: para verificar si el módulo SIM900 está funcionando adecuadamente para entrar en modo comando. Al enviar AT el SIM deberá contestarnos con un OK.
AT+CGMI: nombre del fabricante.

ATI: Ver la información del producto.
AT+IPR=?: el Baud Rate en el que puede operar el SIM.
AT+IPR?: para preguntar el Baud Rate actual.
AT+IPR=XXXX: configura la frecuencia deseada.
AT+COPS?: nombre de la compañía telefónica.
AT+CGSN: visualizar el IMEI del chip utilizado.
AT+CSCS?: tipo de texto.
AT+CSCS="XXX": Configurar a tipo de texto.
AT+CMGF?: Ver el formato de un mensaje, ya sea PDU(0) o SMS(1) .
AT+CMGF=1; // modo texto para SMS
AT+CMGS=04455XXXXXXXX: Enviar un SMS Se despliega el símbolo mayor que '>' Escribir mensaje y al finalizar presiona Ctrl+Z retornará OK si el SMS se envió correctamente.
AT+CMGL=ALL: Sirve para ver todos los mensajes que nos han llegado al SIM.
ATD04455XXXXXXXX: Sirve para hacer una llamada a cualquier teléfono móvil.
ATA: Sirve para contestar una llamada.
ATH: Sirve para colgar una llamada

Por otra parte, se pueden diferenciar dos operaciones software sobre la tarjeta:
a) Programación y verificación de la tarjeta, donde conectando la tarjeta SIM900 GSM/GPRS a un PC, ya sea mediante un conversor FTDI que convierta la señal del puerto USB o bien empleando el conversor de la tarjeta Arduino, es posible verificar el funcionamiento y características de la tarjeta. Para ello, se recurre a un emulador de puerto serie como puede ser Hyperterminal o Teraterm o cualquier otro similar.
b) Programación del sketch en Arduino, donde los comandos AT son interpretados por al ATMega 328 de la tarjeta Arduino y, mediante la comunicación serie, se transmiten a la tarjeta de comunicaciones para que realice las operaciones correspondientes. Esto es, el propio sketch cargado en el microcontrolador incluye los comandos AT, de forma que dependiendo de qué programa este ejecutando, puede ordenar a la tarjeta de comunicaciones enviar o recibir datos. Por el contrario, también puede ser que sea la tarjeta de comunicaciones la que contenga alguna instrucción o datos que necesite transmitir al microcontrolador, por lo que esto debe estar incluido en el sketch.

14.3 Verificación de comunicaciones.

En primer lugar, se procede comprobando la operatividad de la tarjeta de comunicaciones. Para ello, se realizan los siguientes pasos:

1. Se alimenta la tarjeta con 5 VDC por el conector jack y se coloca el selector de alimentación en la posición de alimentación externa. La alimentación proviene de un regulador conectado a una fuente de 12 VDC.
2. Se acopla la antena.
3. Se inserta la tarjeta SIM desbloqueada en su ranura.
4. Se seleccionan los pines D0 y D1 como puerto de comunicaciones serie, mediante el jumper a tal efecto.
5. Se ejecuta Hyperterminal, introduciendo los parámetros necesarios y se verifica la operatividad del módulo mediante los parámetros AT. Ver figuras 32 y 33.

La tarjeta SIM900se alimenta mediante un regulador que está conectado a una fuente de alimentación de 10 VDC. Además, se ha empleado el conversor FTDI de modo que se comunica directamente la tarjeta de comunicaciones con el PC, conectando en los pines Rx y Tx de ambas tarjetas.

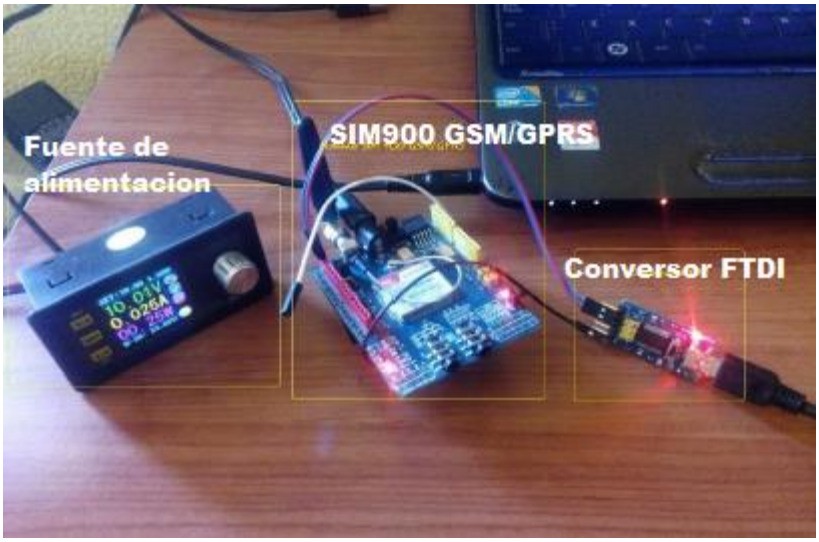


FIGURA 72. VERIFICACIÓN TARJETA SIM900

Una vez abierto Hyperterminal y configurados los parámetros de comunicación, se van introduciendo los comandos AT para ver si la tarjeta de comunicaciones está operativa, figura 5.6. Se puede ver que muestran tanto las velocidades de comunicación serie (AT+IPR=?), como la velocidad actual del módulo (AT+IPR?). De

igual modo, informa del fabricante (AT+CGMI) y del IMEI del chip de comunicaciones (AT+CGSN). Con todo, hay que mencionar que este proceso se puede realizar con cualquier otro emulador de puerto serie, incluso con el monitor serie del IDE de Arduino, sin necesidad de cargar ningún sketch, previamente.

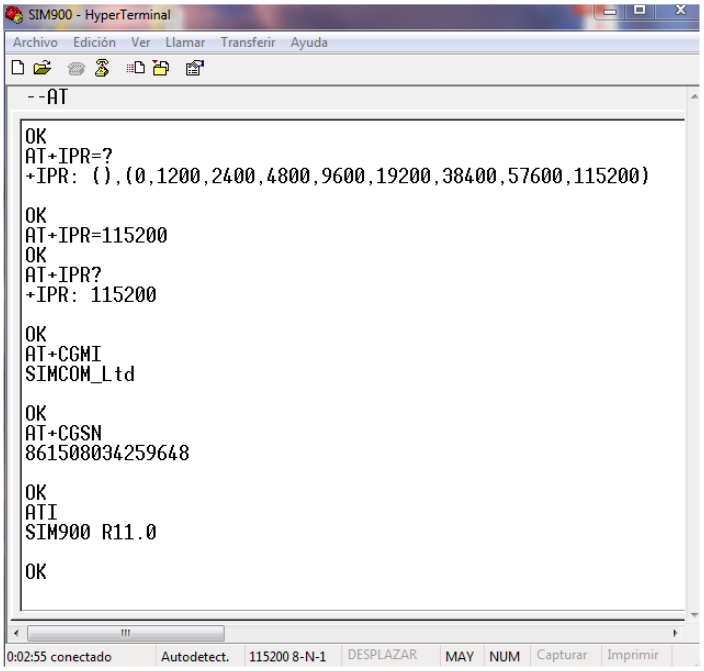


FIGURA 73. SIMULACIONES CON COMANDOS AT

En segundo lugar, el siguiente paso es introducir una tarjeta SIM, para verificar que hay conectividad GSM, figura 33. Esto es necesario, debido a que puede ser que puede haber algún problema como: IMEI de la tarjeta bloqueado, no exista cobertura de servicio, fuente de alimentación sin la energía suficiente, etc. Así, pueden verse los comandos empleados para verificar que hay comunicación mediante la red GSM: el comando AT+COPS? permite conocer a que operadora está conectada la tarjeta, con AT+CMGF? se informa del tipo de mensajes (PDU(0) o SMS(1)), con AT+CMGS=”NUMERO DE TELÉFONO” se envía un SMS, para ello una vez introducido el comando aparece el signo ’>’ donde se escribe el mensaje y se pulsa CTRL+Z para enviarlo. Una enviado se informa sobre este hecho si no hay errores. Finalmente, se pueden realizar llamadas telefónicas empleando el comando

ATDXXXXXXXXX(Nº de teléfono); y se puede aceptar una llamada entrante o finalizarla mediante ATA y ATH, respectivamente.

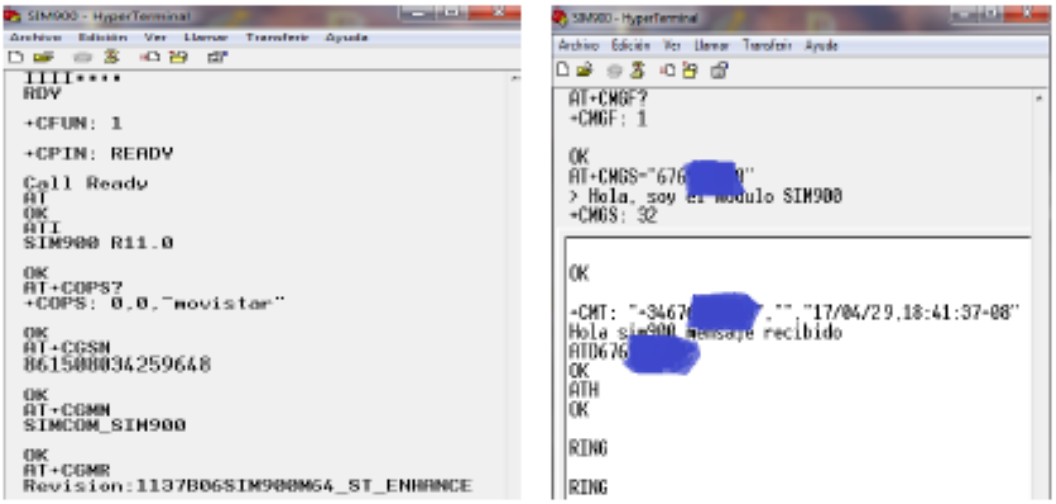


FIGURA 74. COMANDOS AT PRINCIPALES

Existen muchos más comandos AT, pero con verificar algunos que se han probado es suficiente para tener la certeza de que el prototipo está preparado para comunicarse de forma remota.

14.4 Código para envío de SMS.

A continuación se muestra la función que se debe incluir dentro del sketch del sistema para poder enviar SMS. Esta función se invoca desde cualquier punto del sketch, donde sea necesario enviar un mensaje de alarma al número preestablecido en el sketch. Para que esta función se pueda emplear la tarjeta SIM deb estar desbloqueada, es decir que debe tener acceso a la red de su operadora. En caso contrario, se debe incluir en el sketch los comandos necesarios para introducir el PIN y desbloquearla.

```
void sendSMS() {
SIM900.print("AT+CMGF=1\r"); // modo texto para SMS
delay(200);
SIM900.println("AT+CMGR=?");// se activa el código para recpción de
mensajes
```

```
delay(200);
SIM900.print("AT+CNMI=2,2,0,0\r");// comando para poder visualizar los
mensajes
delay(200);
SIM900.print("AT+CMGS=");  //  móvil  del  destinatario  en  formato
internacional
delay(100);
SIM900.print((char)34);//ponemos las comillas ("), para que lo tome
debe ser char de lo contrario el serial envia caracter por caracter
SIM900.print(numMovil);//colocamos número de teléfono preestablecido
SIM900.println((char)34);//volvemos a poner el caracter (")
delay(200);
SIM900.print(smstext); // mensaje a enviar
delay(100);
SIM900.print((char)26); // Codigo ASCI para CTRL+Z,  codigo 26, se
envia el mensaje
delay(500);
Serial.println(F("!!!!!!!!!!!!!!!!!!!!!!!!: "));
Serial.print(F("SMS enviado al numero: "));
Serial.println(numMovil);
escribirLCD("SMS enviado");
delay(4000); // margen de tiempo para el envío del mensaje
 Serial.println(F("SMS procesado!!!"));
 delay(1000);
}
```

14.5 Código para simulación de alarmas SMS empleando sensor DHT11.

El siguiente código se emplea para poner de manifiesto que un pequeño sistema de monitorización de temperatura puede incluir el envío de alarmas mediante sms al emplear la tarjeta SIM 900GSM/GPRS. De este modo, se puede verificar que es factible incluir el envío de mensajes de alarma en el sistema que se ha diseñado en este proyecto. En este caso concreto se ha establecido la temperatura de 28 ºC como límite, de modo que si se supera el sistema envía un mensaje a un número de móvil prefijado en el sketch informando sobre este hecho.

```
#include <dht.h>
```

```
#include <SoftwareSerial.h>
dht DHT;
SoftwareSerial SIM900(7, 8); //pines para comunicacion serie con la SIM900
#define DHT11_PIN 13// para arduino uno entrada de señal de DHT
int pulso=9; //activar el modulo GSM por sw, conectar al pin 9 del modulo
int testigo=12;
////////variables para las comunicaciones///////////////
const char numMovil[]= "676[redacted]" ; // El número de teléfono movil
char inchar=0; // lee los caracteres que envia el modulo de comunicaciones
String mensaje=""; // Para recibir datos del puerto serie de la SIM900
String smstext; // Texto del mensaje que recibirá el destinatario de la solicitud de Info.
void setup(){
  pinMode(testigo,OUTPUT);// pin 12 para ver si se envia SMS
  Serial.begin(19200);
  SIM900.begin(19200);
  SIM900power();
  delay(10000); // tiempo para acceder a la red GSM
  Serial.println("Sensor DHT11 midiendo...");
}
//Función de encendido por software de la tarjeta de comunicaciones//////////////////
void SIM900power() {
Serial.println(F("Activando módulo de comunicaciones.............."));
digitalWrite(pulso, HIGH);
delay(1500);
digitalWrite(pulso, LOW);
//digitalWrite(testigo, LOW);
delay(1000);
Serial.println(SIM900.print("AT+CPIN=2940\r"));// introducir PIN de la SIM
delay(10000);
Serial.println(("Modulo de comunicaciones activado................"));
}
//Función para el envío de SMS//////////////////
```

```
void sendSMS() {
  digitalWrite(testigo, HIGH);
SIM900.print("AT+CMGF=1\r"); // modo texto para SMS
delay(200);
SIM900.println("AT+CMGR=?");//ACTIVAMOS CODIGO PARA RECIBIR MENSAJES
delay(200);
SIM900.print("AT+CNMI=2,2,0,0\r");// comando para poder visualizar los
mensajes
delay(200);
SIM900.print("AT+CMGS=");  //  movil  del  destinatario  en  formato
internacional
delay(100);
SIM900.print((char)34);//ponemos las comillas ", para que lo tome debe
ser char de lo contrario el serial envia caracter por caracter
SIM900.print(numMovil);//colocamos numero de telefono
SIM900.println((char)34);//volvemos a poner el caracter "
delay(200);
SIM900.print(smstext); // mensaje a enviar
delay(100);
SIM900.print((char)26);// Codigo ASCI para CTRL+Z, se envia el mensaje
delay(500);
Serial.println(F("!!!!!!!!!!!!!!!!!!!!!!!!: "));
Serial.print(F("SMS enviado al numero: "));
Serial.println(numMovil);
delay(4000); // margen de tiempo para el envío del mensaje
 Serial.println(F("SMS procesado!!!"));
 delay(1000);
 digitalWrite(testigo, LOW);
}
void loop() {
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  Serial.println("========= ");
  if (DHT.temperature > 28){
```

```
    smstext= "Alarma: temperatura superior a 28°C"; // mensaje de
alrma que recibe el destinatario
     sendSMS();
     }
  delay(10000);  // tiempo de espera entre lecturas de temperatura
}
```

En las siguientes imágenes se pueden ver tanto el pequeño prototipo para probar el sketch, como una serie de mensajes que el móvil destinatario recibe cuando se supera la temperatura indicada.

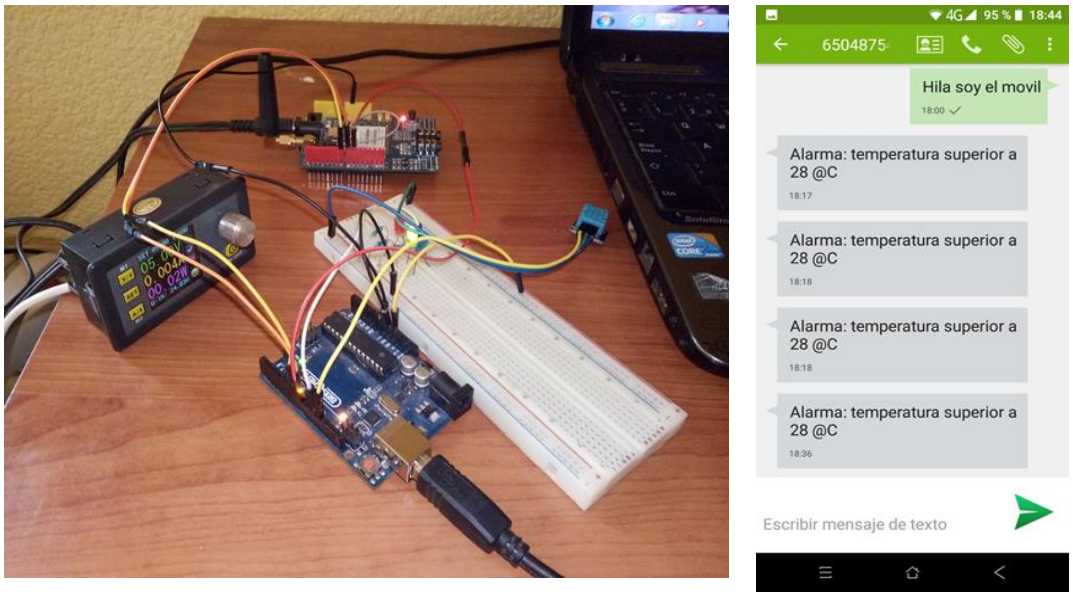


FIGURA 75. PRUEBAS CON SENSOR DHT11 Y ENVÍO DE SMS

14.6 Sketch en Arduino que incluye envío de SMS de alarma (prototipo).

Este sketch es el que se incluye en la tarjeta Arduino del prototipo, pero con las modificaciones necesarias para permitir que se puedan enviar mensajes de alarma, en caso necesario, empleando la tarjeta SIM900 GSM/GPRS.

```
/* ************** TFM Diseño de Sistemas de Comunicación**************
Código de monitorizaciónn y comunicaciones. Se mide el nivel del
deposito de agua y el ph***  ***se actúa sobre las bombas y
```

```
dispositivos de corrección del pH. Se envia un SMS de alarma***
****por nivel bajo o ph peligroso. Se pasa información por puerto
serie al TTGO emisor para el*** ***enlace LoRa entre lo 2 TTGO y al
modulo GMS900 para enviar SMS de alarma.
 ********************Rubén Adrián de la Cámara************************
*********************************29/05/2019********************** */

#include <SoftwareSerial.h>
SoftwareSerial TTGO(7, 8); // Comunicacion serie con el TTGO emisor

// para comunicaciones GSM (SMS)
SoftwareSerial SIM900(2, 3); //pines para comunicacion serie con la
SIM900
const int bombas= 13; //activacion bombas de agua
const int sensores = 12; // testigo sensores midiendo

//char inchar=0; // lee los caracteres que envia el modulo de
comunicaciones
//String mensaje=""; // Para recibir datos del puerto serie

// Para el modulo GSM900
const int pulso=9; //activar el modulo GSM por sw, conectar al pin 9
del modulo GSM
const char numMovil[]= " " ; // Introducir el número de teléfono movil
char inchar=0; // lee los caracteres que envia el modulo de
comunicaciones
String mensaje=""; // Para recibir datos del puerto serie de la SIM900
String smstext; // Texto del mensaje que recibirá el destinatario de
la solicitud de
int temp; // para enviar los mensaje de alarma cada tres ciclos del
bucle (15 minutos)

// para el sensor de nivel
const int EchoPin = 5;  // pin 5
const int TriggerPin = 6; // pin 6
int separacion = 40  ; // (cms)distancia del sensor al nivel lleno del
deposito

// para el sensor de pH
const int analogInPin = A0; // pin A0
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10];

// variables globales de nivel  y pH
int niv;
float pH;
```

```
void setup() {
   SIM900.begin(19200);
   Serial.begin(19200);
   Serial.println(F("Configurando el sistema (setup)........."));
    SIM900power();
   TTGO.begin(115200);
   delay(100);

   pinMode(bombas, OUTPUT);
   pinMode(sensores, OUTPUT);
   pinMode(TriggerPin, OUTPUT);
   pinMode(EchoPin, INPUT);

   Serial.println("Iniciando...");
   temp = 0;
   delay(500);
}

//Activacion del modulo SIM900
void SIM900power()
{
Serial.println(F("Activando modulo de comunicaciones.............."));
digitalWrite(pulso, HIGH);
delay(1500);
digitalWrite(pulso, LOW);
//digitalWrite(testigo, LOW);
delay(1000);
while(SIM900.available() >0)
{
inchar=SIM900.read();
Serial.print(inchar);
}
delay(1000);
Serial.println(SIM900.print("AT+CPIN=2940\r"));// introducir PIN de la
SIM
delay(10000);
while(SIM900.available() >0)
{
inchar=SIM900.read();
Serial.print(inchar);
}
Serial.println(SIM900.print("AT+Cops?\r"));//  a  que  operadora  se  ha
conectado
delay(1000);
while(SIM900.available() >0)
{
inchar=SIM900.read();
Serial.print(inchar);
}
```

```
delay(1000);
Serial.println(("Modulo de comunicaciones activado..............."));
}

//Función para el envío de SMS/////////////////////
void sendSMS()
{
SIM900.print("AT+CMGF=1\r"); // modo texto para SMS
delay(200);
SIM900.println("AT+CMGR=?");//ACTIVAMOS CODIGO PARA RECIBIR MENSAJES
delay(200);
SIM900.print("AT+CNMI=2,2,0,0\r");// comando para poder visualizar los
mensajes
delay(200);
SIM900.print("AT+CMGS=");  //  movil  del  destinatario  en  formato
internacional
delay(100);
SIM900.print((char)34);//ponemos las comillas ", para que lo tome debe
ser char de lo contrario el serial envia caracter por caracter
SIM900.print(numMovil);//colocamos numero de telefono
SIM900.println((char)34);//volvemos a poner el caracter "
delay(200);
SIM900.print(smstext); // mensaje a enviar
delay(100);
SIM900.print((char)26); // Codigo ASCI para CTRL+Z,  codigo 26, se
envia el mensaje
delay(500);
Serial.print(F("SMS enviado al numero: "));
Serial.println(numMovil);
delay(4000); // margen de tiempo para el envío del mensaje
while(SIM900.available() >0)
{
inchar=SIM900.read();
Serial.print(inchar);
}
 Serial.println(F("SMS procesado!!!"));
 delay(1000);
}

//  Función que calcula el niven del depósito
int nivel(){
   int distAgua = ping(TriggerPin, EchoPin);
   long cotaCero = 300 + separacion; // deposito lleno + separacion al
sensor
   int nivel = (cotaCero - distAgua)*100/300;
   return nivel;

  }
```

```
//Función que realiza el disparo del sensor de distancia
 int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;
   digitalWrite(TriggerPin, LOW);   //para generar un pulso limpio
ponemos a LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);  //generamos Trigger de 10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);
   duration = pulseIn(EchoPin, HIGH);   //medimos el tiempo entre
pulsos, en microsegundos
   distanceCm = duration * 10 / 292/ 2;    //convertimos a distancia,
en cm
   Serial.print("Distancia: ");
   Serial.println(distanceCm);
   return distanceCm;
}

// funcion para medir el pH
  float medirPH(){
    for(int i=0;i<10;i++)
 {
  buf[i]=analogRead(analogInPin);
  delay(10);
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
  {
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
  }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;
 float phValue = -5.70 * pHVol + 21.76;
 return phValue;
    }

 // funcion para enviar alarmas SMS
 void alarmaSMS(){
    if(niv < 50){
      Serial.println("Alarma !!! Nivel inferior al 50%");
```

```
        smstext="Alarma: Nivel del deposito inferior al 50%";
        sendSMS();
        delay(1000);
       }
     else if (pH < 6 || pH > 8){
      Serial.println("Alarma !!! pH fuera de rango aceptable");
        smstext="Alarma: pH fuera de rango aceptable";
        sendSMS();
        delay(1000);
       }
  }

 //Funcion para enviar la informacion al TTGO
 void sendToTTGO(int nivel, float ph){

   TTGO.print("Niv ");
   TTGO.print(nivel);
   TTGO.print("%");
   TTGO.print(" pH ");
   TTGO.println(ph);
  }

void loop() {
   digitalWrite(sensores, HIGH); // testigo de sensores leyendo
   Serial.println("Sensores midiendo...");
   niv = nivel();
   pH = medirPH();
   mensaje= (char)niv  +(char)niv;
   delay (5000);
   digitalWrite(sensores, LOW);// testigo sensores apagado
   Serial.print("Nivel del depósito: ");
   Serial.print(niv);
   Serial.println(" %");
   Serial.print("ph del agua: ");
   Serial.println(pH);

   //Enviar la info al TTGO emisor
   sendToTTGO(niv, pH);
   // comprobar si es necesario enviar SMS de alarma
  if (niv<50 || pH<6 || pH > 8){
     alarmaSMS(); // llama a la funcion para enviar SMS de alarma
     Serial.println("Nivel bajo o pH fuera de rango");
     Serial.println("Mensaje enviado.........");
     }
    delay(600000); // lecturas cada 10 minutos
}
//Fin del sketch  ************************************************
```

15. Instrucciones para montaje del prototipo.

Siguiendo las instrucciones que se facilitan a continuación, será posible montar el prototipo y realizar las pruebas necesarias para verificar que el sistema es operativo. En primer lugar, en la tabla 1 se proponen las conexiones entre todos los elementos. Los pines A son analógicos y los D son digitales. Es imprescindible respetar la numeración en las conexiones para que el sketch funcione. Por otro lado, tanto Arduino como el módulo SIM900 deben alimentarse con una fuente que proporcione entre 5 y 9 Vdc, con una intensidad suficiente (al menos 1 A), sobre todo debido a que la tarjeta GSM requiere bastante potencia (más de 300 mW) para establecer comunicaciones con la red. Aparte, como se han empleado dos TTGO las conexiones de ambos son idénticas, pudiéndose realizar en paralelo.

Entradas / Salidas	Pines Arduino	Otras conexiones
Sensor nivel Vdc		5 Vdc
Sensor nivel GND		GND común
Sensor nivel Trigger	D6	
Sensor nivel Echo	D5	
Sensor pH Vdc		5 Vdc
Sensor pH GND		GND común
Sensor pH Po	A0	
TTGO 3.3 V		3.3 Vdc
TTGO GND		GND común
TTGO TXD	D7	
TTGO RXD	D8	
SIM900 pin 7	D3	
SIM900 pin 8	D2	
SIM900 pin 9	D9	
SIM900 GND		GND común

TABLA 14. CONEXIONES ENTRE DISPOSITIVOS

16. Imágenes del prototipo experimental.

A continuación se muestran algunas imágenes de los elementos que forman parte del prototipo construido para pruebas en el depósito, incluyendo algunos comentarios.

1. Sensor de nivel: se ha introducido dentro de una caja de plástico y se ha sellado para evitar que entren salpicaduras de agua que deterioren el sensor.

FIGURA 76. SENSOR DE NIVEL EN EL PROTOTIP

2. Sensor de pH: se ha colocado sobre una placa de polietileno para que flote en el agua y pueda realizar las medidas según sube o baja el nivel de agua en el depósito. Si estuviera fijo sería imposible realizar dichas medidas.

FIGURA 77. SENDOR DE PH EN EL PROTOTIPO

3. Monitorización, comunicaciones y alimentación: conjunto de elementos que forman parte del prototipo y que permiten medir y enviar la información, junto con la fuente de alimentación. La fuente de alimentación regulable alimenta el módulo Arduino y el módulo SIM900, mientras que los TTGO se alimentan a 3.3 Vdc desde una salida a dicho voltaje de la SIM900. De igual modo los sensores se alimentan a 5 Vdc desde un pin de salida de Arduino con esta tensión, siendo todas las tierras comunes.

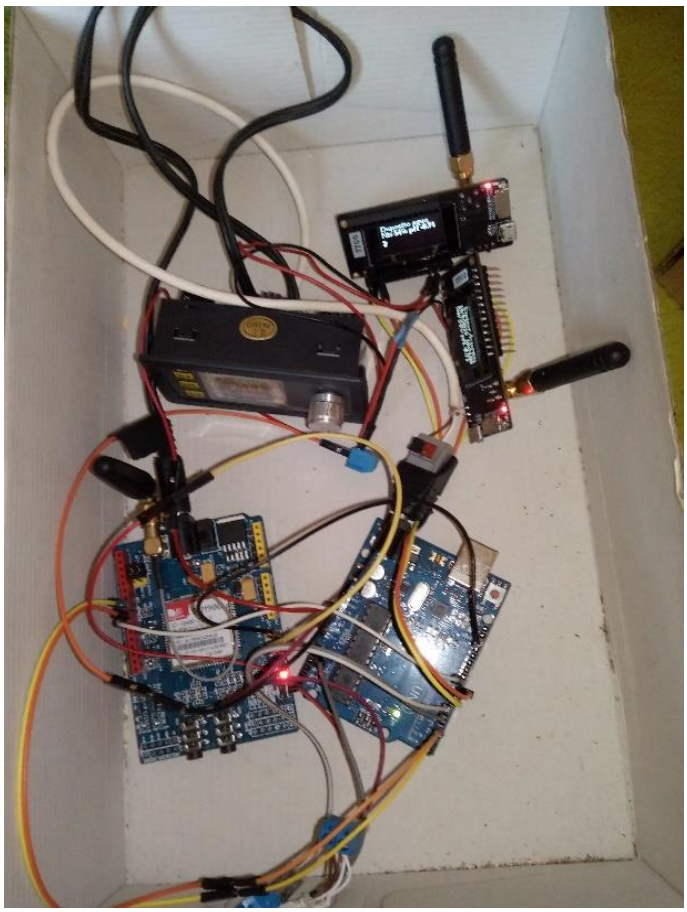


FIGURA 78. ARDUINO, GSM900 Y EMISORES LORA

Enlace al video del depósito:
https://drive.google.com/open?id=1QOvsxuG0se4hnF9qUrxTIERz83-Ni8f3