industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

# UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

# DISEÑO Y DESARROLLO DE UN GATEWAY DE BAJO COSTE PARA MONITORIZAR PARÁMETROS EN AGRICULTURA DE PRECISIÓN

**TRABAJO FIN DE GRADO**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA


**Autor:** **Jonathan Javier Cedeño Chamba**
Director: Juan Antonio López Riquelme
Codirector: Antonio Mateo Aroca


Cartagena, 09 de diciembre del 2020

Universidad Politécnica de Cartagena

# Índice

# Capítulo 1

# Introducción y objetivos

## *1.1 Introducción*

A lo largo de toda nuestra historia el ser humano siempre ha ido avanzando en cuanto a desarrollo se refiere. Se puede ir observando en su desarrollo con la agricultura, la ganadería, la forma de vida e incluso en la interacción con los demás y con su entorno. Esto nos lleva a este tiempo en donde el desarrollo del ser humano sigue en progreso y gracias a él, se pueden observar avances tan significativos como que la lejanía no supone un impedimento a la hora de comunicarse, ya que existen diferentes medios que no se pueden observar a simple vista como Internet, que ayudan a mejorar esta interacción social.

Pero más allá de ello, el ser humano no sólo ha avanzado en cuanto a la interacción con sus semejantes, sino que también ha mejorado su interacción con el entorno. Cada vez es más necesario mejorar este aspecto debido a la cantidad de información que se obtiene y a la utilidad que posteriormente se obtiene de ella. El conocimiento de nuestro entorno supone un gran avance a tareas que requieren de su conocimiento, ya que obtenemos datos precisos que nos ayudan a optimizar tareas y reducir pérdidas.

Por ello, la posibilidad de tener un entorno inteligente mediante dispositivos que proporcionen una respuesta rápida al usuario y se transmita información de manera segura, rápida y eficaz es un objetivo a desarrollar en este siglo. Gracias a las diferentes tecnologías desarrolladas hasta el día de hoy se puede tener dispositivos conectados mediante una arquitectura hardware y software que permita la comunicación y transmisión de datos entre ellos.

Un dispositivo muy avanzado en nuestro tiempo es el *Smartphone*, y es que hoy en día el *smartphone* se ha extendido tanto que hay muy pocas personas que no posean un dispositivo como estos en sus manos. No es para menos debido a la gran facilidad que ofrece con servicios como mensajería, llamadas, entretenimiento, entre muchas otras cosas. Una de ellas es Internet, por el cual nos podemos comunicar con diferentes servicios de todo el mundo simplemente teniendo conexión a una red. Esto abre muchas posibilidades que ofrece el *smartphone* que no ofrecen muchos otros dispositivos, como la posibilidad de obtener un entorno inteligente.

El *smartphone* es un dispositivo muy potente capaz de ejecutar operaciones muy complejas con el cual se pueden visualizar las diferentes conexiones y leer de manera visual la evolución de los datos recolectados del entorno para mejorar la comprensión de ellos y de esta manera proceder a ejecutar tareas anteriormente complejas, de manera más sencilla y con mayor eficacia.

Por lo anteriormente expuesto, el objetivo de este proyecto es plantear una estructura hardware y software para la recolección de datos del entorno de manera eficaz y segura. Una arquitectura que permita mediante diferentes tipos de transmisión de datos una comunicación con el usuario que ayude a comprender de manera visual el entorno y sus características.

## *1.2 Objetivos*

El objetivo principal de este proyecto es proporcionar una arquitectura completa de bajo coste con su diseño y desarrollo que permita realizar una monitorización en tiempo real de un dispositivo sensor que recolecte datos de su entorno y transmita la información hasta el usuario final, que será capaz de leer e interpretar de manera visual la evolución histórica de los datos. Para conseguir dicho objetivo es necesario llevar a cabo los siguientes puntos:

- Proponer una arquitectura hardware de bajo coste para un dispositivo de tipo Gateway que haga de pasarela con nodos sensores remotos desplegados en un dominio agronómico.

- Diseñar, desarrollar y validar el dispositivo Gateway siguiendo la arquitectura definida.

- Diseñar y desarrollar la transmisión de datos mediante un *smartphone* y el dispositivo mencionado.

- Realizar test para validar el funcionamiento del equipo en el entorno.

## 1.3 Fases del Proyecto

Con el objetivo de cumplir con las especificaciones anteriormente mencionadas, se realizará la selección de diferentes tipos de sistemas con los que cuales se permita lograr la conexión y transmisión de datos de forma inalámbrica con dispositivos inteligentes capaces de soportar dichas tecnologías.

Una vez seleccionado el sistema en concreto, se realizarán las tareas de diseño, desarrollo e implementación, tanto a nivel de hardware como a nivel de software. Para llevar a cabo las tareas y los objetivos descritos se establecen las siguientes fases dentro del proyecto:

- Revisar la literatura científica sobre sistemas de monitorización para agricultura de precisión.
- Realizar una propuesta de arquitectura hardware de bajo coste para el sistema.
- Seleccionar los elementos hardware principales del equipo a desarrollar.
- Diseñar y fabricar el dispositivo.
- Validar el funcionamiento del equipo en el entorno.
- Redacción de la memoria del trabajo.

## 1.4 Desarrollo de la Memoria

### 1.4.1 Capítulo 1. Introducción y objetivos

El Capítulo 1 presenta los objetivos a desarrollar durante este trabajo, además de enumerar las diferentes fases de la estructura de la memoria presentada punto por punto.

### 1.4.2 Capítulo 2. Estado del Arte

En el Capítulo 2 se desarrolla un estudio completo de los diferentes tipos de sistemas de bajo coste que se exponen en el panorama actual de las nuevas tecnologías, así como sus protocolos de comunicación y características propias.

### 1.4.3 Capítulo 3. Descripción del Sistema

En el Capítulo 3 se describe a nivel general las características del sistema. Con las opciones planteadas en el capítulo 2, se justifica la elección de cada una de ellas y se prepara el terreno para la programación.

### 1.4.4 Capítulo 4. Descripción de la Arquitectura desarrollada

En el cuarto Capítulo se describe en profundidad todos los procesos y conexiones del sistema, así como toda la programación del software necesario para las distintas partes que componen el sistema.

### 1.4.5 Capítulo 5. Conclusiones y Trabajo Futuro

En este capítulo se detalla todo el progreso del trabajo de manera resumida, argumentando las características generales del mismo y las conclusiones obtenidas tras su desarrollo. A su vez, como parte final, se muestran varias líneas futuras de desarrollo que pueden mejorar notablemente el sistema completo.

# Capítulo 2

# Estado del Arte

## *2.1 Introducción*

A lo largo de este capítulo se realizará un estudio de los diferentes componentes que construirán el sistema desde la lectura de los datos hasta la visualización de ellos pasando por diferentes componentes hardware y diferentes métodos de transmisión de datos con sus protocolos de comunicación concretos.

Con toda la información recopilada, posteriormente, se dedicará un espacio a la justificación de los componentes y protocolos elegidos con los que se construirá el sistema de monitorización de datos. De esta manera se definirán conceptos como:

- Procesadores de una sola placa.
- Protocolos de Comunicación entre sensor y módulo central.
- Bases de Datos.
- Proveedores de Servicios Cloud.

Pero, primeramente, en este apartado como introducción se definirán los conceptos de sensor, módulo central, protocolos de comunicación e IoT.

### *2.1.1 ¿Qué es un sensor?*

En una definición muy general: "un sensor es un dispositivo cuyo propósito es detectar eventos o cambios en su entorno y enviar la información a otros componentes del sistema capacitados de mecanismos con los cuales se pueda recoger dichos datos". Un sensor siempre va acompañado de otros componentes hardware para su correcto funcionamiento.

En la siguiente imagen se puede observar un sensor de presión electrónico como ejemplo:

**Ilustración 2-1: Sensor de presión IFM PN3529.**

Los sensores se utilizan, como se cita en un texto, "en tareas cotidianas desde botones de ascensor sensibles al tacto (sensor táctil) hasta lámparas que se atenúan o se iluminan al tocar la base. Por no hablar de innumerables aplicaciones de las que la mayoría de las personas no son conscientes. Con los avances en micro-máquinas y plataformas de microcontroladores fáciles de usar, los usos de los sensores se han expandido más allá de los campos tradicionales de medición de temperatura, presión o flujo, [1] por ejemplo en sensores MARG. Además, sensores analógicos como potenciómetros y resistencias de detección de fuerza todavía se utilizan ampliamente".

Las aplicaciones incluyen fabricación y maquinaria, aviones y aeroespacial, automóviles, medicina, robótica y muchos otros aspectos de nuestra vida cotidiana. Existe una amplia gama de otros sensores que miden las propiedades químicas y físicas de los materiales. Algunos ejemplos incluyen sensores ópticos para la medición del índice de refracción, sensores vibratorios para la medición de la viscosidad de los fluidos y sensores electroquímicos para monitorizar el pH de los fluidos.

El progreso tecnológico permite fabricar cada vez más sensores a escala microscópica como microsensores utilizando tecnología MEMS. En la mayoría de los casos, un microsensor alcanza un tiempo de medición significativamente más rápido y una mayor sensibilidad en comparación con los enfoques macroscópicos [2][3]. Debido a la creciente demanda de información rápida, asequible y confiable en el mundo actual, los sensores

desechables (dispositivos de bajo costo y fáciles de usar para monitoreo a corto plazo o mediciones de disparo único) han ganado recientemente un crecimiento notable.

## *2.1.2 ¿Qué es un módulo central?*

Un módulo central, también conocido como "unidad de control electrónico o ECU" (del inglés electronic control unit), viene definido como "un dispositivo electrónico normalmente conectado a una serie de sensores que le proporcionan información y actuadores que ejecutan sus comandos". Un módulo central cuenta con software cuya lógica le permite tomar decisiones según la información del entorno proporcionada por los sensores y visualizar estos datos de manera que el usuario pueda comprender dichos datos de manera rápida y concisa.

Los módulos centrales son el cerebro del sistema, los cuales cuentan con determinados protocolos de comunicación para transmitir la información entre sus componentes sensores que captan los datos del entorno. Estos componentes a su vez tienen la capacidad de definir el comportamiento de cada sensor; por ejemplo: su frecuencia de muestreo, sensibilidad, rango determinado de datos, etc. Recolectan la información y la transmiten de forma segura mediante diferentes mecanismos a dispositivos más accesibles al usuario, como un *smartphone* o una nube de Internet (IoT).

**Ilustración 2-2: Unidad Central Electrónica Sector Automóvil.**

## *2.1.3 ¿Qué son los protocolos de comunicación?*

Un protocolo de comunicación es bajo su definición "un sistema de reglas que permite que dos o más componentes de un sistema de comunicaciones se transmitan información a través de un tipo de variación de una determinada cantidad física, el cual define las reglas, la sintaxis, la semántica y la sincronización de la comunicación y los posibles métodos de recuperación de errores". Los protocolos generalmente se implementan mediante una combinación hardware y software, aunque puede implementarse de manera individual [4].

Los sistemas de comunicación utilizan formatos completamente definidos para intercambiar paquetes de mensajes. Los protocolos de comunicación deben estar programados en los componentes que compartirán dicha comunicación, para de esta manera llegar a un acuerdo en donde el protocolo se convierta en un estándar técnico.

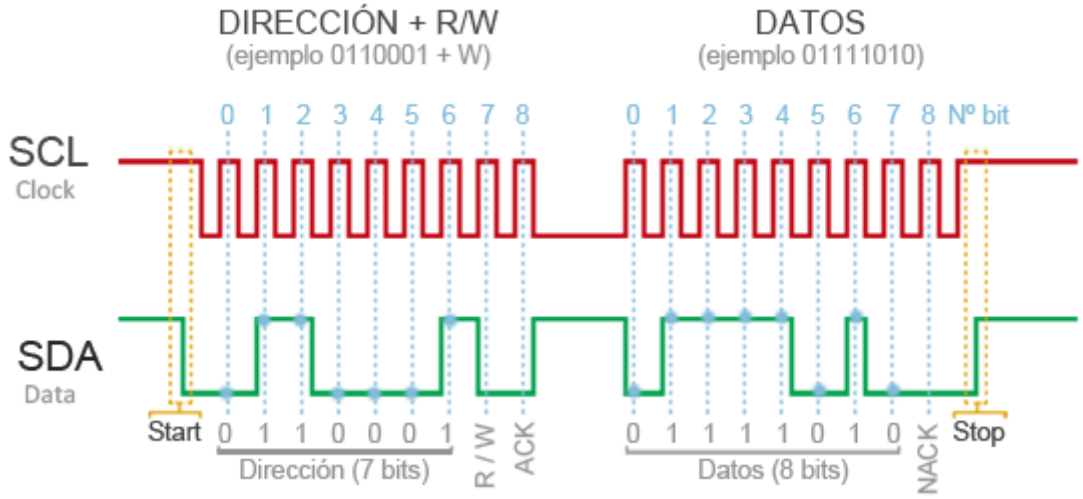


**Ilustración 2-3: Protocolo de Comunicación I²C.**

## *2.1.4 ¿Qué significa Internet of Things o IoT?*

Primeramente, se conoce al Internet de las cosas (del inglés Internet of Things o IoT) como "aquellos sensores y actuadores integrados en objetos físicos que están conectados a través de redes cableadas e inalámbricas, a menudo utilizando el mismo Protocolo de Internet (IP) que conecta Internet".

El término Internet de las cosas tiene por lo menos 16 años, pero la idea real de dispositivos conectados ha existido desde más tiempo atrás, al menos desde la década de los 70. En aquel entonces, la idea a menudo se llamaba "Internet integrado" o "informática

generalizada", pero el término real "Internet de las cosas" se utilizó por Kevin Ashton en 1999 durante su trabajo para la empresa Procter & Gamble. Ashton, que trabajaba en la optimización de la cadena de suministro, quería atraer la atención de la alta dirección hacia una nueva y emocionante tecnología llamada RFID. Debido a que Internet fue la nueva tendencia más candente en 1999 y porque de alguna manera tenía sentido, llamó a su presentación "Internet de las cosas".

Aunque Kevin captó el interés de algunos ejecutivos de P&G (Procter & Gamble), el término Internet de las cosas no obtuvo una atención generalizada durante los siguientes 10 años. Es así que el concepto de IoT comenzó a ganar algo de popularidad en el verano de 2010 cuando se filtró información de que el servicio StreetView de Google no solo había hecho imágenes de 360 grados, sino que también había almacenado toneladas de datos de las redes Wifi de las personas.

En 2011, Gartner, la empresa de investigación de mercado que inventó el famoso "ciclo de las tecnologías emergentes" incluyó un nuevo fenómeno emergente en su lista: "Internet de las cosas".

Al año siguiente, el tema de la mayor conferencia europea sobre Internet, LeWeb, fue "Internet de las cosas". Al mismo tiempo, revistas populares centradas en la tecnología como Forbes, Fast Company y Wired comenzaron a usar IoT como vocabulario para describir el fenómeno.

El término Internet de las cosas alcanzó la conciencia del mercado masivo cuando en enero de 2014 Google anunció la compra de Nest por 3.200 millones de dólares y así llegamos al día de hoy en donde el "Internet de las cosas" es el término más popular para describir este nuevo mundo interconectado [5].


**Ilustración 2-4: Conexiones IoT para un sistema del sector ecológico.**

# 2.2 Procesadores de una sola placa

## 2.2.1 Introducción

Cuando hablamos de procesadores, se define el concepto de procesamiento de datos como la acción de recabar datos y traducirlos a información utilizable. Por ello, los procesadores son aquellos elementos hardware encargados de dicha tarea [6]. Dentro del sistema ocupacional de los procesadores, se pueden definir las siguientes etapas:

1. Recogida de datos.
2. Preparación de datos.
3. Procesamiento.
4. Salida de datos.
5. Almacenamiento de datos.

Hoy en día en el mercado se pueden encontrar diferentes elementos hardware (procesadores) capaces de ejecutar dichas operaciones y que encajen en el concepto de sistema de bajo coste, pero en esta ocasión se expondrán tres de ellos: BeagleBoard, Raspberry Pi y Arduino.

## 2.2.2 BeagleBoard

La BeagleBoard es según su desarrolladora: "una computadora de placa única fabricada por Texas Instruments, presentada en 2008, la cual fue desarrollada por un equipo pequeño de ingenieros y diseñada para hacer prototipos y para ser utilizada como una herramienta educativa para el desarrollo de software de código abierto" [7].

La placa BeagleBoard tiene unas características propias como: "un tamaño aproximado de 75 por 75 mm y pone a disposición todas las funciones básicas de una computadora. El OMAP3530 incluye un ARM Cortex -A8 CPU (que puede funcionar con FreeBSD, RISC OS, Minix, Linux, OpenBSD, o Symbian), un TMS320C64x + DSP para decodificación acelerada de video y audio, y una GPU Imagination Technologies PowerVR SGX530 para proporcionar renderizado 2D y 3D acelerado compatible con OpenGL ES 2.0. La salida de vídeo se proporciona a través de conexiones S-Video y HDMI separadas. Se proporcionan una única ranura para tarjeta SD / MMC compatible con SDIO, un puerto USB On-The-Go, una conexión en serie RS-232, una conexión JTAG y dos conectores estéreo de 3,5 mm para entrada/salida de audio. En cuanto al almacenamiento y la memoria,

están integrados y proporcionan a través de un chip PoP (Package on a Package), 256 MB de RAM (128 MB en modelos anteriores) y 256 MB de memoria flash NAND. En cuanto a la alimentación, la placa utiliza hasta 2 W de potencia y se puede alimentar desde el conector USB o desde una fuente de alimentación de 5 V" [8].

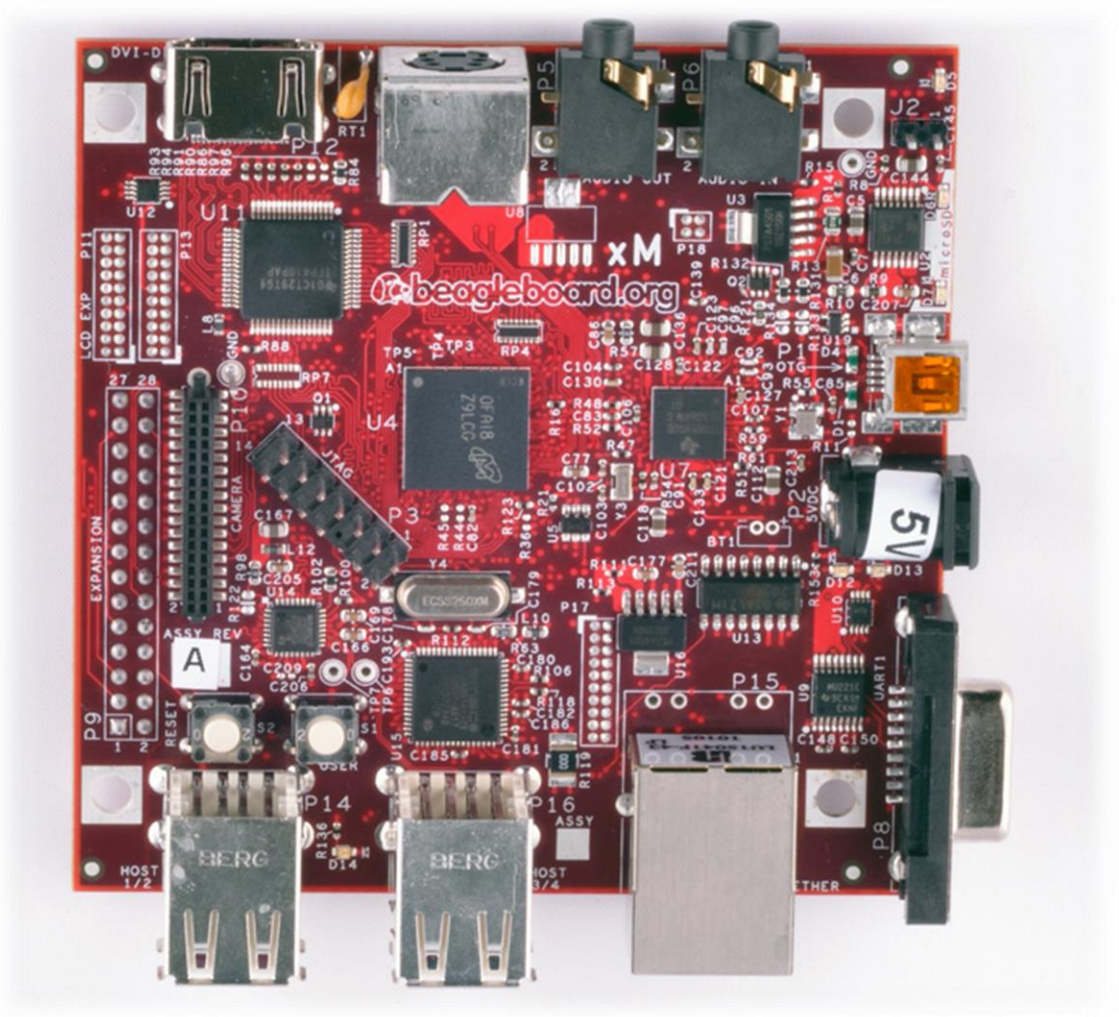


**Ilustración 2-5: Placa BeagleBoard.**

Dentro de la misma desarrolladora, se pueden encontrar las diferentes entregas de placas a lo largo de la historia en base a diferentes prestaciones, como las siguientes:

- BeagleBone Black.
- BeagleBone.
- BeagleBoard -xM.

### *2.2.2.1 BeagleBone*

La placa BeagleBone es según su desarrolladora: "una computadora barebone que fue anunciada a finales de 2011, la cual cuenta con un procesador de 720 MHz llamado Sitara ARM Cortex-A8. A su vez cuenta con una RAM de 256 MB, puerto Ethernet, dos conectores de expansión de 46 pines, ranura micro-SD, un puerto USB y un puerto de dispositivos que incluye conexiones JTAG con depuración de hardware, por lo que, no es necesario un emulador JTAG y un serial de control a bajo nivel" [9].

### *2.2.2.2 BeagleBone Black*

La placa BeagleBone Black fue lanzada el 25 de abril de 2013. Entre otras diferencias respecto a la BeagleBone, incrementa el reloj de procesador a 1 GHz, incrementa la RAM a 512 MB, y añade HDMI y 2 GB de memoria flash eMMC. Dentro de la BeagleBone Black también se encuentra la base kernel de Linux 3.8, que ya contaba la BeagleBone original.

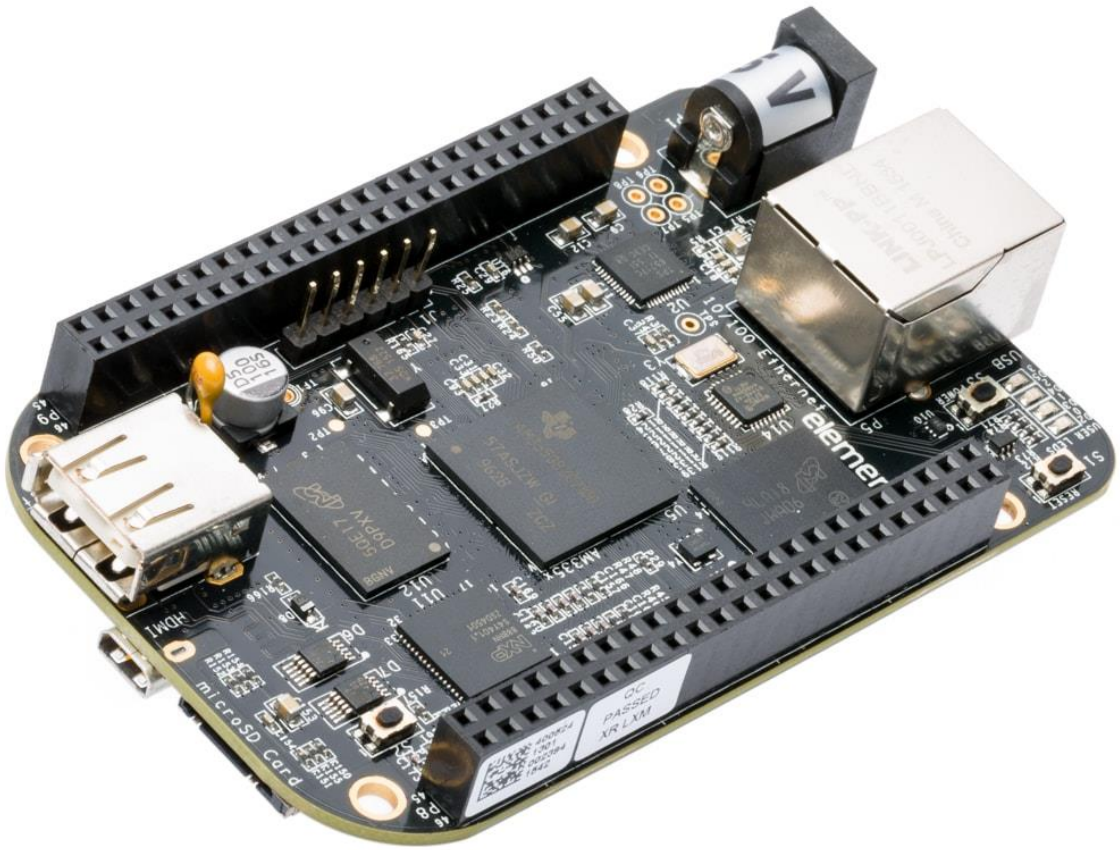


**Ilustración 2-6: Placa BeagleBone Black.**

#### *2.2.2.3 BeagleBoard -xM*

La placa BeagleBoard -xM es una placa a la cual, se le han hecho modificaciones con respecto a la BeagleBoard normal. Esta placa comenzó a distribuirse el 27 de agosto de 2010. El tamaño de la BeagleBoard-xM no supera los 82.55 mm por 82.55 mm y cuenta con una CPU más valoz que su predecesora (la velocidad aumenta de los 720 MHz de la original hasta la velocidad de 1 GHz que tiene esta versión). En cuanto a la memoria, también se nota un cambio. La RAM se incrementa desde los 256 MB hasta los 512 MB de esta versión. A su vez, cuenta con un hub con 4 puertos USB y conector Ethernet. La BeagleBoard-xM elimina la memoria NAND por lo tanto el sistema operativo y otros datos tienen que estar almacenados en una tarjeta micro-SD [10].

### *2.2.3 Raspberry Pi*

La tarjeta Raspberry Pi es un dispositivo con funciones básicas de una computadora, pero de tamaño reducido, la cual está diseñada y fabricada en el Reino Unido con el objetivo inicial de establecer un dispositivo económico para la enseñanza. Desde su lanzamiento, sin embargo, ha crecido mucho más allá del ámbito académico.

Sus orígenes se pueden encontrar en el Laboratorio de Computación de la Universidad de Cambridge en 2006. El científico informático Eben Upton, junto con Rob Mullins, Jack Lang y Alan Mycroft, estaban preocupados porque los estudiantes universitarios de computación entrantes se habían separado de los aspectos técnicos de la computación. Esto se debió en gran parte a los programas de estudios escolares que ponían énfasis en el uso de computadoras en lugar de comprenderlas y a raíz de esta preocupación inicial, se formó la base Raspberry Pi.

Durante los siguientes seis años, el equipo trabajó en el desarrollo de un dispositivo barato y accesible que ayudaría a las escuelas a enseñar conceptos como programación, acercando así a los estudiantes a comprender cómo funciona la informática [11].

Desde su lanzamiento se han ido entregando diferentes versiones de la placa con sus modelos definidos. Así a lo largo de su historia han sido lanzadas las versiones: "Raspberry Pi 1 modelo A, Raspberry Pi 1 modelo B, Raspberry Pi 2 modelo B, Raspberry Pi 3 modelo B, Raspberry Pi 3 modelo B+, Raspberry Pi 3 modelo A y Raspberry Pi 4 modelo B", respectivamente. La última de ellas fue anunciada en junio de 2019.


**Ilustración 2-7: Placa Raspberry Pi 3 Modelo B+.**

Concretamente, la placa Raspberry 3 Modelo B+ fue lanzada en marzo del 2018. Este nuevo dispositivo ofreció un diseño completamente nuevo de la placa a nivel hardware, pero por supuesto, manteniendo la premisa inicial del tamaño reducido, y a su vez, se mantuvieron las mismas posiciones de los elementos que en su modelo anterior (algo importante, ya que siguen valiendo las mismas cajas). Los cambios vinieron a nivel de desarrollo interno, ya que el procesador se cambió por otro más potente que ahora funcionaba a 1.4 GHz, y además inserta mejoras en cuanto a la conectividad como la conectividad "Bluetooth 4.2", "BLE", "Wi-Fi" con banda dual de 2.4 Ghz y 5 Ghz y la tarjeta de red, Gigabit Ethernet mejorada con una tasa de transferencia incrementada hasta los 300 Mbps [12].

### *2.2.4 Arduino*

Arduino dio sus primeros pasos como un proyecto iniciado en el año 2005, el cual estaba enfocado a los estudiantes del Instituto IVREA (IDII), en Ivrea (Italia). Por aquellos tiempos, se usaba el microcontrolador BASIC Stamp, el cual tenía un precio de $100 USD, lo cual suponía un coste elevado para el estudiante medio. Por ello el objetivo era simple; crear una herramienta simple y de bajo coste donde se pudiera crear proyectos digitales sin altos conocimientos técnicos o sin un perfil de ingeniería elevado.

El proyecto al comienzo tenía el nombre de Wiring, el cual constaba de una placa de desarrollo de hardware que a su vez tenía una placa de circuito impreso (PCB) con un

microcontrolador ATmega168. Se desarrollaría dentro de un Ambiente de Desarrollo Integrado (IDE) con una biblioteca de funciones para programar de manera sencilla el microcontrolador. Durante ese mismo año 2005, Massimo Banzi junto con David Mellis y David Cuartielles, colaboraron para insertar el microcontrolador ATmega8 a Wiring, el cual es más económico que el microcontrolador inicial Atmega168, pero en lugar de continuar el desarrollo en Wiring, se separaron del proyecto y lo renombraron Arduino [13].


**Ilustración 2-8: Placa Arduino Uno Rev. 3.**

La placa Arduino Uno es una placa basada en un microcontrolador Atmega328 con características propias como: "6 entradas analógicas, 14 pines de entrada/salida digital (de los cuales 4 pueden ser utilizados para salidas PWM), un resonador cerámico de 16 MHz, una conexión jack para fuente de poder, un conector para USB tipo hembra, un botón reset y un conector ICSP".

Dicha placa cuenta con todo lo esencial para poder operar el microcontrolador, simplemente hay que conectarse al ordenador mediante un cable USB o alimentar la tarjeta mediante una fuente de corriente externa, la cual puede ser un adaptador de corriente o una batería. Para programar la placa se necesita el IDE propiamente desarrollado por Arduino.

## *2.3 Tipos de Protocolos de Comunicación*

### *2.3.1 Introducción*

Junto al desarrollo de nuevas tecnologías y sistemas, la intercomunicación entre diferentes componentes es esencial para el buen funcionamiento del mismo, por lo que,

hablando de transmisión de datos, aquí es donde los protocolos de comunicación han tenido un gran avance durante los últimos años.

Dentro de los protocolos de comunicación, tenemos dos tipos claramente diferenciados:

- Conexión cableada.
- Conexión inalámbrica.

Una red cableada es una red en la que los diferentes componentes del sistema están conectados mediante cables por donde se transmite la información, mientras que una red inalámbrica es una red en la que los diferentes componentes no utilizan un medio físico para la transmisión de datos, sino que se utiliza la modulación de ondas electromagnéticas para transmitir la información.

Como ejemplos de protocolos de comunicación para redes cableadas, se pueden distinguir varios como: $I^2C$, ProfiBus, UART, Ethernet, CanBus, y muchos más. Para las redes inalámbricas también podemos distinguir varios protocolos de comunicación como: Bluetooth, Wifi, ZigBee, Red de telefonía móvil (2G, 3G, 4G o 5G), NFC, entre muchas otras.

Cada tipo de conexión tiene sus pros y sus contras dentro de los objetivos que se busca para un sistema en concreto. La ventaja más notable de una conexión inalámbrica sobre una cableada es la ausencia de cables, lo cual deja margen para el movimiento de los componentes siempre que se encuentren dentro del área de conexión. Aunque a su vez, una comunicación cableada aporta mayor rapidez de transmisión de datos que una conexión inalámbrica, y por no mencionar que las conexiones inalámbricas son menos seguras y robustas. Pero si se busca una transmisión de datos no muy rápida y que provea la facilidad de movimiento, la conexión inalámbrica es la indicada [14].

Para este sistema en concreto se expondrán protocolos de comunicación inalámbricos, como Wifi y Bluetooth.

### *2.3.2 Protocolo de comunicación Wifi*

A día de hoy, no se puede negar que la conexión Wifi ha logrado que podamos comunicarnos y conectarnos con el mundo, estemos donde quiera que estemos. Tanto es así que se superan los13 mil millones de dispositivos que hacen uso de ella, según la Wifi Alliance.

Por lo que no está de más que se exponga un breve repaso de su historia, debido a la gran importancia que tiene esta tecnología en nuestros tiempos, así como los distintos estándares con los que cuenta en la actualidad.

Es curioso mencionar que la invención de la conexión Wifi se le atribuye a Hedy Lamarr, una famosa actriz de cine. Ella compaginaba su trabajo como actriz con el de inventora. Hija de un matrimonio judío, entre medio de todo el conflicto nazi, Lamarr quiso desarrollar tecnología militar para ayudar de alguna forma al gobierno de los Estados Unidos.

Toda esta motivación la llevó a que en 1942 ella patentara lo que llamaría un "sistema de comunicación secreta", el cual a grandes rasgos es lo que vendría a ser hoy en día lo que se conoce como "saltos de frecuencia". Pocos años adelante, concretamente en 1997, IEEE ("Institute of Electrical and Electronics Engineers") creó el primer estándar Wifi, "IEEE 802.11", el cual permitía la transferencia de información a 1 Mbps.

Estas tecnologías fueron la base para comenzar a estandarizar la conexión Wifi, la cual recibió su nombre propiamente: "Wifi", en el año 1999. No tardo más de año cuando se estableció el protocolo IEEE 802.11b, mediante el cual se transferirían los datos a velocidades de 11Mbps, la cual fue la base a partir de la cual surgiría el resto de los estándares que conocemos a día de hoy [15].

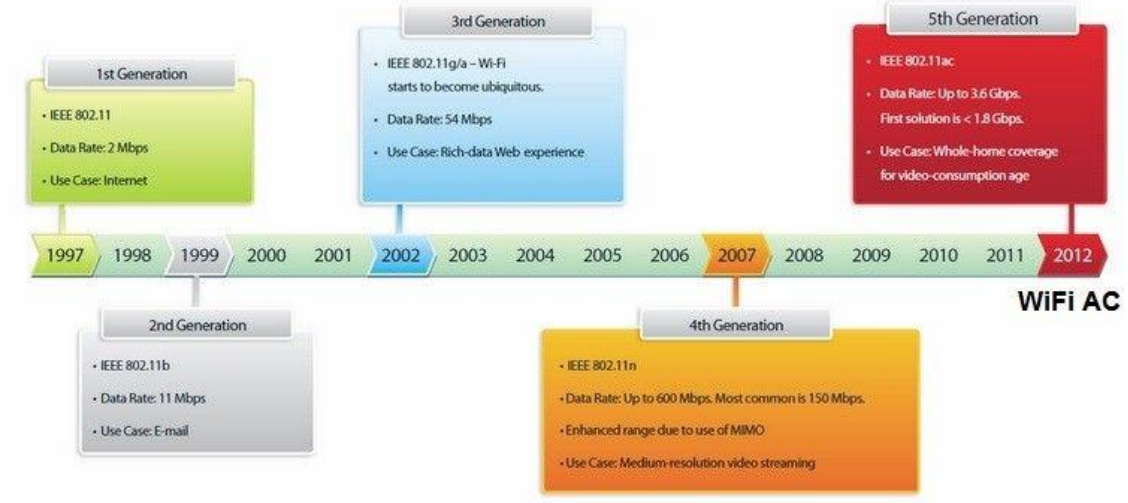


**Ilustración 2-9: Evolución de los estándares Wifi a lo largo de la historia.**

Como se puede observar en la imagen anterior, a lo largo de los años, el estándar Wifi ha ido evolucionando constantemente y así cada vez se ha ido mejorando a sí mismo, ofreciendo cada vez mejor conectividad y mayor rapidez de transmisión de datos, desde 2 Mbps de la primera generación, hasta los 3.6 Gbps (3600 Mbps) de la quinta generación.

En cuanto a sus ventajas se pueden destacar las siguientes [16]:

- Es fácil de implementar y fácilmente escalable según las necesidades del propietario, debido a que, se trata de una red inalámbrica.

- No supone un gasto excesivo en infraestructura si se pretende permitir el acceso a muchos dispositivos.

- La compatibilidad entre dispositivos es total gracias a que es una tecnología mundialmente conocida.

En cuanto a sus desventajas más notables, se pueden destacar las siguiente [16]:

- En cuanto a la calidad de conexión, las redes alámbricas tienen mayor velocidad y no sufren de interferencias y pérdidas de señal.

- En cuanto a la seguridad, existen diferentes softwares capaces de interceptar paquetes de información que viajan por la red, lo que lo hace menos segura.

- No es compatible con otras conexiones inalámbricas ya que el protocolo es distinto.

- Los agentes físicos de nuestro entorno afectarán considerablemente a la señal.


**Ilustración 2-10: Logo del Estándar WiFi.**

## *2.3.3 Protocolo de comunicación Bluetooth*

Según la historia de Bluetooth: "Todo empezó por los años 90 cuando Ericsson se encontraba desarrollando una tecnología que permitiera comunicaciones a corto alcance con el objetivo de utilizar poca energía en los dispositivos (principalmente móviles). Ese proyecto era MCLink".

Al cabo del tiempo, empresas grandes del sector tecnológico como Ericsson, Lenovo, Apple, Motorola, Intel, Nordic, Microsoft, Nokia, Semiconductor y Toshiba, mostraron interés por el producto y formaron una SIG ("Special Interest Group"), lo que viene a ser "un grupo de trabajo conformado por diferentes empresas, quienes aportan capital monetario y humano" [17].

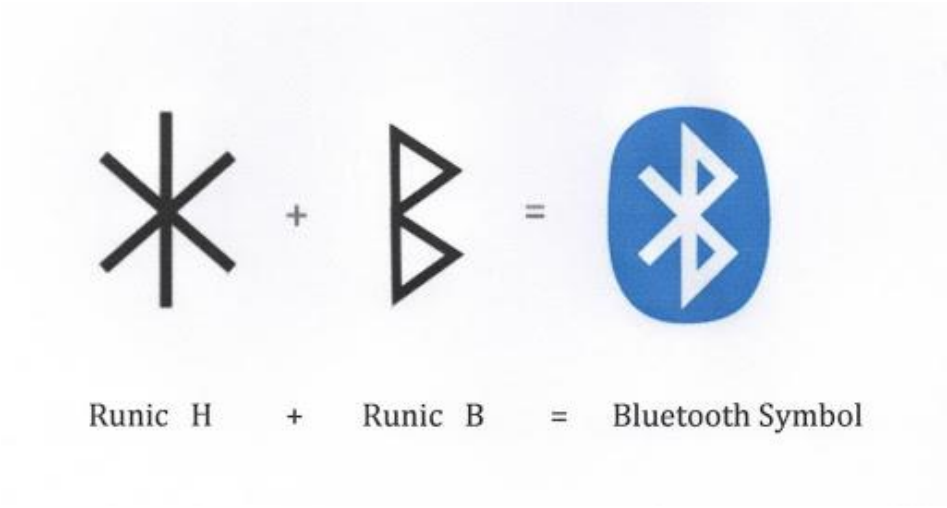


**Ilustración 2-11: Símbolo de Bluetooth (Herald Blåtand = H + B).**

En su primera versión de 1999, la conexión Bluetooth permitía una tasa de transferencia de 0.8 ~ 1Mbps a una distancia de 10 metros. Incluso, no se llegaban a esas prestaciones ya que estas eran las teóricas. Prontamente esa 1.0, cambió a 1.1 y seguidamente a 1.2, la cual fue la más popular.

Posterior a estas versiones, a lo largo del tiempo, Bluetooth fue sacando versiones avanzadas en las cuales se mejoraban las características de sus predecesoras. Después de la versión 1.2, se lanzó la versión 2.0 en 2004, la versión 2.1 en 2007, la versión 3.0 en 2009 hasta la versión 4.0 lanzada en 2010, la cual supone un hito para este protocolo.

La versión Bluetooth 4.0 consta del Bluetooth clásico, pero con un subconjunto dedicado a protocolos de baja energía al cual se le llamaría Bluetooth de baja energía o BLE (del inglés Bluetooth Low Energy) con una pila de protocolo completamente nueva para desarrollar rápidamente enlaces sencillos.

Posterior a este avance, se lanzaron versiones mejoradas como la versión 5.0 en el año 2017, la versión 5.1 en 2019 y como última versión publicada la versión 5.2, la cual fue presentada en enero del año 2020.

En cuanto a sus características generales, se citan las siguientes: "La conexión basada en Bluetooth de baja energía opera a 2.4 GHz (una de las bandas ISM), con una tasa de transferencia de 1 Mbps en la capa física. Está basado en un microchip de bajo coste con

opciones más amplias para su empleo en la industria; además tiene el mismo tamaño de los dispositivos Bluetooth, como los que se encuentra en un teléfono móvil. Tiene soporte para seguridad, ya que emplea el sistema de cifrado AES y esquemas de seguridad configurables" [18].

## 2.4 Bases de Datos

### 2.4.1 Introducción

Cuando se habla de Bases de Datos, como su propio nombre lo indica y se cita: "se habla de un "almacén" que nos permite guardar grandes cantidades de información de forma organizada, la cual luego se puede encontrar y utilizar posteriormente de manera sencilla". El término base de datos fue utilizado por primera vez en 1963 en un simposio celebrado en California, USA, la cual se definió como un conjunto de información agrupada o estructurada [19].

Dentro de sus características más relevantes, se pueden destacar las siguientes:

- "Acceso concurrente por parte de múltiples usuarios."
- "Integridad de los datos."
- "Consultas complejas optimizadas."
- "Seguridad de acceso y auditoría."
- "Respaldo y recuperación."
- "Acceso a través de lenguajes de programación estándar."

Se define un Sistema de Gestión de Base de Datos (SGBD) como "un tipo de software específico, dedicado a servir de interfaz entre la base de datos, el usuario y sus aplicaciones" [19]. Dentro de los diferentes tipos de SGBD, se destacarán en este apartado los tres siguientes: Modelo Entidad-Relación, Modelo Orientado a Objetos y Modelo de Series Temporales.

### 2.4.2 Modelo Entidad-Relación

El modelo Entidad-Relación se define como "un modelo que se basa en la existencia de objetos a los que se les da el nombre de "entidades", y asociaciones entre ellos, llamadas

"relaciones", el cual es el más utilizado para el diseño conceptual de bases de datos, a día de hoy".

Una entidad es a efectos prácticos "cualquier objeto o elemento del cual se pueda almacenar información en la Base de Datos". Las entidades pueden ser concretas como una persona o abstractas como una fecha. Dentro del tipo de entidades, se pueden distinguir dos: entidad débil y entidad fuerte. Una entidad débil es "aquella entidad cuya existencia depende de la existencia de otra entidad", mientras que una entidad fuerte es aquella que su existencia depende de ella misma.

Una relación se define como "la asociación que existe entre dos a más entidades, cuyas identidades involucradas en una determinada relación se denominan entidades participantes". Se le denomina grado de la relación al número de participantes que intervienen en dicha relación. Por ejemplo, la relación "Inquilino-Apartamento" es de grado 2 o binaria, ya que intervienen dos entidades. Dependiendo del número de participantes, existen varios tipos de relaciones, como la relación uno a uno 1: 1, relación uno a muchos 1: N o la relación muchos a muchos N: M [20].

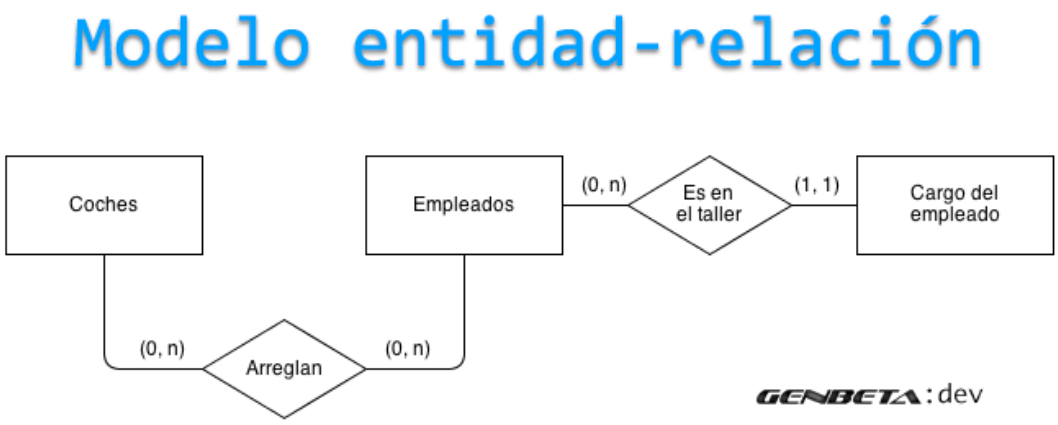


**Ilustración 2-12: Diagrama ejemplo del Modelo Entidad-Relación.**

## *2.4.3 Modelo Orientado a objetos*

Se define el modelo orientado a objetos como "un modelo de datos tanto los datos como sus relaciones están contenidos en una única estructura conocida como objeto" [21].

Dentro de las ventajas que se pueden destacar de este modelo, encontramos los siguientes:

- Los conjuntos de datos complejos pueden guardarse y consultarse de forma rápida y sencilla.

- Los códigos de identificación se asignan automáticamente a cada objeto.
- Funciona bien con lenguajes de programación orientados a objetos.

Por otro lado, dicho modelo también cuenta con una serie de desventaja e inconvenientes que se pueden destacar:

- El uso de las bases de datos orientada a objetos no está muy extendido.
- En algunas situaciones, la gran complejidad puede acarrear problemas de rendimiento.

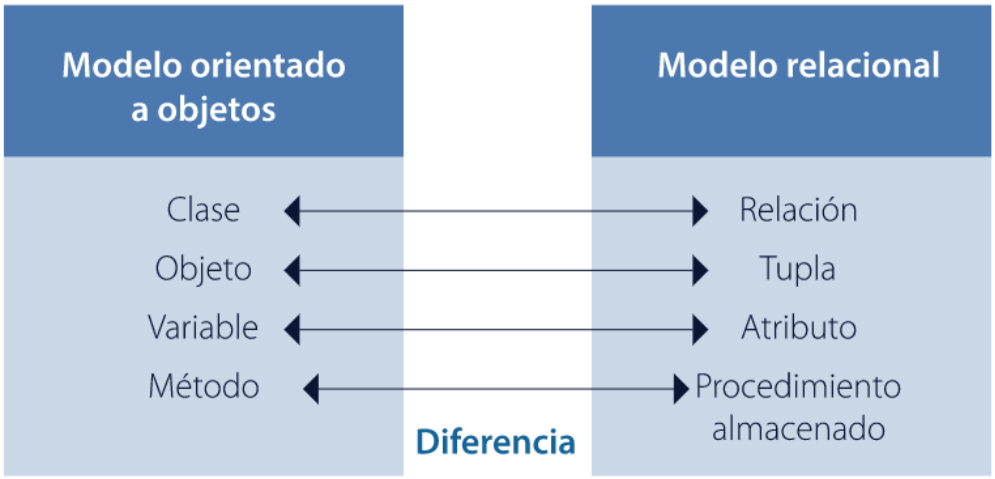


**Ilustración 2-13: Diferencias entre diferentes modelos.**

## *2.4.4 Modelo Series Temporales.*

En la era de los datos, la combinación del tiempo con las bases de datos, ha dado fruto a la popular base de datos de series temporales. Una base de datos de series de tiempo es esencialmente una base de datos distribuidos a lo largo de una marca de tiempo. Es un sistema optimizado para manejar datos o matrices de datos indexados por intervalos de fecha/hora.

En cuanto a sus características temporales, se puede observar la marca de tiempo la cual puede alcanzar niveles de precisión de segundos y milisegundos, aunque en determinados escenarios, incluso puede llegar al nivel de los nanosegundos. Dentro de estas características se ubica la frecuencia de muestreo, la cual se puede definir como el intervalo con el que se indexa un dato a la serie temporal.

En cuanto a los datos, estos se agregan en secuencias de datos y pueden ser multidimensionales correlacionados. Estos datos cubren principalmente eventos, estados y

valores, lo cual hace a este modelo uno muy asequible a sistemas de lectura de datos del entorno con variación temporal [22].

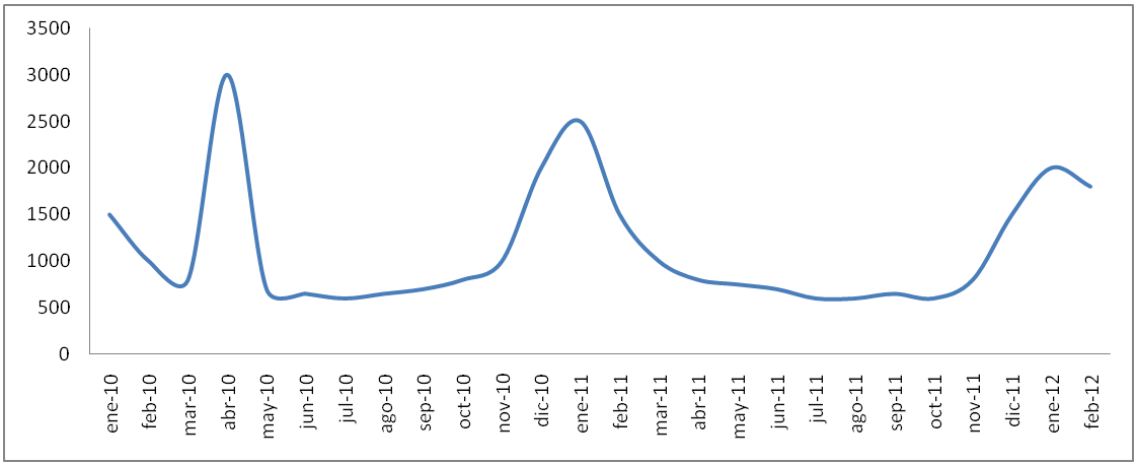


**Ilustración 2-14: Evolución de un valor a lo largo del tiempo.**

## 2.5 Proveedores de Servicios Cloud

### 2.5.1 Introducción

Para hablar de servicios en la nube, definamos primeramente el concepto de nube. La nube es una zona de Internet donde se permite almacenar y acceder a datos en lugar de un convencional disco duro. En realidad, el concepto de nube es metafórico, ya que se le llama así a una gigantesca infraestructura de servidores de Internet que aceptan conexiones y reparten información mientras flota.

Para acceder a dichos datos y utilizarlos basta con tener conexión a Internet, y de esta manera los datos se hacen visibles como si de un disco duro físico se tratara, pero con la ventaja de que no ocupan espacio físico cercano al usuario.

Grandes empresas ya hacen uso de ellas dentro de sus aplicaciones. Empresas como: Facebook, Google, Twitter, Pinterest, etc. En todos estos casos, como cuentas Gmail o Hotmail, hay datos guardados de correos, ya sean mensajes, fotos o vídeos los cuales se encuentran almacenados en sus servidores y son accesibles para el usuario.

Por supuesto, este método supone un gran número de ventajas en cuanto al almacenamiento de datos como el acceso desde cualquier sitio y con varios dispositivos que tengan acceso a Internet. Todo el software y datos están en un mismo sitio lo que conlleva a un ahorro significativo en software y hardware.

En cuanto a las desventajas más notables de los servicios en la nube, es principalmente aquello que lo hace fuerte: la conexión a Internet, ya que, sin ella, no se puede acceder a los datos. Lo bueno es que, con el paso del tiempo, este problema cada vez es menor debido al desarrollo constante en cuanto a conexión a Internet. Hoy en día es muy asequible tener una conexión decente y estable a Internet [23].

**Ilustración 2-15: Diagrama de dispositivos conectados a una nube.**

Dentro de los diferentes proveedores de Servicios en la nube, destacaremos dos de ellos, los cuales pueden servir para monitorizar datos desde cualquier parte del mundo, bastando con una conexión estable a Internet. Los dos proveedores de los cuales se hablará son: ThingSpeak y Xively.

### *2.5.2 ThingSpeak*

Definido por las propias palabras de sus desarrolladores, "ThingSpeak es un servicio de plataforma analítica de IoT que permite agregar, visualizar y analizar flujos de datos en vivo en la nube, mediante el cual se pueden enviar datos a ThingSpeak desde cualquier dispositivo, crear visualizaciones instantáneas de datos en vivo y enviar alertas usando servicios web como Twitter y Twilio".

Con la analítica de MATLAB dentro de ThingSpeak, se puede escribir y ejecutar el código de MATLAB para realizar preprocesamientos, visualizaciones y análisis. La construcción de prototipos y sistemas de IoT sin necesidad de configurar servidores o desarrollar software específico ahora es asequible para los ingenieros y científicos gracias a ThingSpeak [25].

ThingSpeak

**Ilustración 2-16: Logo de ThingSpeak.**

Dentro de las características principales de ThingSpeak, se pueden destacar las siguientes, citadas por el propio desarrolador:

- "Configurar fácilmente los dispositivos para que envíen datos a ThingSpeak utilizando los protocolos de IoT más populares."
- "Visualizar los datos de los sensores en tiempo real."
- "Utilizar la potencia de MATLAB para dar sentido a tus datos de IoT."
- "Ejecutar análisis de IoT automáticamente en base a horarios o eventos."
- "Prototipo y construcción de sistemas de IoT sin necesidad de crear servidores o desarrollar software web."

En resumen, propiamente dicho por su desarrolladora: "con ThingSpeak, el usuario puede crear aplicaciones de registro de datos de sensores, aplicaciones de seguimiento de ubicación y una red social para los objetos conectados, con actualizaciones de estado".

### *2.5.3 Xively*

Xively (anteriormente conocido como Cosm and Pachube) es una plataforma de Internet de las cosas (IoT) propiedad de Google. Xively ofrece a las empresas o usuarios una forma de conectar productos, administrar dispositivos conectados y los datos que producen e integrar esos datos en otros sistemas.

Todos los dispositivos se pueden conectar a Xively Cloud para procesarlos en tiempo real y archivarlos en la nube. Los desarrolladores de aplicaciones de IOT pueden escribir la interfaz para las aplicaciones de IoT según sus requisitos. Esto ayuda en la administración conveniente de aplicaciones con Xively Cloud y otras API.

Xively es muy popular entre las empresas que se ocupan de la fabricación y el desarrollo de dispositivos basados en IoT. Las empresas que utilizan Xively pueden confiar en la conectividad segura de los dispositivos, así como en la capacidad de gestión de datos perfecta. [25]

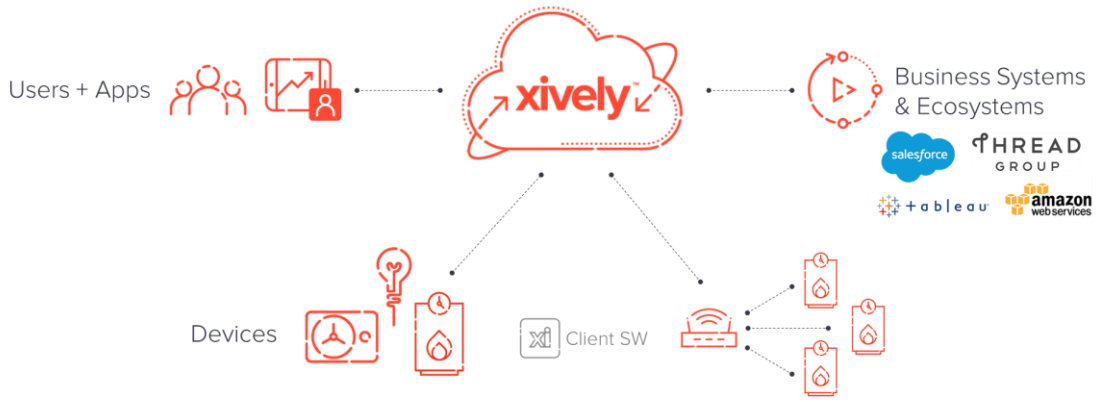


**Ilustración 2-17: Diagrama de dispositivos conectados a una nube Xively.**

Las bibliotecas Xively Python también se pueden usar para incrustar código Python según las API de Xively. Se proporciona una interfaz web de Xively para su fácil implementación. Xively también viene con soporte para múltiples idiomas y plataformas. Se pueden implementar protocolos HTTP, API, MQTT, lo cual hace que la conectividad del dispositivo sea mucho más fácil con Xively Cloud.

## *2.6 Conclusiones*

Dentro de este capítulo se ha dado un repaso a los diferentes componentes hardware, tecnologías software y protocolos de comunicación existentes a día de hoy con el fin de tener una visión amplia del entorno tecnológico del que se dispone para realizar el sistema de monitorización de datos de bajo coste propuesto para este proyecto.

En relación a las placas de desarrollo se propusieron tres opciones las cuales eran: BeagleBoard, Raspberry Pi y Arduino. Para la realización de este proyecto, se tomará la placa Raspberry Pi como módulo central de recepción de datos, debido a su buena relación calidad-precio, lo cual cumple con la especificación de sistema de bajo coste. Por lo tanto, la placa BeagleBoard queda descartada por su precio elevado.

La placa Arduino es la más barata de las opciones, pero Raspberry cumple con mayores características, aunque su precio sea más elevado. Arduino cuenta con una velocidad de procesamiento de datos de 16 MHz mientras que Raspberry llega hasta los 700 MHz. En cuanto al tamaño de la RAM, Arduino cuenta con 2 KB mientras que Raspberry cuenta con 256 MB. A parte de especificaciones notablemente mejores en cuanto a placa, Raspberry cuenta con la distribución de Linux, lo cual la hace más potente en cuanto a programación para la distribución de datos a servidores iCloud y para la gestión de bases de datos.

En cuanto al método de transmisión de datos desde el sensor hasta el módulo central, se ha elegido la conexión mediante Bluetooth de baja energía (BLE) debido a su bajo

consumo, lo cual concuerda con un sistema de bajo coste. No se ha tomado en cuenta conexiones tipo cableada debido a que se pretende poner el sensor en zonas de recogida de datos donde la facilidad de movimiento es considerable.

A parte de por el bajo consumo que supone mantener la comunicación mediante BLE, para la recogida de datos simples como lectura de temperatura en un sector de agricultura no se necesitan paquetes de datos muy elevados, por lo que no se necesita de una elevada transmisión de datos para poder llevarlo a cabo; mediante BLE sería suficiente. Todo ello sumado con la mayor facilidad de programación que requiere establecer comunicación BLE comparado con la conexión Wifi, se ha optado por establecer la transmisión de datos mediante Bluetooth de Baja Energía.

Posteriormente, dichos datos quedarán almacenados en una base de datos y de esta manera poder tener acceso a ellos para su utilización cuando sea requerida. En cuanto al Sistema de Gestión de Base de Datos se dispondrá de un modelo de Series Temporales, debido a la característica temporal de la recogida de datos, ya que lo que se pretende es visualizar la evolución de dichos datos a lo largo del tiempo, marcando una frecuencia de muestreo.

Estos datos quedarán reflejados en un servicio en la nube mediante ThingSpeak. La elección de ThingSpeak sobre Xively se ha basado en la tipología de dichos datos. Los datos que se pretenden almacenar son de tipo numérico, por lo que se dispone de toda la potencia de cómputo que otorga Matlab para los datos que se almacenan en una nube de ThingSpeak. En añadido a esto, la documentación referente al desarrollo de servicios con ThingSpeak ha sido más clarificadora comparado con los servicios cloud de Xively.

Mediante todo el estudio hecho en este capítulo con las elecciones definidas en este apartado, se procede a entrar en detalle de cada uno de estos componentes y servicios describiendo el sistema de manera general.

# Capítulo 3

# Descripción del sistema

## *3.1 Introducción*

A lo largo de este capítulo, se expondrá en más detalle la configuración usada para los diferentes componentes, protocolos y servicios utilizados para la realización de este proyecto, sin olvidar que el objetivo que se pretende conseguir es una monitorización de parámetros tomando datos de un entorno enfocado a la agricultura de precisión.

Para ello, se tomarán cada una de las elecciones anteriormente planteadas y con ello se preparará el terreno sobre el cual se volcará todo el desarrollo del sistema, tanto a nivel de hardware como a nivel de software.

Cabe recalcar que a lo largo de este proyecto no se tomará en cuenta el desarrollo por parte del sensor, ya que se centrará el trabajo en la monitorización y utilización de los datos provistos por el mismo. Como consecuencia de ello, para su correcta implementación, se ha tomado una simulación de un sensor mediante un microcontrolador que genere números los cuales puedan ser transferidos y leídos por parte del módulo central. El microcontrolador encargado de dicha tarea es el ESP32 de Espressif Systems, del cual a continuación se expondrán sus características y la justificación de su elección.

## *3.2 Microcontrolador ESP32 como sensor*

El microcontrolador ESP32 es, según su fabricante: "la denominación que se le da a una familia de chips SoC (del inglés System-on-Chip) de bajo coste y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada, el cual emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía". Espressif Systems es la encargada de crear y desarrollar el microcontrolador ESP32, aunque es fabricado por TSMC utilizando su proceso de 40 nm. A su vez, es un sucesor directo del ESP8266, otro SoC.

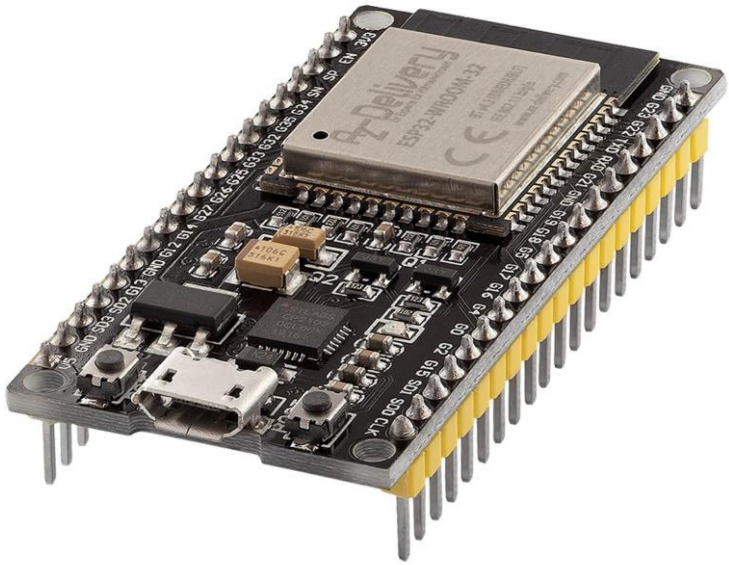


**Ilustración 3-1: Placa de desarrollo ESP32-DevKitC.**

Dentro de las características del procesador, con este microcontrolador se obtiene velocidades de procesamiento de 160 o 240 MHz, rindiendo hasta 600 DMIPS con un co-procesador de ultra baja energía (ULP). En cuanto a la memoria, se consta de 520 KiB de SRAM. El ESP32 es muy asequible cuando se pretende utilizar dicho controlador para aplicaciones de IoT, ya que cuenta con conectividad Wifi, estándar 802.11 b/g/n, y con conectividad Bluetooth v4.2 BR/EDR y BLE (Bluetooth de Baja Energía).

En cuanto a sus características periféricas, según sus especificaciones: "tiene interfaces de 12-bit SAR ADC de hasta 18 canales, 2 x 18-bit DACs, 10 sensores de tacto (capacitivos GPIOs), 4 interfaces SPI, 2 interfaces I2S, 2 interfaces I2C, y 3 interfaces UART. También cuenta con controladores host SD/SDIO/CE-ATA/MMC/eMMC, controlador

esclavo SDIO/SPI, interfaz Ethernet MAC con soporte para el protocolo IEEE1588, un bus CAN 2.0 y muchas otras prestaciones" [26].

Dentro de las placas ESP32 se pueden encontrar varios kits de desarrollo lanzados por Espressif los cuales cada uno cuentan con sus propias características. Los más conocidos son ESP32-DevKitC, ESP32-DevKitM-1, ESP-WROVER-KIT-VE, ESP32-PICO-KIT, ESP32-LyraT y ESP32-LCDKit. Para la realización de la simulación del sensor, se ha escogido la placa ESP32-DevKitC la cual es una placa de desarrollo de nivel de entrada. Cuenta con todos los pines del módulo ESP32 expuestos y es fácil de conectar y usar. El tipo de módulo que utiliza en concreto es la versión ESP32-WROOM-32 [27].

## *3.2.1 Entorno de Desarrollo ESP32*

El entorno de desarrollo se pude definir como aquel programa informático que permite la utilización de diferentes métodos y acciones para la programación de un microcontrolador. A la hora de programar el microcontrolador ESP32, se pueden encontrar varias opciones, dependiendo del nivel de abstracción que se requiera.

Para este proyecto se utilizará la herramienta provista por Arduino para programar sus placas, Arduino IDE, que, con una serie de configuraciones, se puede programar el microcontrolador ESP32. El software Arduino IDE de código abierto facilita la escritura de código y su carga en la placa.

### *3.2.1.1 Instalación del entorno Arduino IDE*

Para la correcta instalación del entorno Arduino IDE, hay que desplazarse hasta la página oficial de Arduino y dentro de su apartado Software y Downloads, se encuentra el archivo de instalación del mismo.

La última versión estable del software es la versión Arduino IDE 1.8.13, la cual se procede a su instalación. Como cualquier programa software en un entorno Windows, se siguen los siguientes pasos:

- Se aceptan los términos y condiciones de la licencia.


**Ilustración 3-2: Ventana Términos y Condiciones Arduino IDE.**

- Se seleccionan todos los complementos y drivers necesarios.

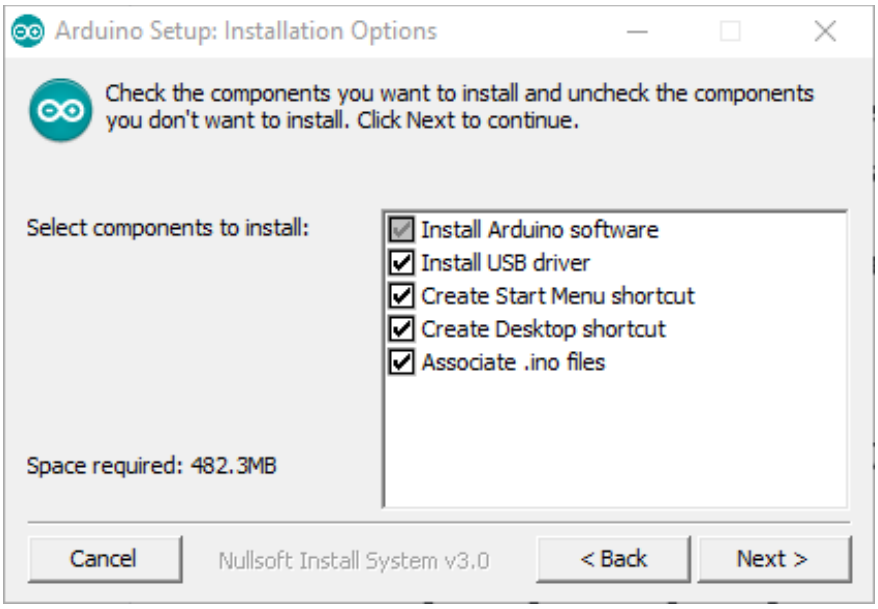


**Ilustración 3-3: Ventana Complementos y Drivers Arduino IDE.**

- Posteriormente, se selecciona la ruta de instalación y se presiona "install".

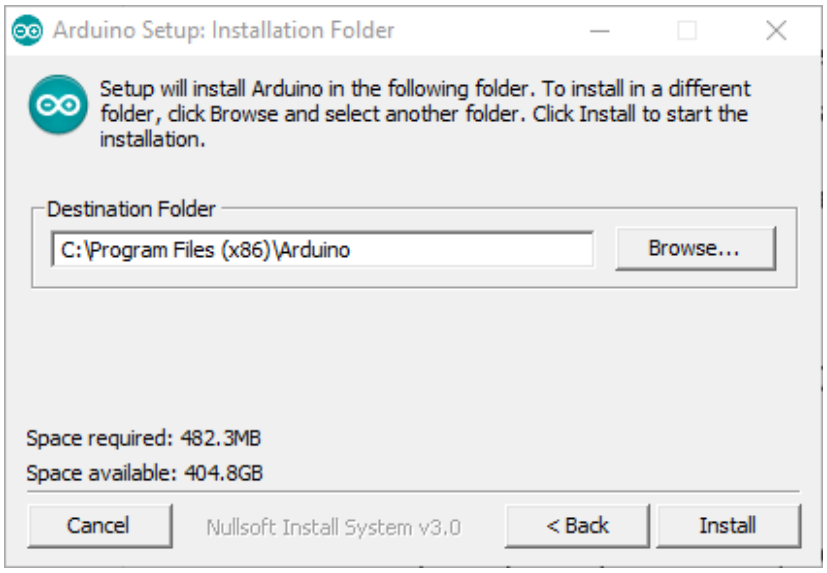


**Ilustración 3-4: Ventana Ruta de Instalación Arduino IDE.**

Al momento, la instalación comenzará a realizarse dentro del sistema Windows, por lo que tan solo hay que esperar unos minutos hasta que termine y se complete toda la instalación.

### *3.2.1.2 Instalación del plugin ESP32 Add-on*

Este complemento al entorno de desarrollo Arduino IDE, permitirá que el entorno pueda comunicarse con la placa de desarrollo ESP32, y de esta manera, se pueda programar utilizando dicho IDE. Para su correcta instalación, los pasos a seguir son los siguientes:

- Se abre el entorno de desarrollo Arduino IDE y en las opciones desplegables, de la parte superior, se selecciona "File > Preferences".

- Dentro de este apartado, se selecciona el recuadro "Additional Boards Manager URLs" dentro de la sección "Settings", y se coloca la siguiente dirección https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json.

- Una vez hecho esto, se da a "OK" para aplicar los cambios realizados dentro del entorno.

- Posteriormente, hay que abrir el apartado de "Tools > Board > Boards Manager…", en donde si se busca "ESP32", aparecerá un módulo el cuál hay que instalar para que el entorno se comunique correctamente con la placa ESP32.

### *3.2.1.3 Creación de un proyecto*

Para realizar un test de que todo el entorno está correctamente configurado y listo para trabajar sobre la placa, primeramente, hay que mirar dentro de "Tools > Board" donde debería aparecer nuestro dispositivo ESP32 como una opción de conexión.

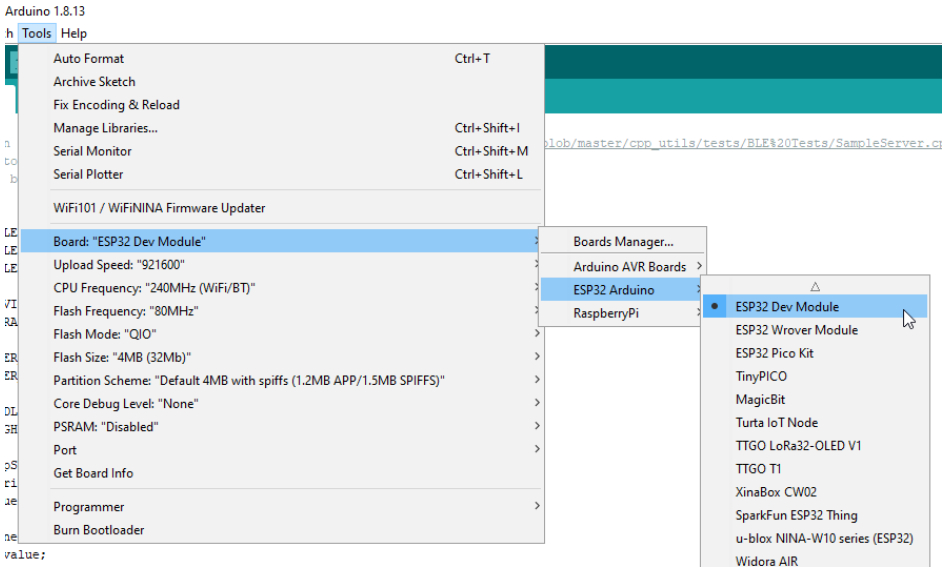


**Ilustración 3-5: Dispositivo ESP32 como placa disponible.**

Para poder programar y flashear la placa ESP32, el kit debe estar conectado a un puerto USB del ordenador mediante el cual, el programa flasheará el microcontrolador. Una vez conectado, se observa que dentro de "Tools > Port", está el puerto correcto seleccionado, y después de esto, el entorno podrá compilar, flashear y programar el microcontrolador sin ningún problema.

Simplemente para abrir un nuevo proyecto y comenzar a programar, nos dirigimos a "File > New", y se podrá escribir un código para nuestra placa ESP32.

## *3.3 Módulo central Raspberry Pi*

Como se planteó en el capítulo anterior, se partirá de un módulo central para la gestión de los datos recolectados del sensor (ESP32). Para esta tarea, se escogió una placa de desarrollo Raspberry Pi por sus enormes prestaciones para aplicaciones IoT y su bajo coste.

Dentro de las diferentes versiones lanzadas por la compañía, la placa que se utilizará para la realización de este trabajo será la placa Raspberry Pi 3 Modelo B+. Dicha placa consta con lo necesario para poder llevar a cabo el proyecto. Cuenta con conexiones inalámbricas robustas y programación dentro de un sistema basado en Linux.

Dicha placa cuenta a su vez con una entrada para una tarjeta micro-SD donde estarán guardados todos los datos correspondientes al sistema, dígase, el sistema operativo, archivos temporales, archivos de desarrollo, procesos del sistema, configuraciones de entorno, y demás datos. Por lo que primeramente para hacer funcionar correctamente la placa, se ha de insertar en la tarjeta micro-SD un sistema operativo que Raspberry Pi entienda y pueda correr.

Para realizar dicha tarea, nos desplazamos a la página oficial de Raspberry y dentro del apartado "Software" (https://www.raspberrypi.org/software), se encuentra la herramienta dada por Raspberry para grabar su sistema operativo dentro de la tarjeta micro-SD.

Una vez descargado el programa, simplemente hay que seguir los pasos escogiendo el sistema operativo que se quiere grabar en la Raspberry y la ranura micro-SD donde se quiere grabar. Para este proyecto se escogió la opción más utilizada y recomendada por Raspberry, Raspberry Pi OS de 32-bit, la cual está basada en Debian, que es una extensión de Linux.


**Ilustración 3-6: Aspecto de la herramienta de grabación Raspberry.**

Una vez realizado este paso, se inserta la tarjeta micro-SD dentro de la ranura apropiada en la tarjeta Raspberry y se alimenta la placa. Monitorizando con una pantalla mediante la conexión HDMI que dispone la tarjeta, se podrá observar el progreso de configuración de la placa y su inicio. Una vez dentro, todo estará listo para la programación de la tarjeta.

## 3.4 Protocolo de Comunicación Bluetooth en la tarjeta Raspberry Pi

Como se expuso en el capítulo 2 de este proyecto, se decantó para la realización del mismo un sistema de recogida de datos basado en el protocolo Bluetooth debido a su menor peso de programación y su menor consumo. Dentro de las dos variantes de Bluetooth, concretamente se desarrollará la comunicación mediante el protocolo Bluetooth Low Energy (BLE o Bluetooth de Baja Energía) debido a su consumo mínimo.

Para la realización de esta tarea, tanto sensor (microcontrolador ESP32) y módulo central (tarjeta Raspberry) tienen que tener las configuraciones listas para poder establecer comunicación mediante dicho protocolo.

Por parte del ESP32, tan solo habría que escribir código relacionado con el protocolo para poder establecer qué rol lleva y qué acciones realizará, mientras que por el lado de la Raspberry hay más configuraciones y paquetes de instalación que hay que instalar dentro de su sistema operativo.

El código escrito en el proceso que realizará la tarjeta Raspberry será del tan conocido lenguaje de programación Python. Es por ello que se utilizarán las librerías de Bluetooth desarrolladas con Python para su implementación. A esta API o conjunto de librerías se le denomina "BluePy".

La instalación de BluePy requiere de unos ciertos pasos para comenzar a utilizarlo dentro de la tarjeta. Primeramente, se abre un terminal de comandos y se escribe la siguiente línea:

➔ sudo apt-get install python-pip libglib2.0-dev

Con ella lo que se hace es actualizar si está instalada la extensión que nos permita programar con Python, o si no, la instala completamente, y a su vez, también instala las librerías referentes a GLib 2.0, que son necesarias para la correcta implementación del protocolo BLE.

Seguidamente, insertamos el comando:

➔ sudo pip install bluepy

Mediante este comando las librerías referentes a BluePy quedarán listas para ser utilizadas dentro de cualquier archivo de proceso que se genere en la tarjeta Raspberry. Tan solo hay que recordar insertar las librerías en el código llamando a BluePy y a su extensión para BLE, escribiendo la línea "from bluepy import btle".

## 3.5 Servicios de Bases de Datos en la tarjeta Raspberry Pi

Como se expuso durante el capítulo 2, de los tres tipos de gestión de bases de datos expuestos, se tomó el modelo de Series Temporales, debido a la característica temporal intrínseca de la recogida de datos del sistema. En este caso, la tarjeta Raspberry después de recoger los datos del sensor mediante BLE, procederá a insertar dicho dato en la base de datos, para su posterior utilización o monitorización.

Esta base de datos constará de dos simples características, una de ellas es el valor en sí, y otra es la característica temporal, con la que se podrá observar la evolución del mismo a lo largo del tiempo en un diagrama temporal. Dicho diagrama servirá para controlar el estado de la magnitud física que esté leyendo el sensor, para su posterior interpretación.

Efectivamente, dentro de los sistemas de gestión de base de datos basados en series temporales se pueden encontrar varios, pero para la realización de este trabajo, se decidió programar el almacenamiento de la base de datos mediante InfluxDB.

Con ánimo de validar que la información se esté guardando de manera apropiada dentro de la base de datos, se instala también un servicio de consulta de datos mediante Internet llamado Grafana.

### 3.5.1 InfluxDB

InfluxDB según su definición: "es una base de datos de series de tiempo de código abierto (TSDB) desarrollada por InfluxData, la cual está escrita en Go y optimizado para un almacenamiento y recuperación rápidos y de alta disponibilidad de datos de series temporales en campos como el monitoreo de operaciones, métricas de aplicaciones, datos de sensores de Internet de las cosas y análisis en tiempo real".

Las consultas continuas se ejecutan periódicamente y almacenan los resultados en una medición objetivo [28].


**Ilustración 3-7: Logo comercial de InfluxDB.**

#### *3.5.1.1 Instalación de InfluxDB en la tarjeta Raspberry Pi*

Para su utilización en la tarjeta, primeramente, se instalarán los paquetes necesarios dentro del sistema operativo de Raspberry, mediante el cual se pueda acceder al servicio de gestión de base de datos y su correcta programación.

Primeramente, se añaden los repositorios de influx a apt mediante los comandos:

- "wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add – "
- "source /etc/os-release"
- "echo "deb https://repos.influxdata.com/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/influxdb.list"

Posteriormente, actualizamos apt con los nuevos repositorios añadidos e instalamos en el mismo comando influxdb.

- "sudo apt update && sudo apt install -y influxdb"

Después configuramos el sistema para poder correr el servicio influx mediante los comandos:

- "sudo systemctl unmask influxdb.service"
- "sudo systemctl start influxdb"
- "sudo systemctl enable influxdb.service"

Ahora se podrá correr el servicio influxdb simplemente mediante el comando "influx".

### *3.5.2 Grafana*

Grafana es un según su definición: "un software libre basado en licencia de Apache 2.0, que permite la visualización y el formato de datos métricos además de permitir crear cuadros de mando y gráficos a partir de múltiples fuentes, incluidas bases de datos de series de tiempo como Graphite, InfluxDB y OpenTSDB".

Según Wikipedia: "Grafana es multiplataforma sin ninguna dependencia y también se puede implementar con Docker. Está escrito en lenguaje Go y tiene un HTTP API completo. Además de administrar cuadros de mando clásicos (adiciones, eliminaciones, favoritos), Grafana ofrece compartir un cuadro de mando actual mediante la creación de un

enlace o una instantánea estática del mismo. Todos los paneles de control y las fuentes de datos están vinculados a una organización, y los usuarios de la aplicación están vinculados a organizaciones a través de roles. Grafana evita que los usuarios sobrescriban accidentalmente un panel de control. Existe una protección similar cuando se crea un nuevo panel de control cuyo nombre ya existe. La herramienta ofrece la posibilidad de configurar alertas, si es necesario" [29].


**Ilustración 3-8: Logo comercial de Grafana.**

### *3.5.2.1 Instalación de Grafana en la tarjeta Raspberry Pi*

Como en el caso anterior de InfluxDB, para la correcta utilización del servicio de Grafana, todos los paquetes relacionados con este servicio deben ser instalados dentro del sistema operativo.

Primeramente, añadimos los paquetes de Grafana al apt.

- "wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add – "
- "echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list"

Actualizamos apt con los nuevos repositorios añadidos e instalamos Grafana.

- "sudo apt update && sudo apt install -y Grafana"

Después de su instalación, configuramos el inicio de Grafana tal y como lo hicimos con InfluxDB.

- "sudo systemctl unmask grafana-server.service"
- "sudo systemctl start grafana-server"

➔ "sudo systemctl enable grafana-server.service"

Posteriormente, se necesita crear un perfil dentro de InfluxDB para poder acceder a él desde Grafana cuando se necesite visualizar los datos que están siendo almacenados dentro de la base de datos. Para ello, dentro de influxdb, el cual se puede correr simplemente escribiendo el comando "influx", se crea una base de datos la cual se llamará "home" y se la utiliza para crear un perfil llamado "grafana" con todos los privilegios. Por lo que dentro de influx se escriben los siguientes comandos:

➔ create database home

➔ use home

➔ create user grafana with password 'admin' with all privileges

➔ grant all privileges on home to grafana

Ahora la placa Raspberry está preparada para poder almacenar datos en series temporales mediante InfluxDB y Grafana.

## *3.6 Servicios de Cloud en la tarjeta Raspberry Pi*

Tal y como se dijo en la conclusión del Capítulo 2, como servicio Cloud se optará por ThingSpeak debido al tipo de dato numérico que se emplea, para sus futuras implementaciones gracias a la potencia matemática de Matlab. Con Matlab todos los datos recibidos mediante HTTP a la nube, pueden ser procesados y utilizados para controlar mejor las operaciones con dichos datos, o crear nuevas implementaciones a partir de ellos.

Todos aquellos datos recolectados vía BLE desde el sensor y recibidos en la Raspberry, serán transferidos a la base de datos mediante InfluxDB, pero el proceso no se queda ahí. Estos datos almacenados en la base de datos de influx, se volcarán a su vez en la nube de ThingSpeak mediante el protocolo HTTP. Una vez allí, el usuario final con su cuenta de ThingSpeak, podrá acceder al historial de datos y ver la evolución del mismo desde cualquier dispositivo que tenga conexión a Internet.

Para comenzar a operar con ThingSpeak, es necesario que el usuario tenga una cuenta activa dentro del sistema de ThingSpeak. Con esta cuenta podrá monitorizar sus propios paneles y personalizarlos.

### *3.6.1 Establecimiento de cuenta de ThingSpeak*

Lo primero de todo para poder iniciar con ThingSpeak es crearse una cuenta asociada a ellos. Para ello, tenemos que dirigirnos a la página oficial de ThingSpeak y crearnos una cuenta insertando cualquier correo que tengamos a disposición con los datos del usuario en concreto.

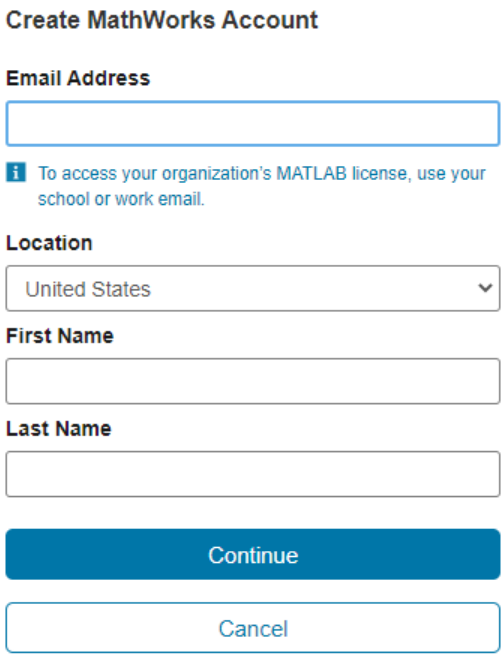


**Ilustración 3-9: Creación de cuenta en ThingSpeak.**

Después de crear la cuenta, se ingresa dentro de la plataforma de ThingSpeak, donde se puede observar un apartado denominado "Channels". Al ser una cuenta nueva, no se dispondrá de ningún canal creado, por lo que se verá un icono denominado "New Channel" para poder crear el primer canal de datos.

Dentro de las propiedades del canal a la hora de crearlo, se pueden observar las siguientes como:

- Name: En este apartado se ingresa el nombre del canal.
- Description: En este apartado se ingresa una pequeña descripción del canal si así lo requiere el usuario.

- Field 1, 2, 3…: Los campos denominados field son campos donde se pueden insertar el número de tipo de datos que se insertarán en la nube. En el caso que se aplica actualmente, solo se necesita uno.

Una vez se tiene el canal creado, dentro del apartado API Keys se podrá observar la llave con la que se puede acceder a la escritura de los datos en la nube. Esta llave se utilizará posteriormente en la programación de la transmisión de datos para llevarlos desde la Raspberry hasta la nube mediante el protocolo HTTP.

## *3.7 Conclusiones*

En este capítulo se ha hecho un recorrido del sistema completo pasando por todos los mecanismos y protocolos que se utilizarán para conseguir alcanzar el objetivo de monitorización de datos. En resumen, se parte del sensor (ESP32, con la generación de números aleatorios) el cual transmitirá mediante BLE dichos datos al módulo central Raspberry Pi, el cual recogerá los datos del ESP32.

La tarjeta Raspberry Pi procesará estos datos de manera que los almacenará en una base de datos mediante un sistema de gestión de base de datos tipo series temporales, InfluxDB. Para poder comprobar que los datos se están almacenando correctamente, se dispone de la herramienta de visualización de datos en bases de datos Grafana. Estos datos se volcarán a su vez en la nube de ThingSpeak mediante el protocolo HTTP, donde se podrá visualizar la evolución de los mismos desde cualquier dispositivo móvil con conexión a Internet.

Con todos los componentes hardware configurados correctamente para hacer uso de todas las herramientas mencionadas, se dispone la memoria del proyecto a detallar en profundidad toda la programación de los mismos teniendo en cuenta cada proceso del que se ha hablado anteriormente.

# Capítulo 4

# Descripción de la Arquitectura desarrollada

## *4.1 Introducción*

Con todos los elementos hardware, entornos de desarrollo, librerías descargadas e instaladas en los dispositivos y configurados correctamente, se puede proceder a la programación de los procesos que llevarán al sistema a su completa ejecución.

En este capítulo, se expondrán todos los procesos de forma detallada, tanto en su programación como en su lógica de ejecución, desde la primera lectura del ESP32 hasta el último dato monitorizado en la nube de ThingSpeak, aportando pruebas de desarrollo y conclusiones.

En primera instancia, se procederá a evaluar el diagrama general del sistema, reiterando los procesos ya mencionados de forma organizada y concisa. Posteriormente a este apartado se procederá a explicar cada uno de los procesos con sus respectivos diagramas de flujo y explicación de la metodología implementada.

En última instancia se procederá a tomar conclusiones mencionando las características principales del sistema en su totalidad.

## 4.2 Diagrama general del sistema

A continuación, se expondrá un diagrama de flujo general del sistema con sus respectivas conexiones y flujo de datos.

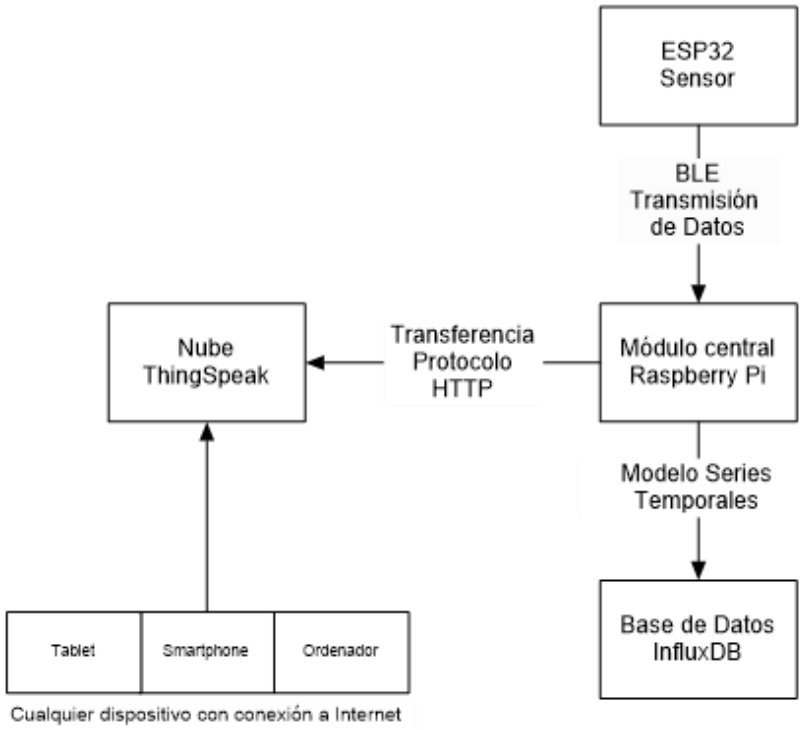


**Ilustración 4-1: Diagrama General del Sistema.**

Como se puede observar desde el diagrama con todas sus características, a lo largo de este capítulo se tomarán los apartados siguientes:

- ESP32 como sensor.
- Transmisión de datos entre ESP32 y Raspberry mediante BLE.
- Transmisión de información hacia la base de datos mediante el sistema de gestión de datos de series temporales InfluxDB.
- Configuración de Grafana como visualización de la base de datos InfluxDB.
- Transferencia de información mediante protocolo HTTP hacia la nube de ThingSpeak.
- Conexión desde un dispositivo a la nube de ThingSpeak y monitorización de datos.

## *4.3 Programación de sensor en ESP32*

Como ya se mencionó anteriormente, este trabajo se centra en la recepción y procesamiento de los datos recibidos desde el sensor puesto en campo, pero es necesario tener a disposición una serie de datos que se puedan procesar. Es por ello que para simular el comportamiento de un sensor se tomó en cuenta el microcontrolador ESP32, ya que tiene muchas características que lo hacen una buena elección para aplicaciones de IoT.

La implementación del ESP32 como sensor se basa en una generación de valores aleatorios comprendidos entre 0 y 255 (1 byte) que el mismo controlador debe crear con determinada frecuencia. La tasa de frecuencia escogida como generación de los números aleatorios es de 1 segundo, simulando un sensor capaz de leer un parámetro físico de su entorno con una tasa de refresco de 1 segundo.

Como se habló anteriormente, la programación del ESP32 se ha hecho en el entorno Arduino IDE, por lo que el código para escribir dicha implementación está basado en el lenguaje de programación C++. Para la generación del número aleatorio se creó una función que se encarga de dicha tarea la cual devuelve el número en cuestión para poder almacenarlo en una variable global. La función en cuestión consta de las siguientes líneas:

```
uint16_t GenerateRandomValue(void){
  uint16_t value;
  value = rand() % (UPPER_LIMIT_RANDOM_NUMBER-LOWER_LIMIT_RANDOM_NUMBER+1)
          + LOWER_LIMIT_RANDOM_NUMBER;
  return value;
}
```

Simplemente se llama a la función 'rand' la cual devuelve un número comprendido entre 0 hasta una constante llamada 'RAND_MAX', pero que gracias a las operaciones que le preceden se logra establecer un intervalo deseado. Los valores de las definiciones creadas 'UPPER_LIMIT_RANDOM_NUMBER' y 'LOWER_LIMIT_RANDOM_NUMBER' es de 255 y 0, respectivamente. Por lo que cuando se quiera obtener un número aleatorio mediante esta función, lo único que habría que hacer es llamarla asignando el valor de retorno a una variable cualquiera como se ha hecho en este código de la siguiente manera:

```
value = GenerateRandomValue();
```

Posteriormente a esto, se espera un segundo (1000 milisegundos) mediante la función 'delay' para volver a ejecutar en un loop infinito esta función.

```
delay(1000);
```

En resumidas cuentas, el diagrama de flujo de esta parte del código quedaría de la siguiente forma:

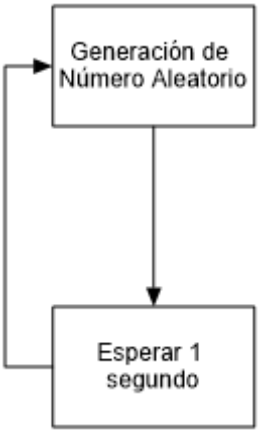


**Ilustración 4-2: Diagrama Simulación de Sensor en ESP32.**

Y así de esta manera se ha conseguido simular mediante el microcontrolador ESP32 un sensor capaz de leer una magnitud física cambiante a lo largo del tiempo con una tasa de frecuencia de 1 segundo.

## *4.4 Conexión BLE entre ESP32 y Raspberry Pi*

La transmisión del número generado aleatoriamente desde el microcontrolador ESP32 hasta el módulo central Raspberry Pi se ha hecho mediante el protocolo de conexión inalámbrico Bluetooth, en concreto mediante su variante en baja energía, BLE. Dicho protocolo debe ser implementado en ambos dispositivos para que su conexión sea compatible y estable.

### *4.4.1 Perfiles Cliente y Servidor*

Dentro de este protocolo se pueden divisar dos perfiles claramente diferenciados el uno del otro. Estos perfiles son los siguientes:

- Servidor: Este es el dispositivo que expone los datos que contiene para que otro dispositivo se pueda conectar a él y pueda consumir dichos datos. El servidor es el dispositivo que acepta comandos entrantes y envía respuestas, notificaciones e indicaciones.

- Cliente: Este dispositivo es aquel que interactúa con el servidor conectándose a él y consumiendo los datos que el servidor expone para su utilización. A su vez, también puede controlar, si el servidor lo permite, el comportamiento del mismo.

Este es el dispositivo que envía comando y solicitudes y acepta notificaciones e indicaciones.

En nuestro caso en particular, en esta conexión BLE, la Raspberry actúa de cliente conectándose al ESP32 el cual actúa como servidor, ya que quien controla los datos y los expone para su utilización es el microcontrolador ESP32, mientras que la tarjeta Raspberry recibe estos datos y los consume.

## *4.4.2 Servicios y Características del protocolo*

### *4.4.2.1 Servicios*

Dentro del protocolo se encuentra una agrupación de atributos en concreto denominada Servicio, la cual sirve como una etiqueta que agrupa atributos relacionados entre sí, es decir, que satisfacen una función específica en el servidor. Estos atributos generalmente son Características, aunque también se pueden encontrar otros tipos de atributos como declaraciones de servicio, declaraciones de características, entre otros.

### *4.4.2.2 Características*

Una característica siempre se encuentra dentro de un servicio y representa una pieza de información/datos que un servidor elige exponer a un cliente. Las características poseen a su vez diferentes atributos que ayudan a su definición, como, por ejemplo, las propiedades de dicha característica. Las propiedades de las características representan los permisos con los que el cliente puede acceder a su valor. Entre ellos se encuentra el permiso de lectura, escritura, notificación e indicación.

## *4.4.3 Implementación del Servidor ESP32*

Como se ha comentado anteriormente, el microcontrolador actuará de servidor el cual expondrá al cliente su número generado aleatoriamente para que, de esta manera, el cliente haga uso de ella conectándose al ESP32.

Como partes del proceso de implementación de la comunicación del servidor ESP32, primeramente, se crean tres variables globales tomando en cuenta las librerías provistas por Espressif (desarrolladora del microcontrolador ESP32), las cuales son las siguientes:

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
```

Mediante estas tres librerías añadidas al código, se pueden definir las tres variables encargadas de gestionar la parte del servidor, el servicio y la característica correspondiente. Las variables creadas globalmente son las siguientes:

```
BLEServer* pServer = NULL;
BLEService *pService = NULL;
BLECharacteristic* pCharacteristic = NULL;
```

En un principio se definen como NULL, aunque después en el código de configuración del servidor se asignarán como es debido. Primeramente, definimos el comportamiento como servidor y definimos los servicios y características que tendrá el perfil:

```
BLEDevice::init("ESP32-Sensor");
pServer = BLEDevice::createServer();
pService = pServer->createService(SERVICE_UUID);
pCharacteristic = pService->createCharacteristic(
                                        CHARACTERISTIC_UUID,
                                        BLECharacteristic::PROPERTY_READ
                                      );
pService->start();
```

Mediante la llamada a la función 'init' se inicia el protocolo BLE pasándole el nombre visible de nuestro dispositivo. Posteriormente se define el dispositivo como un servidor mediante 'createServer'. Los servicios y características, análogamente se definen con las funciones 'createService' y 'createCharacteristic'. Las definiciones generales de 'SERVICE_UUID' y 'CHARACTERISTIC_UUID' simplemente son las direcciones UUID de cada uno de ellos. Nótese como al definir la característica, se le ha otorgado una propiedad de lectura. Esto quiere decir que el cliente solo podrá leer en esta característica, mas no escribir.

Posterior a esto, se implementa la configuración para que el dispositivo servidor ESP32 pueda ser leído por un cliente que escanee los dispositivos cercanos y que, al conectarse a dicho servidor, este exponga su servicio completo.

```
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
pAdvertising->addServiceUUID(SERVICE_UUID);
pAdvertising->setScanResponse(true);
pAdvertising->setMinPreferred(0x06);
pAdvertising->setMinPreferred(0x12);
BLEDevice::startAdvertising();
```

Una vez hecho toda la configuración del perfil, tan solo queda almacenar periódicamente el valor aleatorio generado dentro de la característica. Para ello, tan solo hay que llamar a una función que asigne este valor a la característica correspondiente:

```
pCharacteristic->setValue((uint8_t*)&value, 1);
```

De esta manera, toda la programación del sensor ESP32 quedaría resuelta a la espera de que un cliente se conecte y lea de su característica, el valor generado por el microcontrolador.

Si se completa el diagrama anterior añadiendo la comunicación BLE escrita en el código, toda la programación del ESP32 que actuará como simulación de un sensor quedaría de la siguiente forma:

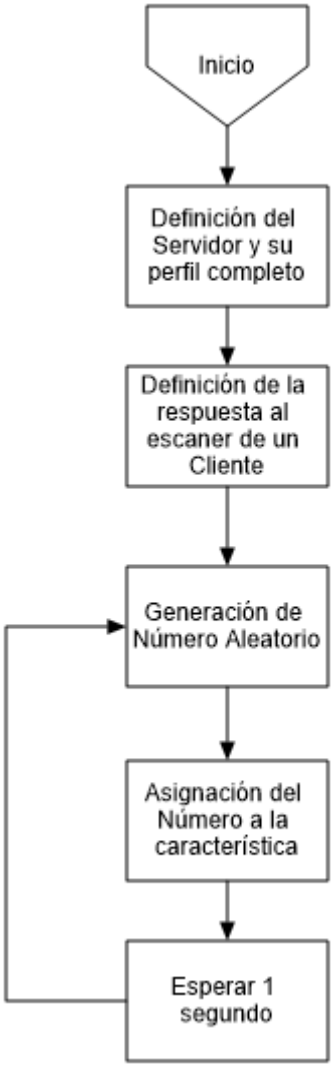


**Ilustración 4-3: Diagrama Comportamiento de Sensor en ESP32.**

### *4.4.4 Implementación del Cliente Raspberry Pi*

La tarjeta Raspberry Pi actúa como cliente en este perfil de conexión. Es aquel quien recibe los datos del servidor ESP32 y los consume, procesándolos como se verá más adelante. Es por ello que, en este caso, la tarjeta Raspberry es quien debe conectarse al ESP32, y no al contrario.

Primeramente, cabe destacar que la programación de la implementación de este perfil en la Raspberry se hace a través del lenguaje Python, mediante las librerías BluePy, las cuales a bajo nivel utilizan el código Bluez, otorgado por Bluetooth para su desarrollo. Es por ello que primeramente se debe importar la librería correspondiente al archivo python:

```
from bluepy import btle
```

Con esta librería importada ya se puede comenzar a escribir código que permita la conexión con el servidor y la lectura de su servicio y característica. Primeramente, se define la conexión con el servidor conocida su dirección Bluetooth de la siguiente manera:

```
dev = btle.Peripheral("80:7d:3a:b7:5d:26")
```

Una vez conectado al servidor, se dispone de todos sus perfiles y características para su utilización. Por ello ahora se procede a acceder a sus perfiles conocida la dirección UUID del perfil, de la siguiente manera:

```
Value_Service = dev.getServiceByUUID(Value_Service_UUID)
```

Con el servicio ubicado dentro de la variable 'Value_Service', se procede a tomar una característica de dicho servicio mediante el método:

```
Value_Characteristic = Value_Service.getCharacteristics(Value_Characteristic_UUID)[0]
```

Cabe destacar que anteriormente se han utilizado dos variables: Value_Service_UUID y Value_Characteristic_UUID, los cuales son simplemente variables de tipo UUID definidas anteriormente:

```
Value_Service_UUID = btle.UUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b")
Value_Characteristic_UUID  = btle.UUID("beb5483e-36e1-4688-b7f5-ea07361b26a8")
```

Una vez guardada la característica dentro de 'Value_Characteristic', tan solo se procede a leer la variable solo si tiene propiedad de lectura. Posteriormente a dicha característica se le aplica el método 'read' mediante el cual se permite la transferencia del

valor desde el servidor hasta el cliente. A este valor se le aplican métodos para transformar dicho valor en un valor entero leíble, y se lo imprime en pantalla para su supervisión:

```
if(Value_Characteristic.supportsRead()):
    value = binascii.b2a_hex(Value_Characteristic.read())
    value = int(value, 16)
    value_global = value
    print ("Value read: ")
    print(value)
```

Una vez terminada la transferencia de los datos desde el servidor hasta el cliente, este se desconecta del servidor ya que no volverá a leer los datos hasta dentro de un minuto, mediante el método 'disconnect':

```
dev.disconnect()
```

Se ha tomado la medida de desconectarse y volverse a conectar cuando se tenga que leer el dato, ya que supone un consumo de energía menor debido a que no tiene que estar manteniendo la conexión durante todo ese tiempo de espera. Para que el código espere 60 segundos se utiliza el método 'time.sleep':

```
time.sleep(60)
```

Todo este proceso se repite en un bucle infinito en el que el cliente toma el valor del servidor cada 60 segundos. De esta manera, el diagrama de la conexión cliente de la tarjeta Raspberry Pi sería la siguiente:

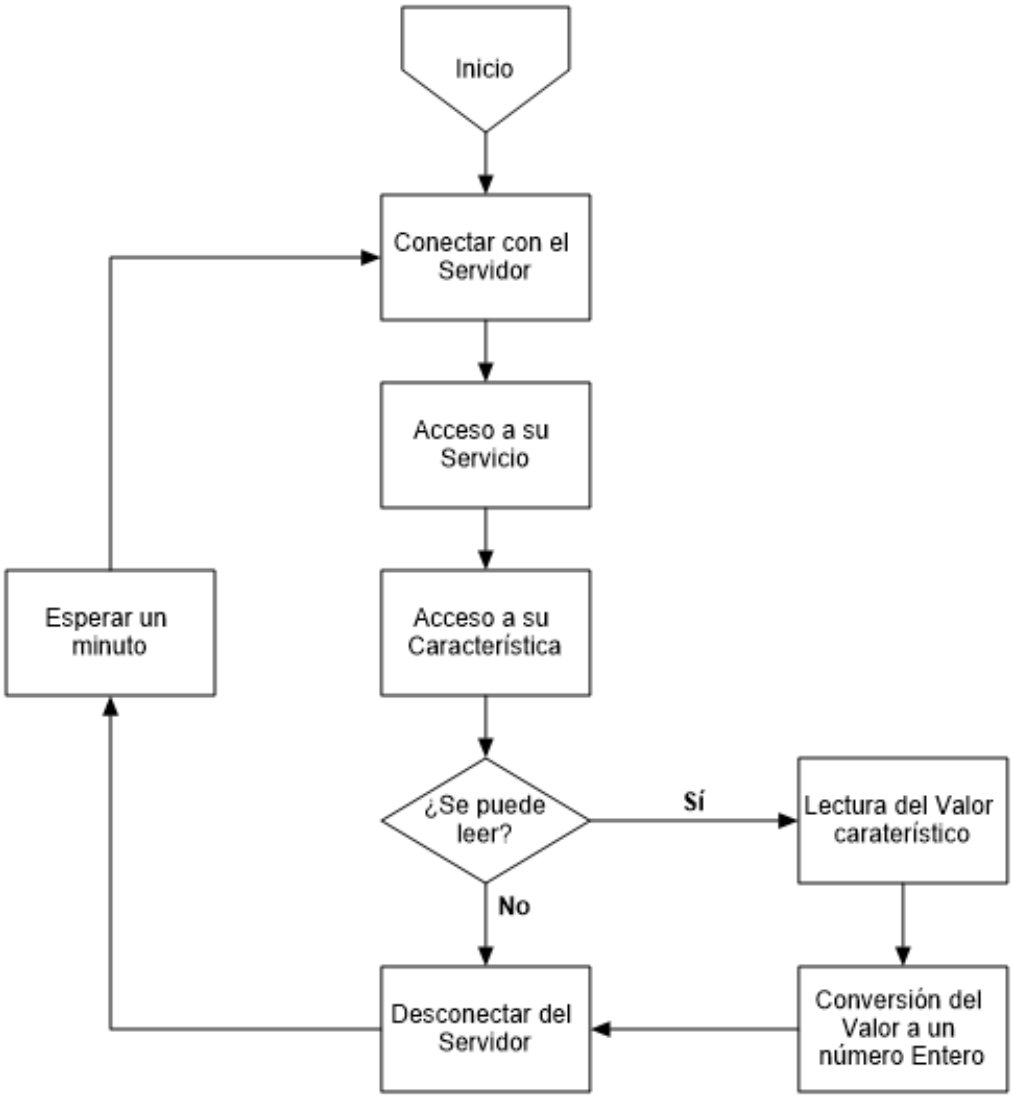


**Ilustración 4-4: Diagrama Comportamiento Cliente de la tarjeta Raspberry Pi.**

## *4.5 Implementación del SGBD InfluxDB*

Durante el Capítulo 3 se preparó el terreno en la tarjeta Raspberry Pi para la implementación de la gestión de bases de datos (SGBD) de influx. Cabe recalcar que al igual que con la implementación del protocolo cliente de BLE en la Raspberry, también se utilizará Python como lenguaje de programación para la implementación de dicho sistema.

Primeramente, se añaden las librerías python al sistema operativo para poder usarlas posteriormente en el código. Para ello, se ejecuta el siguiente comando en el terminal de Linux:

- "sudo apt-get install python-influxdb"

Ahora ya están habilitadas las librerías de InfluxPy, las cuales se usarán para crear la base de datos y a su vez ir añadiendo cada uno de sus elementos. Para poder utilizar los métodos procedentes de InfluxPy, los añadimos insertando la siguiente línea en el encabezado del código:

```
from influxdb import InfluxDBClient
```

Como es lógico pensar, antes de comenzar a escribir la implementación de los datos en la base de datos correspondiente, dicha base de datos debe ser creada al empezar el proceso. Por ello, lo primero que debe hacer el código es crear un cliente (la tarjeta Raspberry) que construirá una base de datos y seguidamente se colocará dentro de ella para poder escribir toda la información que debe reflejarse en ella. Todo este proceso viene definido de la siguiente forma en el código:

```
client = InfluxDBClient(host='localhost', port=8086)
client.create_database('SensorDB')
client.get_list_database()
client.switch_database('SensorDB')
```

Ahora en este punto, el código se encuentra colocado correctamente dentro de la base de datos, listo para comenzar a escribir la información en ella. Como paso previo a la escritura de los datos sobre la base, se debe moldear los datos que se van a escribir con campos bien definidos en formato json. Para la aplicación que nos incumbe, se le dio un campo el cual tiene el nombre de la medida: 'SensorValue'; un tag denominado 'user' en el cual viene definido 'Jonathan'; y un campo de valor denominado 'value' en donde se almacenará el valor recogido desde la conexión BLE. Dicha configuración se almacena de la siguiente forma en el código:

```
value_body_influx = [
    {
        "measurement": "SensorValue",
        "tags": {
            "user": "Jonathan"
        },
        "fields": {
            "value": value_global
        }
    }
]
```

Definido el cliente con su puerto dedicado, la base de datos y el modelo de dato que se insertará en ella, tan solo queda escribir el dato dentro de la base de la siguiente forma:

```
client.write_points(value_body_influx)
```

Ahora, como en el caso anterior, el código espera 60 segundos hasta que un nuevo valor es leído para guardarlo dentro de la base de datos, mediante el método 'time.sleep'.

Como se puede observar, el flujo de datos desde la lectura del valor transmitido mediante BLE hasta la escritura del mismo en la base de datos, es fluido y se puede graficar mediante el siguiente diagrama, que engloba toda la escritura de la base de datos.

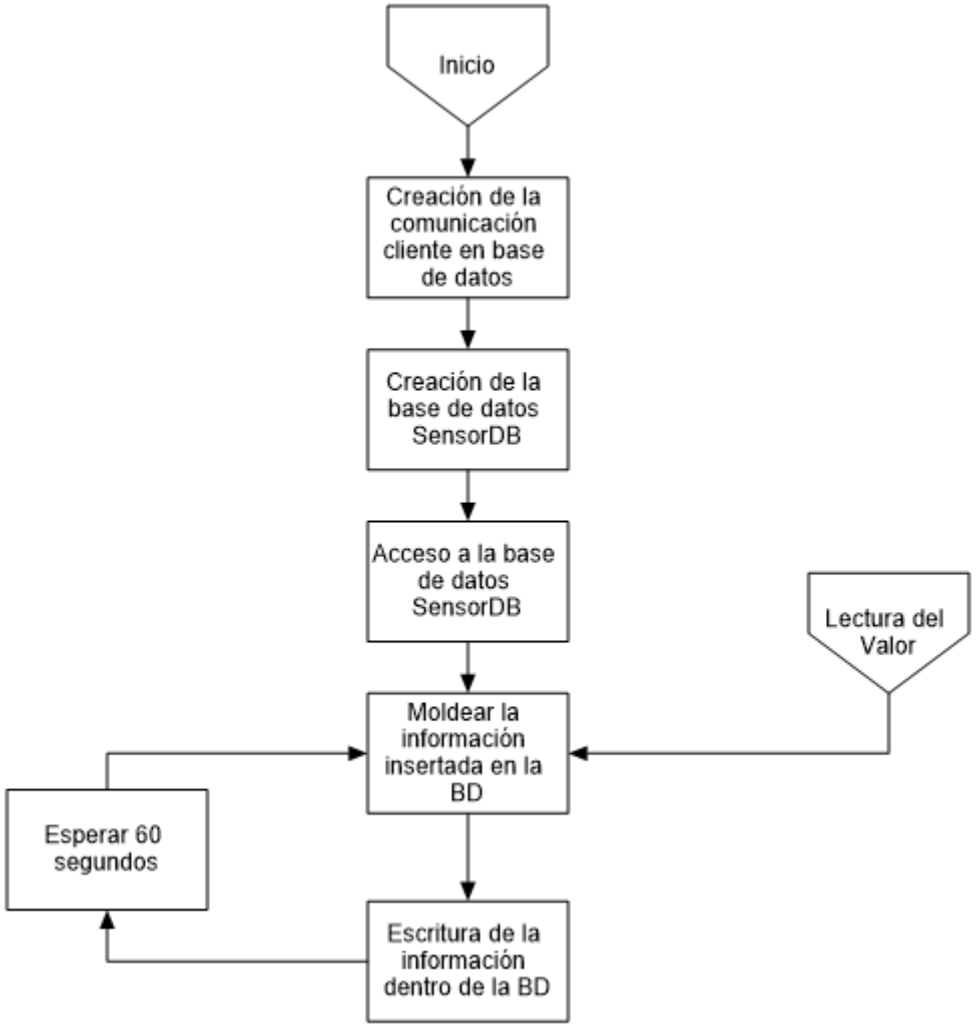


**Ilustración 4-5: Diagrama Escritura de la información mediante InfluxDB.**

## *4.5.1 Visualización de Datos mediante Grafana*

A manera de observación de los datos insertados en la base de datos de InfluxDB, se instaló en el sistema operativo, el servicio de Grafana mediante el cual se puede graficar la evolución de los datos insertados a lo largo del tiempo. Para ello, lo primero es acceder a su servicio mediante el navegador web escribiendo 'http://<ipaddress>:3000', donde la ipaddress es la dirección física de nuestra placa Raspberry.

Posteriormente, se procede a configurar una nueva fuente de información dentro de Data Source > Add new Data Source. Dentro de ese apartado se verá reflejado InfluxDB como fuente de información, por lo que se rellenan los parámetros de configuración de la base de datos InfluxDB y se creará una nueva fuente de información de manera exitosa.

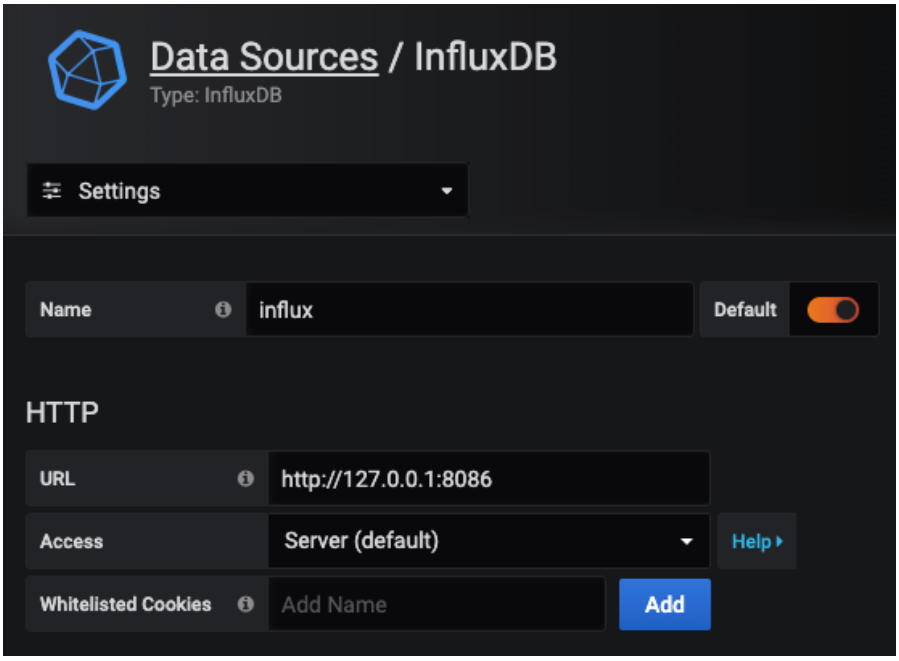


**Ilustración 4-6: Configuración de InfluxDB en Grafana.**

Con esto preparado, nos dirigimos a los dashboard y allí se pueden crear las gráficas temporales recogiendo la fuente de información que se acaba de crear.

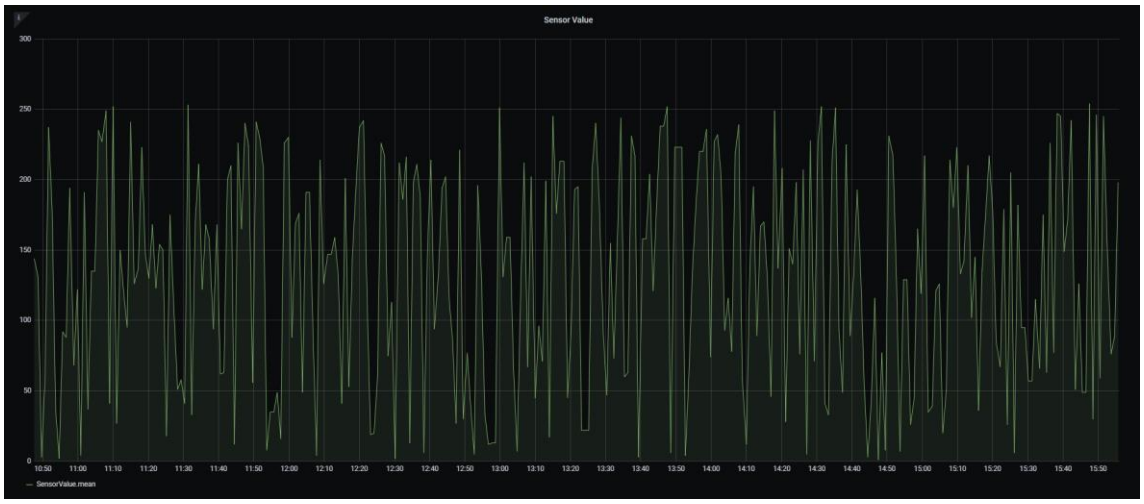


**Ilustración 4-7: Evolución del valor numérico leído por BLE.**

## 4.6 Transferencia de los datos hacia ThingSpeak

Cuando se accede a la cuenta de ThingSpeak, como se comentó anteriormente, dentro del apartado API Keys, se puede encontrar la llave para escribir dentro del gráfico que se haya creado anteriormente. Esta llave mezclada con la URL base dada por ThingSpeak para la escritura de valores en el apartado API Requests, son los elementos clave que permitirán que se almacenen los datos dentro de la nube.

Primero que todo, las librerías que se utilizarán para la implementación de esta parte del código ya vienen instaladas dentro del sistema operativo de raspberry. Tan solo, hay que llamarlas para comenzar a operar con sus métodos. Cabe destacar que para su implementación también se utilizará el lenguaje de programación Python debido a su alta aplicabilidad para estas tareas. De esta forma, primeramente, se incluyen las librerías en el código escrito:

```
import sys
import urllib2
```

Para proceder a la escritura de los valores dentro de la nube de ThingSpeak, se hace uso de los datos anteriormente citados de la nube (llave y URL base) y se los declara como unas constantes de la siguiente manera:

```
myAPI = 'UMCX8CMU86V2XF7Q'
baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI
```

De esta forma, dentro de la variable 'baseURL' queda reflejada la dirección HTTP, casi en su totalidad, a la que se hará referencia en el momento de abrir el socket de comunicación.

Con el terreno debidamente preparado, ahora se procede a la transferencia HTTP de los valores hacia la nube ThingSpeak. Para ello, abrimos el socket rellenando la última parte de la dirección con el valor correspondiente que se requiere enviar, y se envía el valor. Seguidamente se cierra el socket de comunicación. Todo ello se hace mediante las siguientes líneas de código:

```
conn = urllib2.urlopen(baseURL + '&field1=%s' % (value_global))
print conn.read()
conn.close()
```

Posteriormente, la ejecución del código espera 60 segundos hasta volver a recibir un valor desde la comunicación BLE para poder enviarlo a la nube repitiendo la última parte del código.

Dicho flujo de información quedaría reflejado en un diagrama de flujo de la siguiente manera:

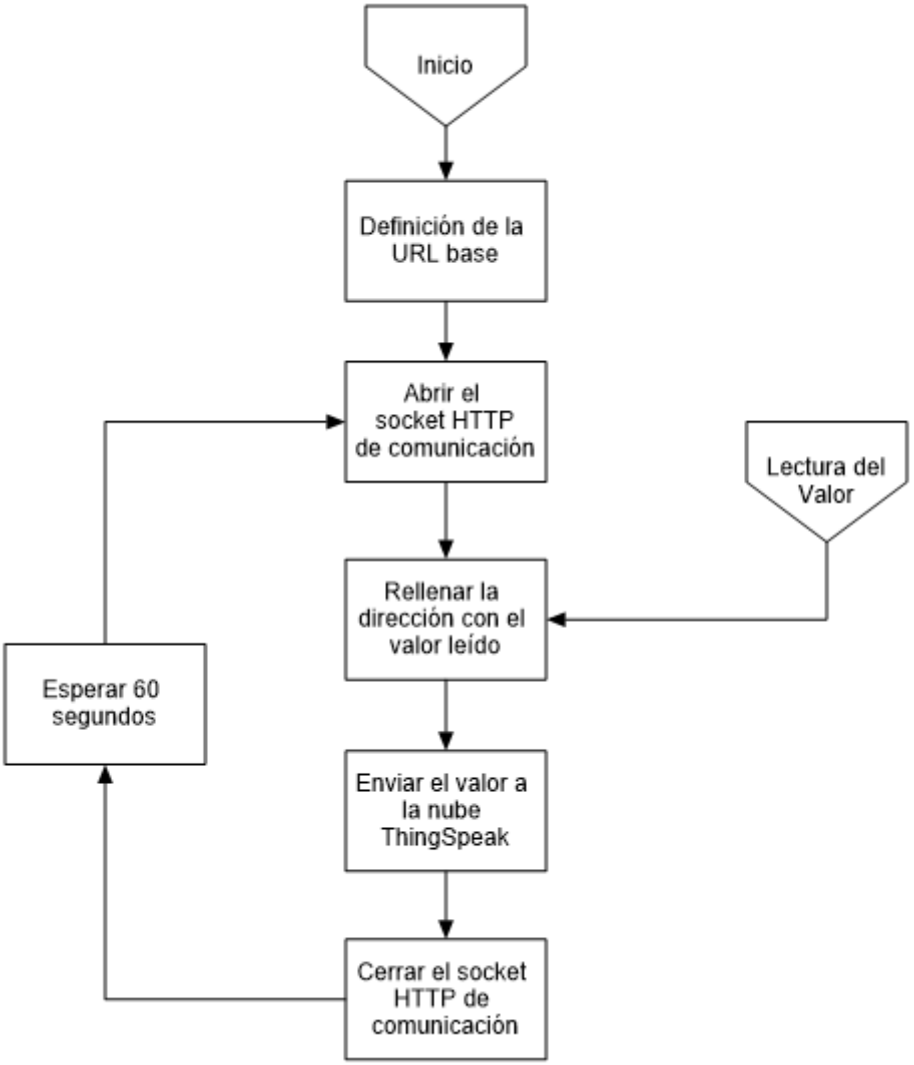


**Ilustración 4-8: Diagrama de flujo transmisión del valor a la nube ThingSpeak.**

## *4.7 Conexión a la nube ThingSpeak*

La ventaja más amplia de la aplicación de este método es que desde cualquier dispositivo que disponga de conexión a Internet, se podrá monitorizar la evolución de los valores en un gráfico temporal, y no tan solo eso, sino que dichos datos pueden ser utilizados para estudios posteriores mediante las herramientas de Matlab.

Una vez los valores se estén almacenando correctamente dentro de la nube de ThingSpeak, lo que se verá en cuanto se actualice el gráfico es algo parecido a lo siguiente:

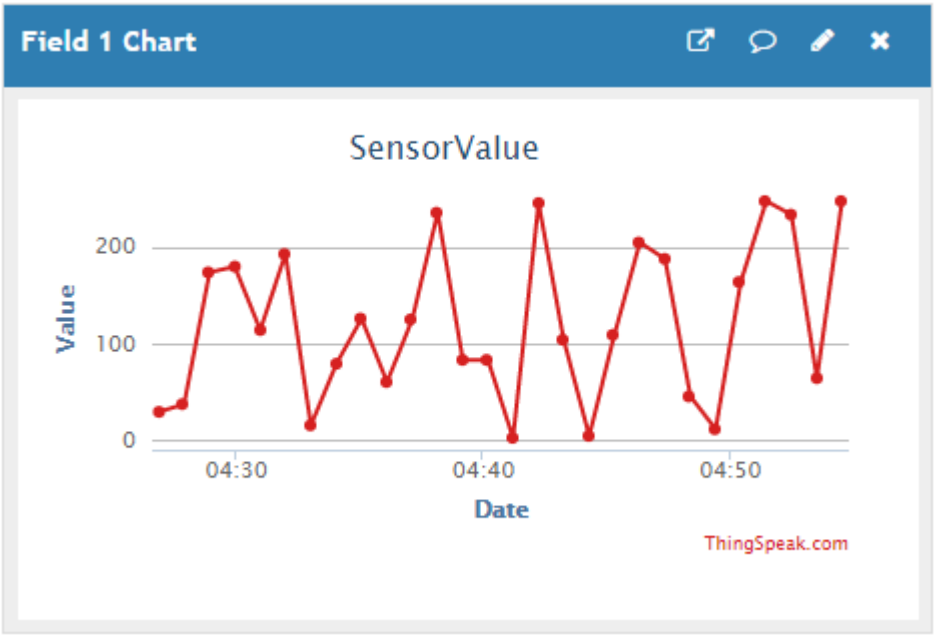


**Ilustración 4-9: Evolución temporal de un valor visto desde ThingSpeak.**

En cuanto a la conexión con la nube, bastará con que el usuario acceda a su cuenta de ThingSpeak desde cualquier dispositivo con conexión a Internet, ya sea una tablet, un smartphone o un ordenador portátil.

## *4.8 Conclusiones*

Como se ha podido observar a lo largo de todo el capítulo, se ha detallado en profundidad la elaboración de todo el sistema proceso a proceso y configuración a configuración. Se partió desde un valor generado desde el microcontrolador ESP32 que viajó en un flujo de información desde el microcontrolador hasta la nube de ThingSpeak, pasando por diferentes elementos hardware y por diferentes protocolos de comunicación.

El sistema en su totalidad es un sistema que cumple con su función principal, y tiene una serie de características propias:

- Soporta la conexión BLE entre dispositivo sensor y módulo central.
- Tiene una tasa de monitorización de valor de 1 minuto.
- El sensor tiene una tasa de refresco de magnitud física de 1 segundo.

# Capítulo 5

# Conclusiones y Trabajo Futuro

## *5.1 Conclusiones*

Este proyecto partía de la idea de realizar un sistema de monitorización de parámetros enfocados al sector de la agricultura. Para ello, se dispuso de diferentes herramientas tanto hardware como software, con las que se pudiera llevar a cabo el trabajo completo.

Primeramente, se realizó un estudio de mercado donde se plantearon varias opciones y varios caminos a seguir para la realización del proyecto. De entre todos ellos, se escogieron los métodos, mecanismos y componentes que más se adaptarán a las premisas iniciales para el proyecto. Dichas premisas estaban claras, se quería diseñar un sistema de bajo coste que cumpliera con su función.

Una vez elegidos los componentes y mecanismos, se procedió a la instalación de todos los drivers, librerías, plugins y entornos que el sistema requería para su total implementación. Dentro de todos los elementos del entorno, también se siguió con la premisa clara, y se escogió aquello que fuera más asequible para una implementación no muy compleja, y, por lo tanto, no muy costosa.

Ahora sí, con el entorno preparado y listo, se procedió al desarrollo y programación de todos los procesos que intervenían dentro del sistema para cumplir con los objetivos propuestos desde el principio.

Como resultado final, se consiguió el objetivo principal, el cual se puede reflejar en las gráficas de evolución de los datos de ThingSpeak. La monitorización de los parámetros es precisa y no ha supuesto un coste muy elevado en su desarrollo.

## *5.2 Trabajo Futuro*

Como trabajo futuro, principalmente, se podría seguir la línea del smartphone, ya que se podría implementar una interfaz web que permitiría configurar parámetros del sistema como la tasa de monitorización que por defecto está establecida en 1 minuto. Desde dicha interfaz, incluso se podrían crear perfiles en donde uno de los campos de dicho perfil sea la llave URL de la nube de ThingSpeak, para de esta forma, poder elegir desde la interfaz hacia qué nube se envía el valor, y qué valor se envía del sistema.

En efecto, se ha mencionado varios valores en el párrafo anterior ya que otra futura línea de proyecto sería que el modulo central se pudiera conectar de forma automática a varios sensores de campo y los distribuyera individualmente y así desde la interfaz web poder elegir donde alojar cada magnitud de cada sensor que el módulo central está leyendo.

De esta forma, usuario de ThingSpeak e interfaz quedarían ligados mediante una correcta utilización de las llaves que proporciona ThingSpeak para cada gráfico de la nube.

Además, otra tarea futura importante sería la de probar el sistema en condiciones reales de funcionamiento, la cual ha sido complicado realizarla por motivos de la pandemia COVID-19. Para ello, sería necesario adquirir un mástil, una caja estanca, una batería y un panel solar, así como otros elementos electrónicos necesarios. De esta manera, se podría instalar el equipo en un cultivo y realizar pruebas de funcionamiento en condiciones reales.

# Bibliografía


1. Bennett, S. (1993). A History of Control Engineering 1930–1955. London: Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers. ISBN 978-0-86341-280-6.

2. Jihong Yan (2015). Machinery Prognostics and Prognosis Oriented Maintenance Management. Wiley & Sons Singapore Pte. Ltd. p. 107. ISBN 9781118638729.

3. Ganesh Kumar (September 2010). Modern General Knowledge. Upkar Prakashan. p. 194. ISBN 978-81-7482-180-5.

4. "Protocolo de comunicación inalámbrica", publicado el 1 de diciembre de 2004

5. Knud Lasse Lueth (2014). Why the Internet of Things is called Internet of Things: Definition, history, disambiguation. https://iot-analytics.com

6. ¿En qué consiste el procesamiento de datos? https://www.talend.com/

7. Computer Hope (2017). BeagleBoard. https://www.computerhope.com/

8. Características de la BeagleBoard. https://beagleboard.org/

9. Jonathan Angel (2011). $89 dev board includes Cortex-A8 CPU, Ethernet, JTAG. https://archive.is/

10. (2017). BeagleBoard-xM. http://beagleboard.org/

11. Andrew K. Dennis (2016). Raspberry Pi Computer Architecture Essentials.

12. Rubén Velasco (2018). Análisis: Raspberry Pi 3 Modelo B+. https://hardzone.es/

13. Barragán, Hernando. «The Untold History of Arduino». http://arduinohistory.github.io/ (en inglés). Consultado el 2 de marzo de 2018.

14. Redes Inalámbricas vs Redes Cableadas: Ventajas y Desventajas. https://dispositivoinalambrico.com/

15. Ricardo Aguilar (2019). El WiFi cumple 20 años: de arma para luchar con los nazis a ser usado por 13.000 millones de dispositivos. https://www.genbeta.com/

16. Wifi - Conexión inalámbrica. https://www.ciset.es/

17. (2011). La historia del nacimiento de Bluetooth. https://www.fayerwayer.com/

18. Radio Versions. https://www.bluetooth.com/

19. Damián Pérez Valdés (2007). ¿Qué son las bases de datos? http://www.maestrosdelweb.com/

20. Gestión de Bases de Datos. https://gestionbasesdatos.readthedocs.io/

21. (2020). Base de datos orientada a objetos: el secreto mejor guardado de los modelos de bases de datos. https://www.ionos.es/

22. Equipo TD (2019). ¿Qué Son Las Bases De Datos De Series Temporales? https://tienda.digital/

23. Angel Gutierrez (2019). Qué es la nube de Internet, sus peligros y sus ventajas. https://www.aboutespanol.com/

24. ThingSpeak. https://es.mathworks.com/help/thingspeak/

25. Saket (2019). What is Xively and How it Helps in Building IoT Applications? https://www.it4nextgen.com/

26. ESP32. https://www.espressif.com/en/products/socs/esp32

27. Development Boards. https://www.espressif.com/en/products/devkits

28. InfluxDB. https://dbdb.io/db/influxdb

29. The analytics platform for all your metrics. https://grafana.com/grafana/