Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación de la tecnología de Internet de las Cosas en el ámbito educativo

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Antoni Giménez Rodríguez

Tutores: Sara Blanc Clavero y José Vicente Benlloch Dualde

2020-2021

Resumen

Durante este trabajo de fin de máster se abordará la introducción de la tecnología Internet de las Cosas en el contexto de la enseñanza secundaria. Se propondrán actividades para la visualización de datos proporcionados por dispositivos Arduino mediante el uso de distintas Clouds IoT. Dichas actividades incorporan distintas tecnologías informáticas, uso de bases de datos de Firebase, programación de placas Arduino, uso de máquinas virtuales, creación de un blog con WordPress y el uso del protocolo estándar de mensajería MQTT, así como el diseño de una pequeña aplicación tanto para escritorio Windows como para dispositivos Android. Todas las actividades propuestas están diseñadas para adecuarse a los marcos educativos que marca la Generalitat Valenciana y el marco europeo para las competencias digitales DigComp.

Palabras clave: Arduino, IoT, Cloud, DigComp, Educación Secundaria, Firebase, Android, WordPress, VirtualBox, MQTT

Resum

Durant aquest treball de fi de màster s'abordarà la introducció de la tecnologia Internet de les Coses en el context de l'ensenyament secundari. Es proposaran activitats per a la visualització de dades proporcionades per dispositius Arduino mitjançant l'ús de diferents Clouds IoT. Aquestes activitats incorporen diferents tecnologies informàtiques, ús de bases de dades de Firebase, programació de plaques Arduino, ús de màquines virtuals, creació d'un blog amb WordPress i l'ús del protocol estàndard de missatgeria MQTT, així com el disseny d'una xicoteta aplicació tant per a escriptori Windows com per a dispositius Android. Totes les activitats proposades estan dissenyades per a adequar-se als marcs educatius que marca la Generalitat Valenciana i el marc europeu per a les competències digitals DigComp.

Paraules clau: Arduino, IoT, Cloud, DigComp, Educació Secundaria, Firebase, Android, WordPress, VirtualBox, MQTT

Abstract

During this master's thesis, the introduction of Internet of Things technology in the context of secondary education will be addressed. Activities will be proposed for the visualization of data provided by Arduino devices using different IoT Clouds. These activities incorporate different computer technologies, use of Firebase databases, Arduino board programming, use of virtual machines, creation of a blog with WordPress and the use of the standard MQTT messaging protocol, as well as the design of a small application both for Windows desktop and Android devices. All the proposed activities are designed to comply with the educational frameworks set by the Generalitat Valenciana and the DigComp digital skills European framework for.

Keywords: Arduino, IoT, Cloud, DigComp, Secondary Education, Firebase, Android, WordPress, VirtualBox, MQTT

Índice de contenidos

Índice de ilustraciones

Índice de Tablas

1. Introducción

Hoy en día, no existe un currículo perfectamente pautado sobre qué contenidos y con qué métodos se tiene que enseñar en los institutos de secundaria. Es por ello que, dependiendo del docente asignado para el curso, el alumnado puede recibir clases de informática sobre el manejo de programas ofimáticos y, por ejemplo, realizar una actividad de cómo "calcar" un documento en papel en MS Word. Sin embargo, el año siguiente, podría haber otro docente en la misma asignatura que enseñe al alumnado a programar.

En este proyecto se pretende la creación de unas guías y actividades didácticas que puedan orientar al profesorado en su práctica docente, al tiempo que sean motivadoras y divertidas para el alumnado. Dichas actividades están basadas en tecnología de Internet de las Cosas (IoT), uso de plataformas en la red, comúnmente conocidas como *clouds,* y en el manejo de las famosas placas microcontroladoras y programables Arduino.

Motivación

En esta sección se justifica el interés del autor en la realización de este proyecto, así como las motivaciones personales y técnicas. Los motivos por los cuales he decidido desarrollar el siguiente proyecto son:

- Aprender una tecnología desconocida para el autor como era el desarrollo en Arduino, ya que es una tecnología relativamente sencilla y se pueden realizar infinidad de proyectos en el ámbito del IoT, actualmente en auge.
- Pasión por la docencia. Desde mi cuarto año de grado, siempre he tenido en mente el trabajar como docente. Después de estar trabajando unos 4 años en el sector privado, me he decidido a dar el paso y apostar por la docencia en Secundaria, teniendo como una meta futura el intentar dar el paso a ser docente universitario en la que ha sido mi facultad durante los seis años de formación universitaria.
- Una vez terminado el máster de Ingeniería Informática, tengo intención de realizar el Máster Universitario en Profesor/a de Educación Secundaria especialidad: Informática. Por ello, quiero realizar una serie de guías/actividades dirigidas a niños de secundaria para que yo mismo pueda aplicarlas en mis futuras clases. Dado que en dicho ámbito existe un abanico muy grande de posibilidades, se han querido plasmar una serie de actividades con el fin de hacer más amenas las clases y trabajar con cosas cuyos resultados sean visibles en poco tiempo.

Objetivos

El objetivo final del trabajo es la elaboración de guías didácticas que orienten al profesorado de informática y tecnología en su práctica. A su vez, queremos acercar el mundo tecnológico a los niños y jóvenes y así, animarlos a que encaminen sus estudios al mundo de la informática. Para poder alcanzar el objetivo principal debemos lograr los siguientes objetivos secundarios:

- Crear un proyecto en Firebase (así como el manejo de una base de datos en tiempo real y notificaciones Cloud Messaging de Firebase).
- Crear una pequeña aplicación en Android Studio para poder recibir las notificaciones de Firebase.
- Crear una aplicación de escritorio para descargar la base de datos alojada en Firebase.
- Recepcionar y tratar valores de sensores (temperatura, humedad…) desde un Arduino.
- Crear y configurar una máquina virtual basada en CentOS7.
- Instalar un servidor WordPress y manejar distintos complementos.
- Utilizar un bróker MQTT, publicar y descargar contenido.
- Recepcionar contenido del bróker MQTT y escribir en pantalla el resultado mediando Arduino.
- Crear un *dashboard* utilizando distintas *Clouds* dedicadas al IoT.

Metodología

En la realización de este proyecto, una de las características principales es que se trata de un trabajo de fin de máster concertado, por tanto, los cambios que nos va a pedir el cliente van a ser mínimos. Dado que en el proyecto se tienen que investigar diversas tecnologías, los cambios normalmente vienen dado en sentido contrario, es decir, cuando nos encontramos con una dificultad técnica, de la cual no encontramos una fácil salida, se consensúa con los tutores el camino a seguir.

En cuanto a las reuniones, hemos realizado una media de una reunión por semana con los tutores para resolver dudas, realizar el seguimiento y mostrar el progreso del trabajo. Dichas reuniones se han realizado a través de la plataforma Microsoft Teams que nos proporciona la universidad. Hemos creado un equipo nuevo dentro de la plataforma y hemos gestionado desde ahí la gestión documental, así como el calendario de las reuniones y las fechas más importantes

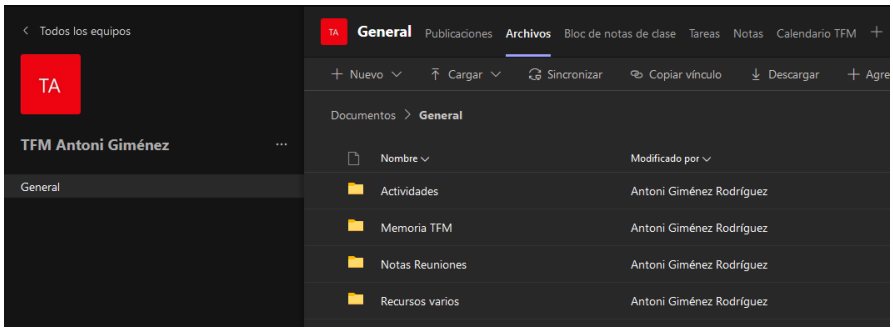


Ilustración 1. Captura documentos equipo Microsoft Teams

Al tratarse de un trabajo individual, hemos tenido que trabajar de forma clásica. Aunque haya tareas que no dependan unas de otras, hemos tenido que abordarlas de manera secuencial. Sin embargo, sí existen ciertas tareas, como la elaboración de los manuales y la redacción de esta memoria, que se ha ido desarrollando a medida que se avanzaba en el proyecto.

En la elaboración de los manuales, se han ido redactando tal y como iba avanzando la implementación final, ya que, si esperamos a realizarlos al finalizar la implementación, obtendríamos un nivel de detalle muy inferior a la forma en que lo hemos realizado.

El proyecto consta de ocho fases en total, pero lo hemos sintetizado en estas tres grandes agrupaciones, tal y como se puede apreciar en la ilustración 2:

Ilustración 2. Principal agrupación de las fases del proyecto

El **inicio** está compuesto de las siguientes tres fases:

Fase Inicial**:** Cuya única tarea es agrupar las reuniones iniciales con los tutores del proyecto.

*Brainstorming***:** En esta fase, en una primera tarea pensamos en una serie de actividades y posibles proyectos que podríamos llevar a cabo. Para finalizar con esta fase, hicimos una reunión con los tutores para exponer las ideas y seleccionar las más relevantes.

Definición final de las actividades**:** Una vez seleccionadas las ideas más relevantes, nos reunimos para tratar de encajar unas con otras, y así poder crear actividades con una complejidad mayor. Finalmente, redactamos los objetivos finales de cada actividad.

El **desarrollo** agrupa la implementación de la aplicación de escritorio para descargar la base de datos en Firebase y la realización de las tres actividades seleccionadas.

La aplicación de escritorio ha sido subdividida en tres tareas:

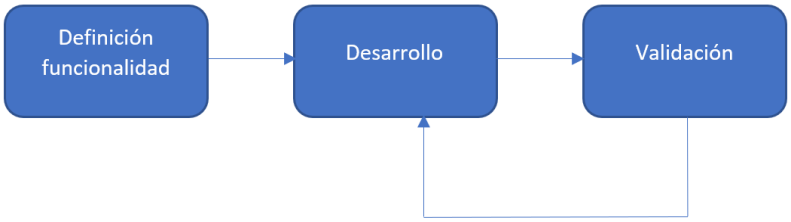


Ilustración 3. Tareas Aplicación escritorio

Como muestra la ilustración 3, una vez realizado el desarrollo, se ha validado con los tutores que se había cumplido con la definición inicial. Si alguna de estas validaciones no era satisfecha, iterábamos la fase de desarrollo y posterior validación, tantas veces como fuera necesario.

Por su parte, cada actividad ha sido subdividida en las siguientes tareas:

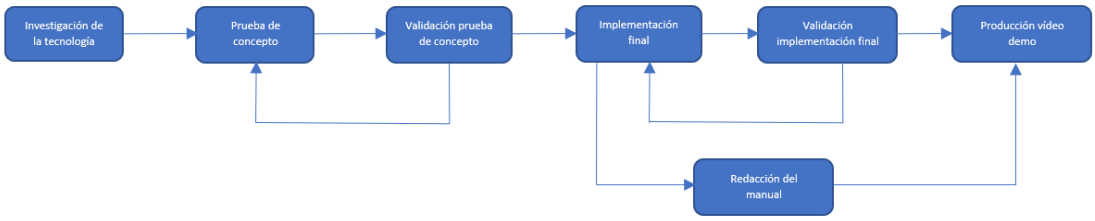


Ilustración 4. Tareas de las actividades.

Tal y como podemos observar en la ilustración 4, las actividades se han dividido en siete tareas, siendo todas ellas secuenciales, a excepción de la redacción del manual. En las tareas de validación, tanto para la prueba de concepto como para la implementación final, se ha iterado, tantas veces como ha sido necesario para poder validar dichas tareas.

La última agrupación **final**, corresponde con la fase de redacción de la memoria y la presentación final del trabajo.

Hemos hecho uso de un repositorio de código de GitHub para registrar los distintos *issues*, o problemas, así como para mantener un histórico de nuestro código. El uso de dicho repositorio, también nos facilita la divulgación de nuestro código, tanto en los anexos como en el intercambio de conocimiento.

Dichos repositorios se pueden encontrar en la siguiente URL:

https://github.com/antonigimenezrodriguez?tab=repositories

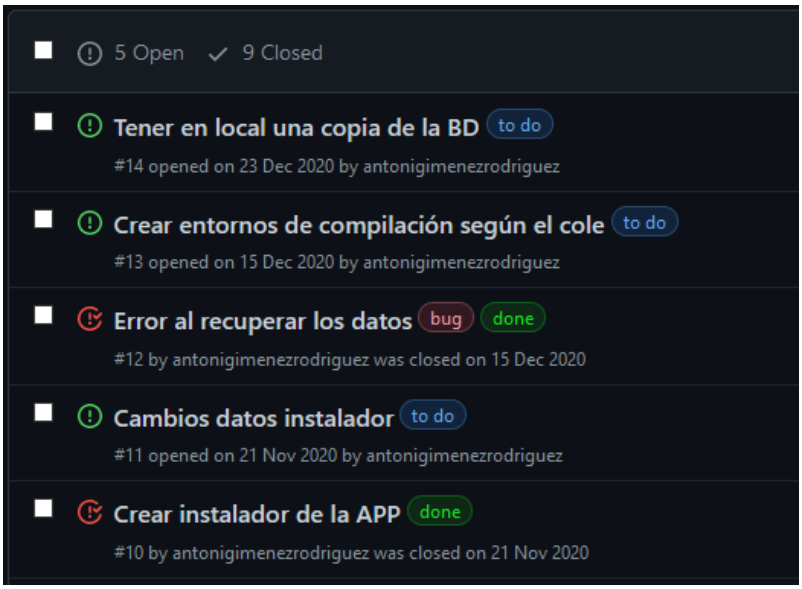


Ilustración 5. Panel Issues GitHub

La ilustración 5 corresponde a un ejemplo del panel *Issues* de GitHub.

Colaboraciones

Este trabajo, es fruto de una colaboración con el proyecto educativo a nivel europeo "eSGarden – School Gardens for Future Citizens"[1]. El proyecto está liderado por la tutora del proyecto (Sara Blanc) y también participa el tutor (José V. Benlloch). Dicho trabajo tiene como objetivo la incorporación de las TIC a los huertos escolares a través de sensores conectados a dispositivos Arduino mandando dichos valores a una base de datos de Firebase. (Grindei, Blanc and Benlloch-Dualde 2019)

En una primera instancia, y dado que el manejo de datos en Firebase puede resultar complicado, se ha diseñado una aplicación de escritorio Windows, que se encarga de descargar en Excel una muestra de dichos valores, filtrados por el jardín seleccionado y por el campo dentro de ese jardín.

Gracias a esta herramienta, se pueden realizar los análisis, que el usuario considere oportunos, de una forma mucho más sencilla y amena que si se recogieran los valores directamente de Firebase.

En una segunda instancia, se ha diseñado una aplicación para Android que muestra notificaciones de Firebase Cloud Messaging. También se ha diseñado una librería para Arduino que es capaz de analizar los valores recogidos por los sensores.

En caso de superar ciertos umbrales, envía una notificación a todos los dispositivos que tienen instalada la aplicación de Android, a través de los servicios de Firebase Cloud Messaging, indicando qué sensor está recogiendo datos fuera de los intervalos correctos.

[1] Web del Proyecto: https://esgarden.blogs.upv.es/
Proyecto europeo Erasmus+: 2018-ES01-KA201-050599

2. Estado del arte

Actualmente, el uso de tecnologías de Internet de las Cosas (IoT) se encuentra muy extendido. Tecnologías inalámbricas ultra rápidas y de bajo consumo como son el 4G (o LTE) y el 5G han favorecido la proliferación en el uso de dichos dispositivos.

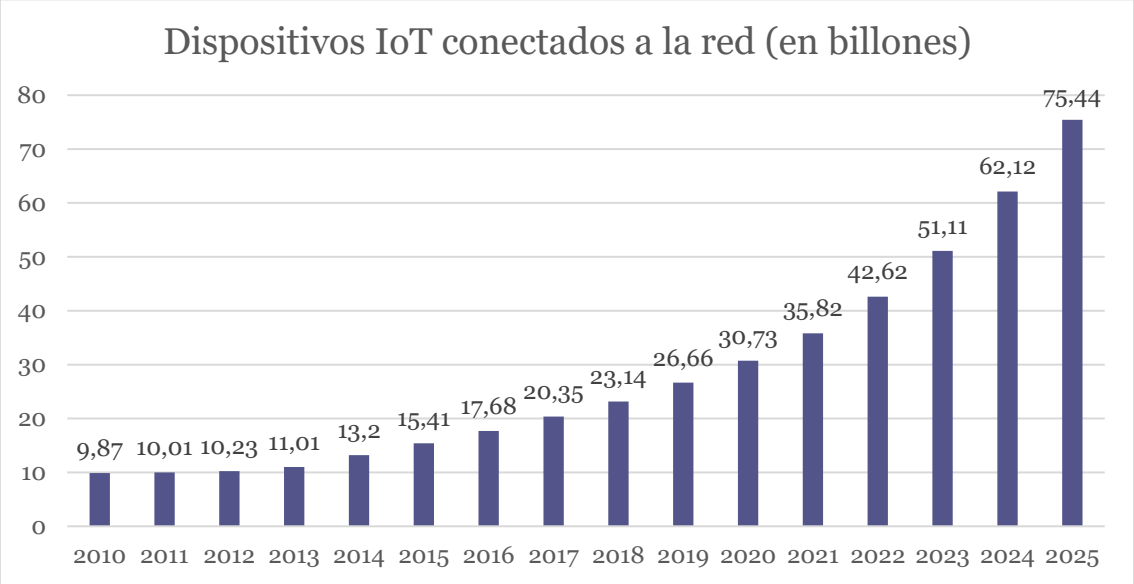


Ilustración 6. Gráfico número de dispositivos IoT conectados a la red

La ilustración 6 muestra el número de dispositivos IoT conectados a la red desde 2010, así como una previsión de su evolución hasta 2025 (dicho gráfico fue elaborado en 2020). A partir de 2015 hasta el año 2021, ha aumentado un 230% el número de dispositivos conectados. Se estima que desde 2021 hasta 2025, aumente un 210% más, lo que supone un 490% desde el año 2015(«Las grandes estadísticas del Internet de las Cosas (IoT) ~ IoT World Online» 2020).

Se observa que, si se traza una línea con los distintos valores por año, se obtiene una gráfica de tipo exponencial. El punto donde comienza el crecimiento exponencial es en los años 2013-2014 (lanzamiento del 4G), mientras que en el año del lanzamiento del 5G (2019) se observa otro salto relevante en cuanto a dispositivos conectados.

Los principales usos de la tecnología IoT hoy en día son:

- Dispositivos *wearables* como relojes inteligentes equipados con sensores para la detección de la presión arterial, pulso, temperatura, altitud, presión, localización…
- Refrigeradores inteligentes con cámaras web que permiten a los usuarios ver contenido remoto en tiempo real y recibir notificaciones de fechas de caducidad y cantidad de productos…

- Cepillos de dientes capaces de detectar la cantidad de sarro que se elimina durante el cepillado, la existencia de caries…
- Automóviles conectados a la red que permiten sincronizarse a la agenda del usuario, guardar citas, destinos cotidianos, avisar de accidentes… Actualmente se está trabajandoen la introducción de la tecnología IoT para la conducción autónoma a través de la sensorización de todo tipo y la posibilidad de conectarse con vehículos cercanos.
- Televisores conectados a internet que pueden navegar por la red, instalar software de terceros, así como reproducir contenido multimedia.
- Zapatos que se sincronizan con aplicaciones móviles u otros dispositivos para medir el rendimiento del usuario, tales como la cadencia de pasos, la distancia recorrida…

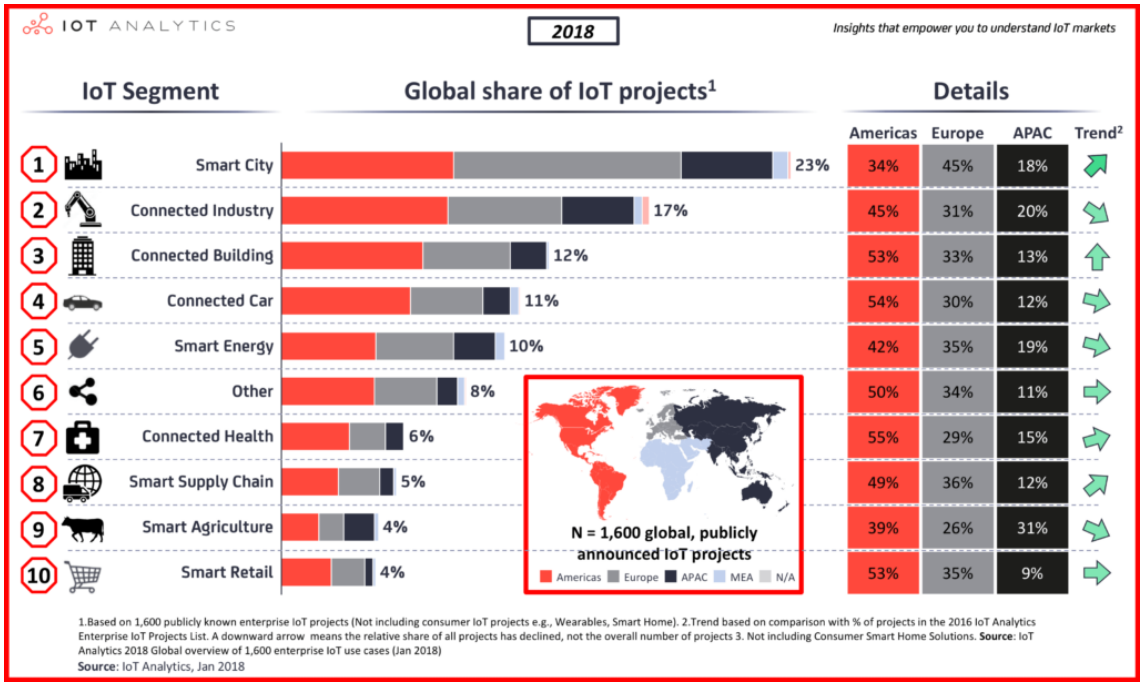


Ilustración 7. Industrias uso Internet de las Cosas. Fuente: https://www.infoplc.net/plus-plus/tecnologia/item/105149-proyectos-estrella-iot

La ilustración 7 muestra los diez ámbitos que más usan el Internet de las Cosas, siendo las *Smart Cities* las que tienen mayor cuota de mercado. Tal como se observa en la última columna, hay un buen número de ámbitos que se encuentra con tendencia al alza.

Hardware

Las principales placas de desarrollo para IoT son las siguientes:

- **SmartEverything**. Es una división de la empresa Arrow Electronics cuyo desarrollo se enfoca principalmente a grandes empresas.
  La familia SmartEverything consta de varias placas creadas en torno a tecnologías inalámbricas dirigidas a aplicaciones de IoT.

Las bibliotecas de controladores permiten un desarrollo rápido y fácil utilizando Atmel Studio o Arduino IDE. El formato de Arduino facilita la adición de *shields*[2] en la parte superior de la placa.

Arrow, no fabrica placas, sino que se basa en placas ya existentes (como puede ser Arduino) y otros componentes (como pueden ser módulos de red, de memoria…) para crear sus propias composiciones y, así, ofrecer soporte y soluciones de extremo a extremo para proyectos de IoT. A continuación. se mencionan ejemplos de estas composiciones:

SmartEverything Fox, es un dispositivo final que incorpora un módulo para conexiones Sigfox[3] y es compatible con el IDE de Arduino. Tiene un coste aproximado de 170€.

Las características principales de la placa Fox son:

- Chip SAMD21
- Procesador ARM Cortex M0+
- Ultra bajo consumo
- Chip de autenticación criptográfica
- Módulo SigFox
- Módulo GPS con antena incorporada
- Bluetooth

SmartEverything Lion, es un dispositivo Atmel D21 compatible con LoRa[4]. Su precio aproximado es de 150€

Las características principales de la placa Lion son:

- Procesador ARM Cortex M0
- Ultra bajo consumo
- Chip de autenticación criptográfica
- Módulo GPS con antena incorporada
- Bluetooth
- Modulo conexión LoRa

SmartEverything Tiger, es un microcontrolador compatible con Arduino que implementa la ejecución de distintos hilos de forma simultánea, usado para sistemas embebidos de consumo extremadamente bajo. Su precio aproximado es de 90€.

Las características principales de la placa Tiger son:

- Procesador ARM Cortex M0+
- Bluetooth
- IEEE Std. 802.15.4

[2] Un *shield* en Arduino es la adición de una placa sobre Arduino que se conecta mediante el acoplamiento de sus pines.
[3] SigFox es una red IoT de bajo consumo e independiente de las redes teléfonicas.
[4] LoRa es una tecnología de radio que utiliza un tipo de modulación de radiofrecuencia Semtech.

- **UrsaLeo**. Ofrece conexión directa e inmediata con una plataforma avanzada de servicios en la nube propia. Su uso principal se encuentra en la industria 4.0.
  Está enfocado a la creación de gemelos digitales, representación virtual de un producto o proceso de producción, en 3D. Permite conectar los sensores físicos directamente a sus gemelos digitales para realizar simulaciones o análisis.
  La plataforma UrsaLeo utiliza archivos CAD combinados con escaneos en 3D para recrear, de forma virtual, cualquier entorno. Resulta curioso leer del propio fabricante: "Para poner en marcha la plataforma se requieren unos pocos miles de dólares" («Model Creation – UrsaLeo»).

- **Raspberry**. Ofrece todas las características de un ordenador en forma de una placa reducida en cuanto a tamaño y coste. Se le pueden acoplar periféricos gracias a las conexiones incorporadas (normalmente USB, HDMI, PCI…) y precisa instalar un sistema operativo específico para Raspberry.
  En el momento de la redacción de este trabajo, las últimas versiones de dispositivos son:

  Raspberry Pi Pico, es una tarjeta con base el microcontrolador RP2040 caracterizada por su flexibilidad y bajo coste, aproximadamente 5€.

  Las características principales del Pi Pico son:
  - Procesador ARM Cortex-M0+ 133MHz
  - 264 KB de memoria RAM
  - 2 MB de memoria flash

  Raspberry Pi 4 Model B, es la seña de identidad de Raspberry, la última versión de un microordenador de bajo coste sin perder ninguna funcionalidad de un ordenador convencional cuyo precio se sitúa en torno a los 65€.

  Las principales características del Pi 4 son:
  - Procesador de cuatro núcleos Cortex A72 1.5 GHz
  - Tarjeta gráfica VideoCore VI 500 MHz
  - 2 GB de memoria RAM
  - Conectividad WiFi, Bluetooth y Ethernet Gigabit
  - Decodifica vídeo en 4K (hasta 2 pantallas simultáneas)
  - Conexiones HDMI y USB
  - Cabezal GPIO de 40 pines

  Raspberry Pi 400 Personal Computer Kit, es un miniordenador ensamblado en una estructura en forma de teclado cuyo precio aproximado es de 110€.

  Las características principales del Pi 400 son:
  - Procesador de cuatro núcleos Cortex A72 1.8 GHz
  - Tarjeta gráfica VideoCore VI 500 MHz
  - 4 GB de memoria RAM LPDDR4
  - Conectividad WiFi y Bluetooth
  - Conexiones HDMI, USB y Micro SD

- **Arduino**. Ofrece todos los elementos necesarios para conectar dispositivos, tanto de entrada como de salida, a un microcontrolador. Es de código abierto y está basado en hardware y software flexible y fácil de usar.

  Existen muchos modelos de placas diferentes para Arduino, es por ello por lo que vamos a analizar las placas más populares, así como las que están orientadas a entornos IoT.

  Arduino Uno: Es la placa estándar por excelencia y la más popular. Está basada en el microcontrolador ATmega328P con un precio de 20€ aproximadamente.

  Las características de la placa Uno son:

  - 32 KB de memoria flash
  - 2 KB de memoria SRAM
  - 26 MHz de velocidad de reloj
  - Conexión USB tipo B
  - 14 pines digitales
  - 6 pines analógicos

  Arduino Mega: Es la placa más potente con un microcontrolador de 8 bits (basada en el microcontrolador ATmega2560), así como la que más pines incluye, por lo que es totalmente apta para realizar trabajos más complejos. El precio de esta placa se sitúa sobre los 35€.

  Las características de la placa Mega son:

  - 256 KB de memoria flash
  - 8 KB memoria SRAM
  - 16 MHz de velocidad de reloj
  - Conexión USB tipo B
  - 54 pines digitales
  - 16 pines analógicos

  Arduino NANO 33 IoT: Se trata de la placa de Arduino diseñada en exclusiva para los estándares IoT. Está basada en el microcontrolador SAMD21 Cortex-M0+ de 32 bits, su coste es de 15€ aproximadamente.

  Las características de la placa Nano son:

  - 256 KB de memoria flash
  - 32 KB memoria SRAM
  - 48 MHz de velocidad de reloj
  - Conexión USB tipo micro-USB
  - Módulo Wifi WiFiNINA
  - 14 pines digitales
  - 8 pines analógicos

Arduino Leonardo: Se trata de la placa clásica de Arduino que puede actuar como ratón o teclado. Está basada en el microcontrolador ATmega32u4 con un coste de 20€ aproximadamente.

Las características de la placa Leonardo son:

- 32 KB de memoria flash
- 2.5 KB memoria SRAM
- 16 MHz de velocidad de reloj
- Conexión USB tipo micro-USB
- 20 pines digitales
- 12 pines analógicos

Arduino Micro: Se trata de una placa basada en el microcontrolador ATmega32U4 desarrollada junto a Adafruit[5]. Es una placa análoga a Arduino Leonardo, pero con un tamaño muy compacto y un coste similar de 20€.

Las características de la placa Micro son:

- 32 KB de memoria flash
- 2.5 KB memoria SRAM
- 16 MHz de velocidad de reloj
- Conexión USB tipo micro-USB
- 20 pines digitales
- 12 pines analógicos

Arduino Yun: Se trata de una placa pensada para el diseño de dispositivos conectados entre sí. Combina el poder de Linux con la facilidad de uso de Arduino usando un microcontrolador ATmega32u4 y la placa Atheros AR9331 para poder dar soporte a la distribución Linux Linino OS[6]

Las características de la placa Yun son:

- 32KB de memoria flash
- 2.5 KB memoria SRAM
- 16 MHz de velocidad de reloj
- Conexión USB tipo micro-USB
- Conexión USB tipo A
- 20 pines digitales
- 12 pines analógicos

[5] Adafruit es una empresa hardware de código abierto que diseña, fabrica y vende productos, componentes herramientas y accesorios electrónicos.
[6] Linino OS es una distribución de Linux basa en OpenWrt y permite la escritura de programas en lenguajes como Python y Node.js entre otros.

Las características del procesador Atheros AR9331 son:

- 16MB de memoria RAM DDR2
- 2.5 KB memoria SRAM
- 400 MHz de velocidad de reloj
- Capacidad para tarjetas Micro-SD
- Conexión Ethernet Fast
- Conexión WiFi

Software

Los principales IDEs de desarrollo para Arduino son:

IDE Arduino: Se trata del IDE oficial desarrollado por Arduino. Se usa para escribir, cargar y depurar código en placas compatibles con Arduino. También cuenta con la ayuda de núcleos de terceros para poder cargar y ejecutar código en ciertas placas de otros proveedores.
https://www.arduino.cc/en/software

Arduino IoT Cloud: Se trata de una plataforma *online* desarrollada por Arduino que te permite la creación de código de forma sencilla, así como la posibilidad de crear un panel de visualización de los datos. Lleva incluidas las librerías más populares.

Se trata de una plataforma *freemium*, es decir, es gratuita para la realización de *sketchs* sencillos, pero si se desea incluir más dispositivos, así como añadir más variables, hay que suscribirse a un plan de pago, a partir de 3€ hasta 24€ al mes[7].
https://create.arduino.cc/iot/things

Fritzing: Se trata de un programa libre que nos ayuda en la maquetación de circuitos electrónicos. Incluye un gran abanico de componentes, incluyendo la gran mayoría de las placas microcontroladoras del mercado.

Aunque la principal característica de Fritzing es la realización de los circuitos y esquemas eléctricos, ofrece la posibilidad de escribir código para cargarlo directamente en una placa Arduino.
https://fritzing.org/

Arduino Blocks: Se trata de una plataforma online gratuita que permite programar placas Arduinos a través de bloques. Esta forma de programar Arduino está especialmente indicada para niños en la etapa escolar de primaria.
http://www.arduinoblocks.com/

Ardublock: Se trata de un complemento que se puede adicionar al IDE de Arduino que permite realizar la programación de las placas Arduino con bloques de la misma forma que Arduino Blocks.
http://blog.ardublock.com/

[7] https://store.arduino.cc/digital/create#plans

Tinkercad: se trata de una plataforma online de modelado 3D y simulación de circuitos electrónicos perteneciente a Autodesk.
https://www.tinkercad.com/

Hemos analizado las siguientes opciones para programar aplicaciones en Android:

IDE Android Studio: Es el entorno de desarrollo nativo y oficial para el desarrollo de aplicaciones en Android. Como lenguaje de programación dispone de Java o Kotlin, según deseo del programador.

Android Studio es propiedad de Google e integra, de forma nativa y sin uso de librerías de terceros, las conexiones necesarias para hacer uso de Firebase.

Framework Xamarin con IDE Visual Studio: Es el *framework* propiedad de Microsoft para desarrollar aplicaciones nativas para dispositivos móviles (Android e iOS) en lenguaje .NET.

Xamarin permite crear aplicaciones idénticas para Android e iOS escribiendo solamente una vez la lógica de la aplicación.

Proyectos existentes

Resulta complicado el encontrar proyectos educativos con bases asentadas. Existen numerosos proyectos y ejemplos en la red, pero la gran mayoría se trata de pequeños proyectos creados y difundidos a nivel individual por distintos usuarios. Dichos proyectos nos sirven cono base para la realización de proyectos de más envergadura.

Existen, en la redacción de este trabajo, dos grandes plataformas educativas basadas en Arduino:

"**Proyecto A**"[8]: Se trata de una iniciativa impulsada por el ecosistema emprendedor la UPV startUPV en el sector de la educación, enfocada a centros educativos que proporcionan kits, guías y formación para el profesorado. Los cursos se pagan de forma individual y su coste oscila entre 30€ y 80€ para los cursos completos de iniciación (actividades individuales tales como la realización de un semáforo, un piano…)

Arduino Education[9]: Es un proyecto lanzado por la organización oficial de Arduino cuyo objetivo principal es la de ofrecer unos paquetes dirigidos y adaptados según la edad de los estudiantes, desde alumnos de primaria, hasta alumnos universitarios.

Ofrece un gran abanico de kits adaptados a los distintos niveles que se pueden usar como punto de partida. También ofrece la posibilidad de obtener un certificado en fundamentos de Arduino.

[8] https://www.proyectoa.es/
[9] https://www.arduino.cc/education

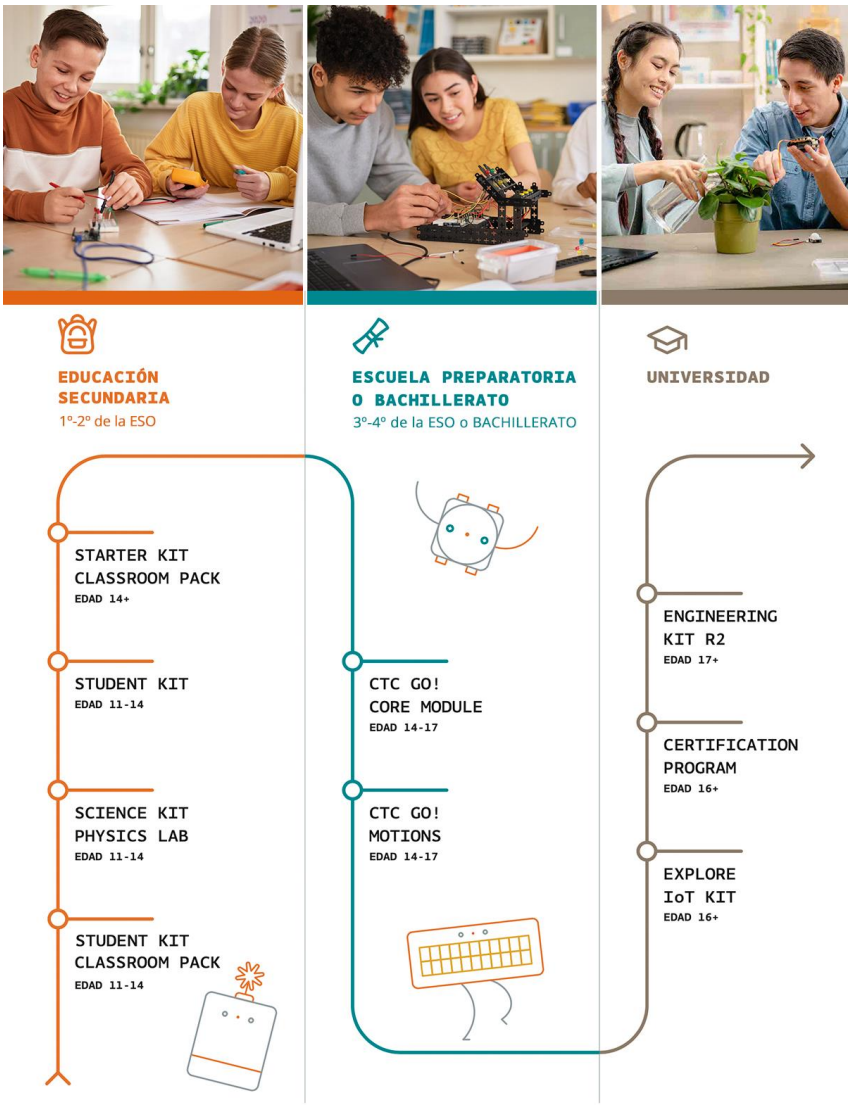


Ilustración 8. Mapa de ruta propuesto por Arduino Education

Como se observa en la ilustración 8, Arduino Education propone el siguiente mapa de ruta para los distintos niveles educativos.

Los precios de estos kits (en mayo de 2021) con el distribuidor tibot son:

Tabla 1. Precios kits Arduino Education

Material	Precio
Student Kit Classroom pack	300€
Science Kit Physics Lab	160€
Student Kit	70€
CTC GO! Core Module	1800€
CTC GO! Motions	1100€
Explore IoT Kit	120€
Engineering Kit R2	250€

Crítica al estado del arte

Después de analizar las distintas placas disponibles, se ha elegido la placa Arduino para la realización del proyecto. Arduino nació dentro del entorno educativo cuando un grupo de estudiantes de la escuela "Interaction Design Institute Ivrea (IDII) en Ivrea (Italia) presentaron un proyecto de final de máster que, finalmente, sentó las bases de lo que se conoce como Arduino («The Making of Arduino - IEEE Spectrum» 2011). A pesar de nacer en un entorno educativo, rápidamente se extendió su uso a diversos entornos, tales como IoT, arte, impresión 3D, drones, robótica, domótica…

Gracias a su bajo coste, se ha popularizado rápidamente y, hoy en día, existe una comunidad muy activa que ha favorecido el crecimiento del uso de las placas. Esto también ha permitido poner la tecnología/electrónica al alcance de todas las personas para la realización de proyectos de educación, soluciones industriales, ciudades inteligentes o proyectos de entretenimiento, entre otros.

Actualmente Arduino nos ofrece las siguientes ventajas:

- Bajo coste.
  Los componentes de Arduino tienen un coste bajo, sobre todo si se compara con otras soluciones similares.

- Fomenta el uso de la programación en el aula.
  Dado que Arduino nos ofrece un IDE de programación sencillo cuyos resultados son fácilmente visibles, es totalmente apto para su uso en el aula y como iniciación para los alumnos en la programación.

- Uso en escuelas e institutos.
  Dado que se trata de componentes hardware muy económicos y de fácil uso, son candidatos perfectos para su introducción en las escuelas e institutos pudiendo tener cada alumno su propio kit.

- Realización de cualquier tipo de proyecto.
  Dado que el abanico de Arduino es muy amplio, se presta fácilmente a la realización de cualquier tipo de proyecto, por ejemplo, un semáforo, recolección de datos, alumbrado inteligente, domótica, sistemas de alarmas…

- Introducción en el mundo tecnológico
  Dada la facilidad de adquisición y programación de las placas Arduino, es una de las mejores formas para introducirse en el mundo de la informática, tecnología o robótica teniendo pocos conocimientos en este campo.

- Sistemas completamente personalizados
  Dado que Arduino es un sistema completamente abierto, se pueden construir sistemas propios integrando los sensores o periféricos que se precisen según las necesidades. Con unos conocimientos avanzados, y dado que Arduino es de código abierto, se puede crear una nueva placa microcontroladora basada en Arduino.

Una vez analizadas las distintas opciones para el desarrollo de código Arduino, hemos decidido usar el IDE oficial de escritorio de Arduino ya que, por la naturaleza de nuestro proyecto, necesitamos incluir y adaptar algunas librerías de terceros.

Como el proyecto eSGarden está realizado con el IDE oficial de Arduino y la primera actividad requiere integración con el proyecto eSGarden, se decidió hacer uso de este en todas las actividades, para así seguir una uniformidad en eltrabajo.

Para la aplicación Android, después de contemplar las dos opciones mencionadas anteriormente, se ha decidido usar el Android Studio.

En una primera instancia, se planteó realizar el desarrollo con Xamarin para, de esta manera, poder crear una aplicación para los dos sistemas operativos móviles y usar el mismo IDE y lenguaje de programación que la miniaplicación de descargar de datos de Firebase. Sin embargo, finalmente, se decidió realizar la implementación por Android Studio por la simplicidad y las ayudas del IDE en la conexión a Firebase.

Esta decisión implica que los usuarios con dispositivos iOS no pueden disponer de la aplicación para las notificaciones.

Una vez analizadas las dos plataformas educativas existentes, hemos decidido realizar nuestros propios proyectos, adecuándolos según los estándares del DigComp (Ftacnik, Sveda and Kires 2020), y haciendo que nuestras actividades cubran, como mínimo, dos de las cinco competencias y cubriéndolas de forma intermedia o avanzada, según la actividad y la competencia.

Esta decisión se ha basado en el resumen ofrecido por cada plataforma, en el que se observa que no se llegan a cubrir los estándares del DigComp como deseamos, así como por su elevado coste. Tal y como observaremos en el apartado de presupuesto, las actividades propuestas en este trabajo tienen un coste sensiblemente inferior a las opciones de las plataformas comentadas anteriormente.

3. Análisis del problema

En este capítulo, introduciremos el marco de competencias digitales europeo DigComp.

Desde la primera Recomendación europea sobre competencias clave para el aprendizaje permanente(«RECOMMENDATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL» 2006), la competencia digital ha sido reconocida como una de las 8 competencias clave por la Unión Europea.

Iniciado como proyecto en 2010, el Centro Común de Investigación (JRC), el servicio de ciencia y conocimiento de la Comisión Europea, desarrolló el marco europeo para la competencia digital, dirigido tanto a ciudadanos como a estudiantes. Desde la primera versión, denominada DigComp 1.0 (Ferrari, Punie and Brečko 2013), se ha convertido en uno de los marcos de referencia para comprender lo que significa ser digitalmente competente a nivel europeo. DigComp 2.0 (Vuorikari et al. 2016) actualizó la terminología y el modelo conceptual, pero también introdujo ejemplos de implementación en diferentes niveles.

La versión actual, conocida como DigComp 2.1 (Carretero, Vuorikari and Punie 2017) se estructura en cinco dimensiones. La dimensión 1 divide la competencia digital en cinco áreas clave: i) Alfabetización en información y datos; ii) Comunicación y colaboración; iii) Creación de contenido digital; iv) Seguridad; y v) Resolución de problemas. La dimensión 2 define cada una de las 21 competencias a través de los resultados del aprendizaje. En la dimensión 3, estos resultados de aprendizaje se asignan en ocho niveles de competencia diferentes, que van desde principiantes hasta altamente especializados. Para completar el marco, la dimensión 4 detalla los conocimientos, las habilidades y las actitudes que se aplican a cada habilidad. Finalmente, la dimensión 5 muestra ejemplos del uso de la aplicabilidad de la competencia para varios propósitos.

La necesidad de preparar estudiantes competentes digitalmente crea nuevas demandas para los educadores, quienes no solo deben ser competentes digitalmente ellos mismos, sino que también necesitan un conjunto de competencias particulares para mejorar su enseñanza y aprendizaje. Para ello, se ha creado un marco de competencias específico para educadores, el Marco Europeo para la Competencia Digital de Educadores (Redecker and Punie 2017).

Una vez estudiado el marco educativo DigComp y comparándolo con el currículum actual de la Generalitat Valenciana para secundaria (Generalitat Valenciana. Consellería d'educació [sin fecha]) observamos que no se tienen en cuenta dichas competencias a nivel europeo. Quizás algunas pueden llegar a cubrirse de una forma muy básica.

Nuestro proyecto trata de la realización de tres prototipos usables y adaptables por los docentes de educación secundaria y bachillerato, en las especialidades de tecnología e informática, que han sido diseñadas para trabajar, como mínimo uno (existen actividades que trabajan más de un aspecto de forma simultánea) de las cinco áreas descritas en el DigComp. Aparte de trabajar las competencias, se han diseñado para alcanzar un nivel intermedio, como mínimo, si bien en algunas de las actividades se llega a trabajar un nivel avanzado.

Es por ello que, en primer lugar, se ha propuesto la realización de una aplicación para descargar los datos de la base de datos del proyecto eSGarden y así poder realizar la explotación de datos pertinente.

Seguidamente, se han propuesto tres prototipos cuyo objetivo principal es la interacción del alumnado con la tecnología. Todos estos tienen como base el uso de placas Arduino. Dicha decisión ha sido tomada después de observar que diversos institutos ya las emplean en sus clases. Al ser estas placas muy económicas, con una baja inversión de dinero, todo el alumnado puede disponer de placas y sensores sin tener la obligación de compartirlo con otro compañero.

La primera actividad corresponde con una ampliación de funcionalidad del proyecto eSGarden, donde se analizan los valores obtenidos por los sensores y, en caso de superar unos umbrales definidos, se manda una notificación vía Firebase Cloud Messaging a los dispositivos móviles que se han descargado nuestra APP.

En la segunda actividad se explica cómo construir una máquina virtual bajo sistema Linux, así como la creación de un blog en WordPress. Seguidamente, se construye el flujo, mediante el uso de componentes tales como complementos, bróker y librerías necesarias, para capturar ciertos eventos en el blog de WordPress, enviarlos a un bróker MQTT[10] y, desde un Arduino, recibir dichos mensajes y escribirlos en una pantalla LCD.

En la última actividad, se explica cómo crear un montaje de diversos sensores para la captura y envío de datos a distintas *Clouds IoT* para su visualización en forma de *dashboard*. Posteriormente, y tal y como comentamos en los futuros trabajos, se explotaría la plataforma ThingSpeak para realizar un análisis matemático en profundidad de los datos obtenidos, así como posibles predicciones según el histórico de datos.

[10] MQTT es un protocolo de mensajería estándar para Internet de las Cosas (IoT)

4. Solución propuesta

Este capítulo se inicia con la descripción del plan de trabajo mediante el correspondiente diagrama de Gantt. A continuación, se explicará el presupuesto disponible para la realización del proyecto, que está dividido en el coste del hardware necesario, así como el coste en horas de trabajo de las personas involucradas.

Seguidamente, se realizará una pequeña descripción de la arquitectura del sistema para, posteriormente, mostrar el diseño detallado.

A continuación, se explicarán las distintas tecnologías utilizadas, tanto hardware como software, para finalizar con el desarrollo de la solución propuesta.

Plan de trabajo

En este proyecto hemos trabajo de forma clásica en cascada, es decir, cuando terminamos una tarea, podemos avanzar a la siguiente. Esto sucede porque en el proyecto solo participa una persona, no existe la opción de paralelizar ninguna tarea. Como podemos observar, en las etapas finales parece que exista dicha paralelización, pero realmente se paran las tareas de redacción de memoria y se empieza con la preparación de la presentación mientras se recibe retroalimentación de la memoria para finalizar con los últimos retoques y modificaciones de esta.

Como podemos observar en la ilustración, el proyecto ha estado dividido en varias fases:

1. Fase Inicial
2. *Brainstorming*
3. Definición de las actividades
4. Descarga base de datos eSGarden
5. Actividad 1. Firebase Cloud Messaging
6. Actividad 2. Blog MQTT 2 Arduino
7. Actividad 3. *Dashboard*
8. Redacción de la memoria

Cada una de estas ocho fases, a su vez, están divididas en diversas tareas. Puede verse esto en el diagrama de Gantt del proyecto en la Figura 9.

Nombre de la tarea	Asignado	Fecha de inicio	Estimación
		2020/11/02	**413h**
Fase Inicial		**2020/11/02**	**15h**
Reuniónes iniciales con los tutores	A S J	2020/11/02	15h
Brain Storming		**2020/11/17**	**9h**
Análisis de ideas	A Antoni Gimenez	2020/11/17	6h
Puesta en común y selección de ideas	A S J	2020/11/23	3h
Definición final de las actividades		**2020/11/26**	**10h**
Redacción de los objetivos finales de las actividades	A S J	2020/11/26	10h
Descarga base de datos eSGarden		**2020/12/06**	**25h**
Definición funcionalidad	A S J	2020/12/06	3h
Desarrollo	A Antoni Gimenez	2020/12/09	20h
Validación desarrollo	A S J	2020/12/29	2h
Actividad 1. Firebase Cloud Messaging		**2020/12/31**	**87h**
Investigación de la tecnología	A Antoni Gimenez	2020/12/31	10h
Prueba de concepto	A Antoni Gimenez	2021/01/15	15h
Validación prueba de concepto	A S J	2021/02/04	3h
Implementación final	A Antoni Gimenez	2021/02/07	35h
Validación Implementación	A S J	2021/03/19	8h
Redacción del manual	A Antoni Gimenez	2021/02/15	15h
Producción vídeo demo	A Antoni Gimenez	2021/05/22	1h
Actividad 2. Blog MQTT 2 Arduino		**2021/03/22**	**56h**
Investigación de la tecnología	A Antoni Gimenez	2021/03/22	8h
Prueba de concepto	A Antoni Gimenez	2021/03/31	10h
Validación prueba de concepto	A S J	2021/04/11	3h
Implementación final	A Antoni Gimenez	2021/04/14	20h
Validación Implementación	A S J	2021/05/04	4h
Redacción del manual	A Antoni Gimenez	2021/04/19	10h
Producción vídeo demo	A Antoni Gimenez	2021/05/23	1h
Actividad 3. Dashboard		**2021/05/09**	**43h**
Investigación de la tecnología	A Antoni Gimenez	2021/05/09	5h
Prueba de concepto	A Antoni Gimenez	2021/05/14	10h
Validación prueba de concepto	A S J	2021/05/24	3h
Implementación final	A Antoni Gimenez	2021/05/27	15h
Validación Implementación	A S J	2021/06/11	4h
Redacción del manual	A Antoni Gimenez	2021/05/31	5h
Producción vídeo demo	A Antoni Gimenez	2021/06/10	1h
Redacción de la memoria		**2021/02/08**	**168h**
Preparación del material	A Antoni Gimenez	2021/02/08	30h
Redacción de la memoria	A Antoni Gimenez	2021/03/10	100h
Revisión de la memoria	A S J	2021/05/24	10h
Últimos retoques y modificación memoria	A Antoni Gimenez	2021/06/13	8h
Preparación presentación	A Antoni Gimenez	2021/06/07	10h
Revisión y pruebas de la presentación	A S J	2021/06/17	10h

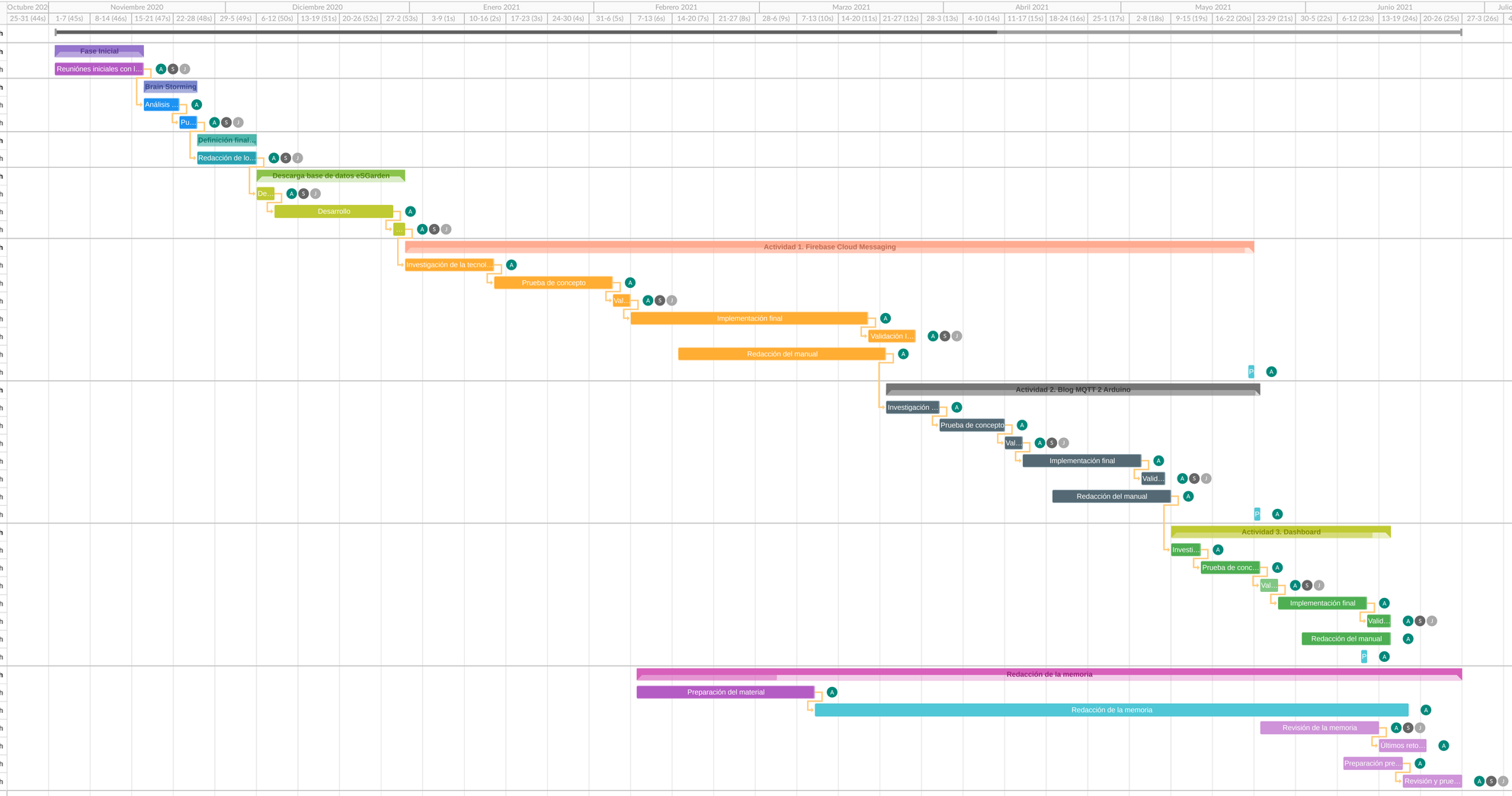


Ilustración 9. Diagrama de Gantt

La fase inicial tan solo contiene una tarea, agrupa las diversas reuniones que se han tenido con los tutores del proyecto.

La fase *brainstorming* está divida en dos fases, por una parte, la recopilación y análisis de todas las ideas y, por otra parte, una reunión con los tutores para la puesta en común y selección de las ideas finales.

La siguiente fase, definición final de las actividades, depende directamente de la fase anterior y consiste en la redacción detallada de los objetivos de las actividades seleccionadas.

Continuaremos con la fase de descarga de la base de datos que contiene las tareas de definición de la funcionalidad, desarrollo de la aplicación y la validación del desarrollo.

Las siguientes tres fases del proyecto, corresponden al desarrollo de las tres actividades con Arduino. Dichas fases contienen las mismas tareas: investigación de la tecnología, prueba de concepto, validación de la prueba de concepto, implementación final, validación de la implementación final y redacción del manual.

La última fase es la redacción de la memoria, cuyas tareas son: preparación del material, redacción de la memoria, validación de la memoria, últimos retoques y modificación de la memoria, preparación de la presentación y ensayo de la presentación.

Esta última fase del proyecto es la única fase que no tiene una dependencia directa de otra y, por tanto, la redacción de la memoria se ha ido avanzando en paralelo con la realización de las actividades. Se empezó a redactar la memoria desde las fases más tempranas de los desarrollos.

Presupuesto

Para analizar el presupuesto final de la realización del proyecto, se va a dividir en dos partes diferenciadas. Por una parte, tendremos los costes materiales y por otra parte los costes de horas de trabajo de las personas involucradas en el mismo.

Costes materiales

En este punto vamos a analizar los costes de licencias y hardware necesarios para la realización del proyecto. Dado que hemos utilizado plataformas que son gratuitas, o en su defecto, nos ofrecen gratuitamente la funcionalidad necesaria sin coste alguno, no tenemos que sumar ningún coste por licencias software y/o plataformas.

Por tanto, nos queda desgranar los costes en cuanto a hardware se refiere. Dichos costes quedan resumidos, para cada actividad, en la siguiente:

Tabla 2. Precios componentes Hardware

Actividad	Hardware	Precio unitario	Unidades	URL
Firebase Cloud Messaging	Arduino nano IoT 33	22,00€	1	Amazon
	Sensor DHT 11	1,24€	1	Ali Express
	Total actividad 1	**23,24€**		
Blog MQTT 2 Arduino	Arduino nano IoT 33	22,00€	1	Amazon
	LCD I2C 20x4	11,00€	1	Amazon
	Total actividad 2	**23,00€**		
Dashboard	Arduino nano IoT 33	22,00€	1	Amazon
	Protoboard + cables	14,00€	1	Amazon
	Sensor DHT 11	1,24€	1	Ali Express
	Sensor de agua	0,50€	1	Ali Express
	Sensor HC-SR04	2,70€	1	Amazon
	Sensor ADXL 345	9,99€	1	Amazon
	Total actividad 3	**50,43€**		

NOTA: los enlaces de la Tabla 1 están disponibles a fecha de la redacción de la memoria, mayo 2021

Podemos utilizar solo un Arduino para la realización de las tres actividades, aunque es totalmente recomendable la utilización de un Arduino por actividad. Por lo tanto, el coste material para la realización del proyecto asciende a un total de **106,67€.**

Costes horas de trabajo

Para realizar el cálculo de las horas de trabajo, nos hemos basado en las tablas salariales de un profesor de secundaria acorde a los sueldos públicos. En dicho cálculo se ha obtenido que un profesor de secundaria, sin trienios ni sexenios, pero contando las pagas extras, percibe una cantidad total mensual bruta de 2844,50€ (Generalitat Valenciana. Conselleria de Hacienda y Modelo Económico 2019). Sabiendo que la jornada semanal de un docente es de 37,5h, se obtiene un precio por hora de aproximadamente 19€ brutos[11].

Una vez hemos obtenido el precio bruto por hora, podemos concluir que el coste en horas de trabajo para esto proyecto asciende a **6840€**

[11] Se ha usado como referencia el salario de un profesor de secundaria, ya que está publicado en el Diari Oficial de la Generalitat Valenciana (DOGV) y es el público objetivo de dicho proyecto.

Arquitectura del sistema

Para describir la arquitectura del sistema, vamos a dividir dicho apartado en las cuatro implementaciones que hemos realizado.

Descarga base de datos eSGarden:

Ilustración 10. Figura descarga base de datos eSGarden

Tal y como muestra la Ilustración 10, la aplicación para descargar la base de datos se conecta a Firebase para descargar la información (huertos, campos y datos de los sensores) para componer y descargar un Excel con dicha información.

Para el correcto funcionamiento de dicha aplicación, tendremos que ejecutarla bajo entorno Windows, disponer de una conexión a internet, así como disponer del paquete Office de Microsoft para poder abrir el fichero generado.

Actividad 1. Firebase Cloud Messaging:

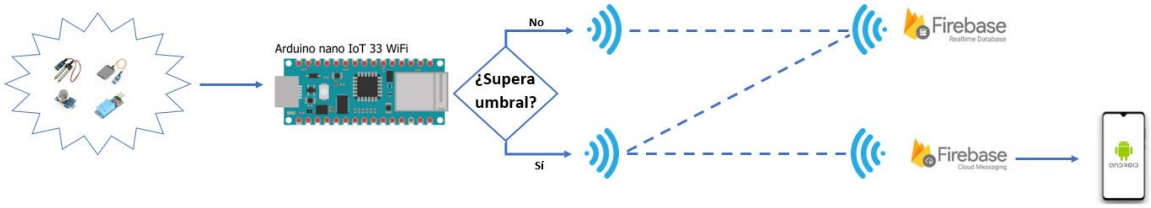


Ilustración 11. Figura Actividad 1. Firebase Cloud Messaging

La Ilustración 11 nos muestra el funcionamiento de la actividad 1. El Arduino nano 33 IoT recibe los datos de los sensores que tiene conectados, realiza un análisis y una transformación de dichos datos (escalados, cambios de unidades…) para almacenarlos en la base de datos de Firebase. En caso de que el valor recogido supere los umbrales establecidos, se prepara un cuerpo de notificación y se informa a Firebase Cloud

Messaging para que notifique a cada terminal Android que tiene nuestra aplicación instalada.

Necesitamos tener disponible una conexión WiFi, con conexión a internet, ya que el acceso a los servidores de Firebase se realiza a través de internet.

Actividad 2. Blog WordPress a Arduino MQTT

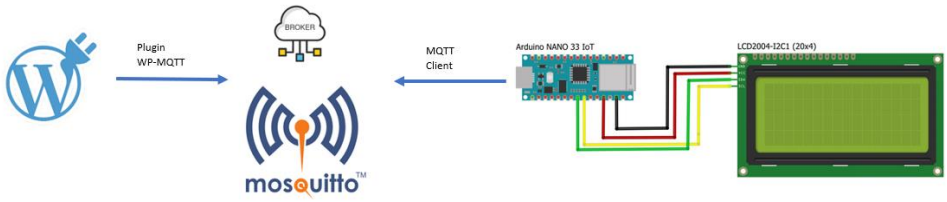


Ilustración 12. Figura Actividad 2. Blog Wordpress a Arduino MQTT

Como podemos observar en la ilustración 12, disponemos de un blog de WordPress con el complemento WP-MQTT instalado y configurado para que recoja ciertos eventos y publique dichos eventos en un tópico especificado de un bróker MQTT. En este caso se está utilizando el bróker público gratuito “mosquitto”.

Por su parte, el Arduino tiene incluida la librería MQTT Client, la cual se encuentra suscrita al mismo tópico en el que se encuentra publicando WP-MQTT. Una vez se recibe un mensaje, este es mostrado en el LCD de 20x4.

Necesitamos tener disponible una conexión WiFi, con conexión a internet, compatible con el Arduino, ya que el bróker MQTT está alojado en un sitio público al que accedemos a través de internet.

Actividad 3. Dashboard

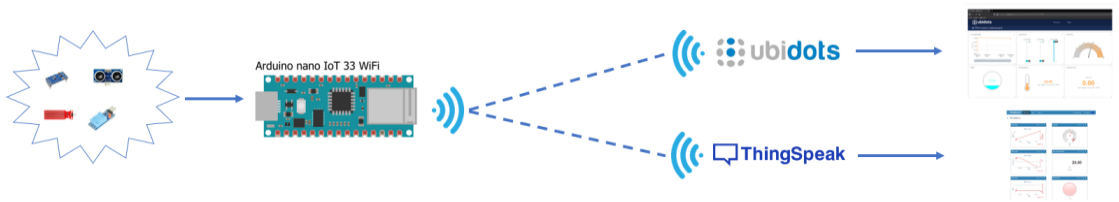


Ilustración 13. Figura Actividad 3. Dashboard

Por último, la Ilustración 13 nos muestra el esquema de la actividad 3. El Arduino nano 33 IoT recibe los datos de los sensores que tiene conectados, realiza una serie de transformaciones, cadenas de inserción en el formato esperado por las plataformas, para poder insertar en Ubidots y en ThingSpeak dichos datos.

Por su parte, ambas *Clouds* tienen configurados unos *dashboards* para mostrar los valores.

Nuevamente, necesitamos tener una conexión WiFi con internet que sea compatible con nuestro Arduino, ya que necesitamos mandar los valores de los sensores a las dos plataformas mencionadas anteriormente.

Diseño detallado

En este capítulo, vamos a explicar todas las actividades con detalle.

En primer lugar, comentaremos la **descarga de datos de eSGarden**. En la siguiente ilustración, se muestra el diagrama de flujo:

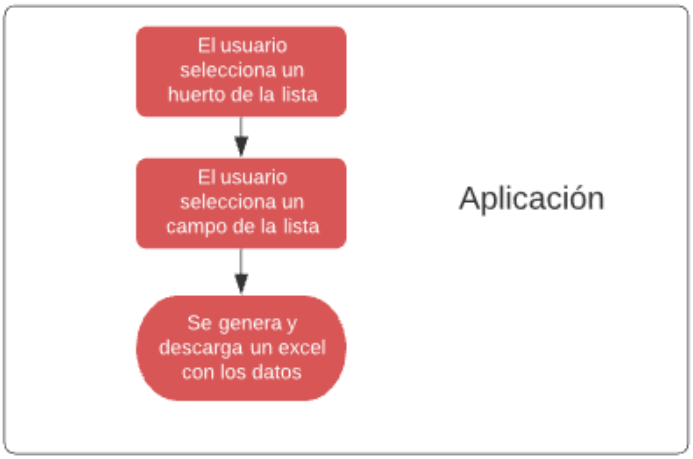


Ilustración 14. Diagrama de flujo descarga de datos eSGarden

A continuación, se muestra el diagrama de secuencia:

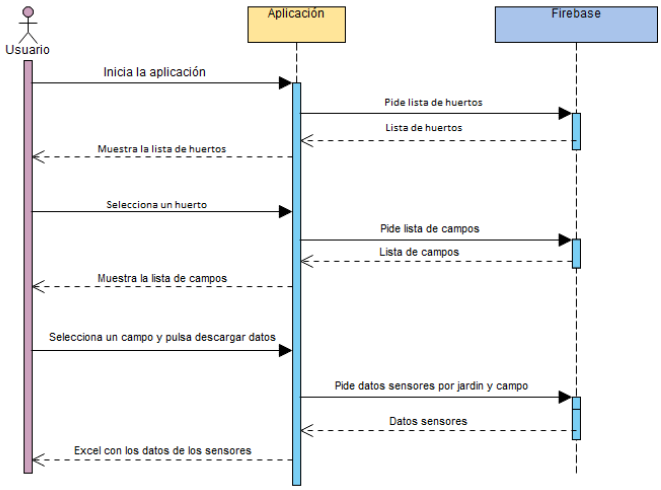


Ilustración 15. Diagrama de secuencia descarga de datos eSGarden

Como se puede observar en la ilustración 15, cuando el usuario inicia la aplicación, pide la lista de huertos a Firebase y se muestran en pantalla. A continuación, si el usuario así lo desea, puede seleccionar un huerto y se cargarán los campos de dicho huerto. Finalmente, puede seleccionar un campo y pedir la descarga de datos. En ese caso, se piden los datos a Firebase y se crea el fichero Excel con dichos datos.

Dado que se trata de una sencilla aplicación, dentro de los requisitos, se incluían los tres casos de usos que se ven en la siguiente ilustración:

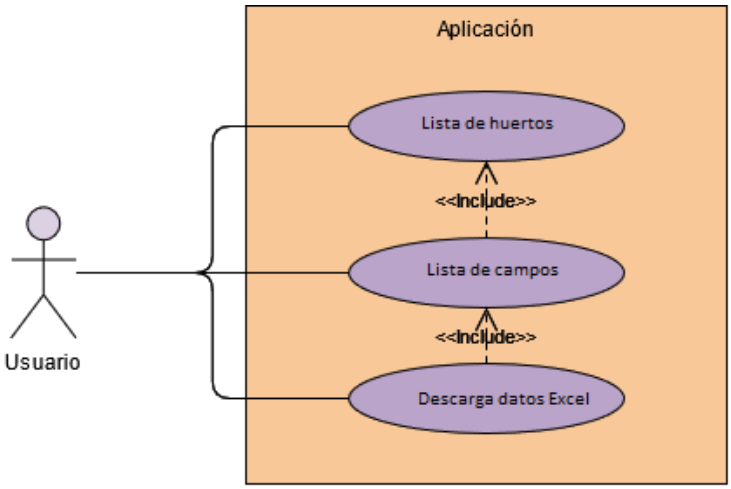


Ilustración 16. Diagrama casos de uso descarga de datos eSGarden

Como se observa en la ilustración, el usuario puede acceder a la aplicación para conocer solo el nombre de los huertos, conocer el nombre de los campos de un huerto, previa selección de este, o realizar la descarga de los datos.

El código fuente de la aplicación se encuentra en el siguiente enlace: https://github.com/antonigimenezrodriguez/eSGarden_DownloadFirebase.git

La aplicación se ha desarrollado en lenguaje C# usando el IDE Visual Studio de Microsoft. Dado que se trata de una aplicación sencilla, no da pie a aplicar patrones de diseño complejos, aunque hemos aplicado el patrón repositorio para el acceso a datos. Por tanto, hemos aplicado el principio de responsabilidad única, es decir, cada método u objeto tiene que realizar una única cosa.

Las clases que se encargan de la vista realizan distintas peticiones a los servicios según las necesidades (pedir los datos a Firebase, generar Excel...). Para pedir los datos a Firebase, se ha diseñado un repositorio que contiene el acceso a Firebase. Para general la hoja de cálculo se ha creado un servicio que genera un Excel a partir de unos datos proporcionados.

Con este principio, se ha conseguido aislar la responsabilidad de cada clase y método. Tenemos localizado en una única clase la creación del cliente de Firebase y el acceso a datos, en otro servicio, la creación del fichero Excel.

En segundo lugar, continuaremos con la **Actividad 1. Firebase Cloud Messaging.** Empezaremos con el diagrama de flujo:

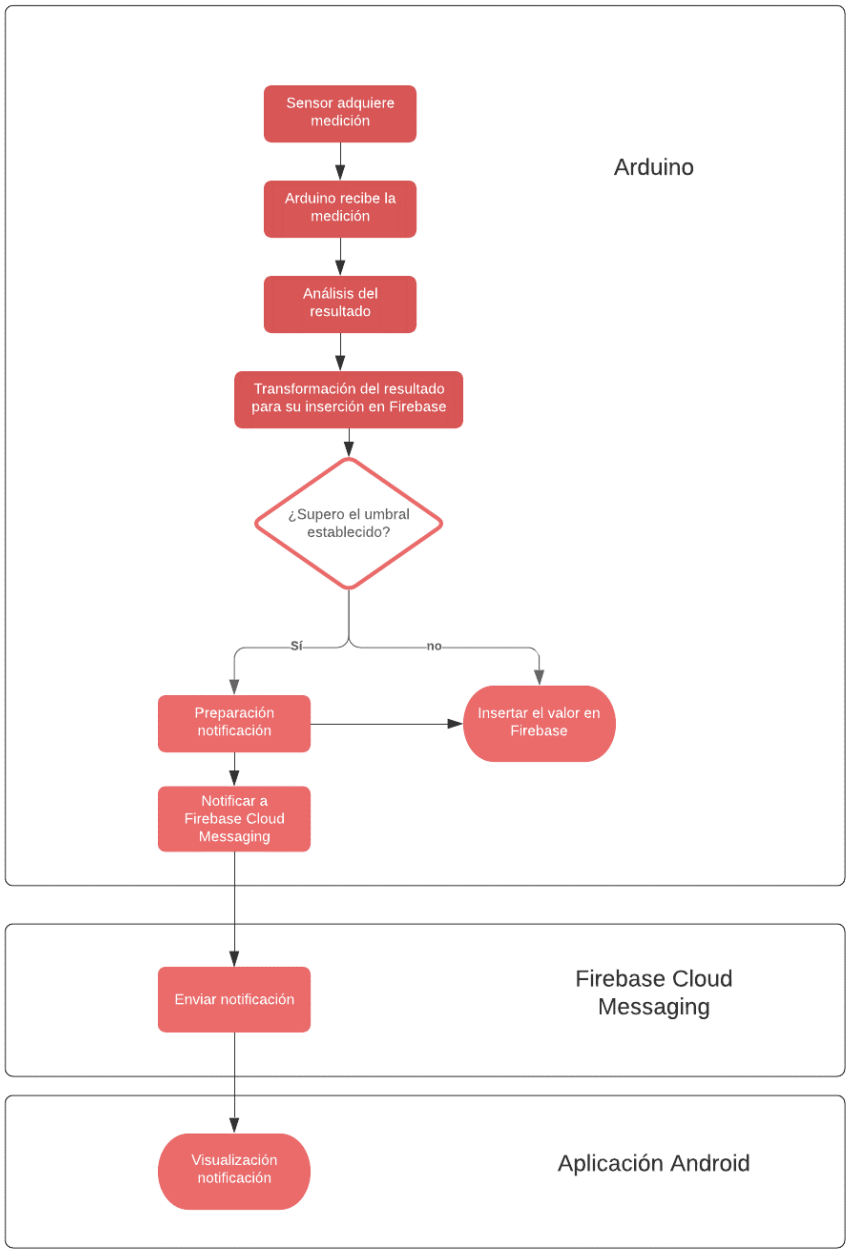


Ilustración 17. Diagrama de flujo Actividad 1. Firebase Cloud Messaging

A continuación, observaremos el diagrama de secuencia para esta actividad:

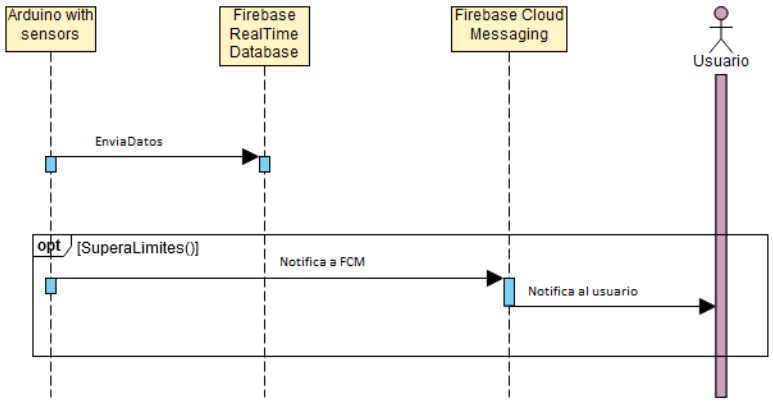


Ilustración 18. Diagrama de secuencia Actividad 1. Firebase Cloud Messaging

Al tratarse de un sistema embebido, como la gran mayoría de sistemas embebidos, no hay interacción directa con el usuario. En este caso, la primera secuencia, enviar datos a Firebase y comprobar los límites, se realiza en cada iteración del Arduino. En caso de que se superen dichos límites, se notifica a Firebase Cloud Messaging con los datos del sensor y se notifica al usuario mediante la aplicación de Android diseñada.

El código de Arduino de la implementación está disponible en el enlace: https://github.com/antonigimenezrodriguez/Arduino-IoT-33-FCM.git

El código de la aplicación para Android de la implementación está disponible en el enlace:
https://github.com/antonigimenezrodriguez/AndroidFirebase_Notifications.git

La parte de Arduino consiste en la realización de una biblioteca diseñada para integrarse en el proyecto eSGarden. Así, cuando se reciba un valor fuera del rango predefinido, se notifica a Firebase Cloud Message.

La aplicación para Android está diseñada en el IDE Android Studio, y contiene una ventana de inicio de sesión que se conecta a la autenticación de Firebase y guarda el token del dispositivo en la base de datos de Firebase. La única funcionalidad que se ha configurado es la de mostrar las notificaciones de Firebase Cloud Messaging.

Dado que la aplicación contiene un sistema de inicio de sesión, la política de privacidad es la misma que la de eSGarden («Privacy Policy of our APP | School Gardens for Future Citizens» 2020). En el Anexo V. Privacidad de nuestra aplicación se encuentra la traducción de dicha política.

Continuaremos con la **Actividad 2. Blog MQTT a Arduino.** Seguidamente se muestra el diagrama de flujo:

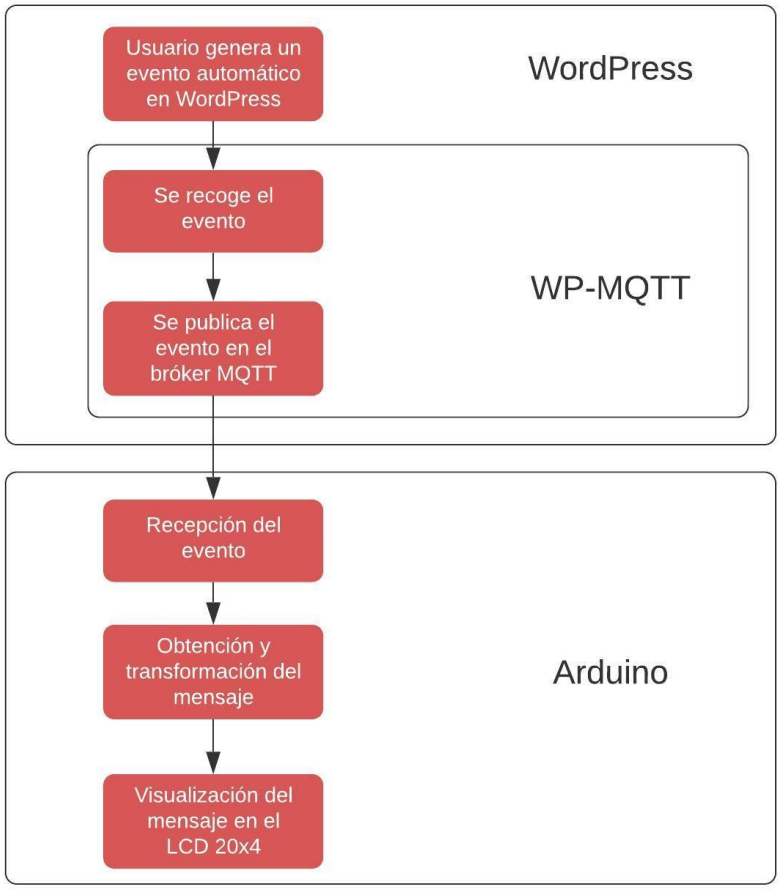


Ilustración 19. Diagrama de flujo Actividad 2. Blog MQTT a Arduino

Seguiremos con el diagrama de secuencia:

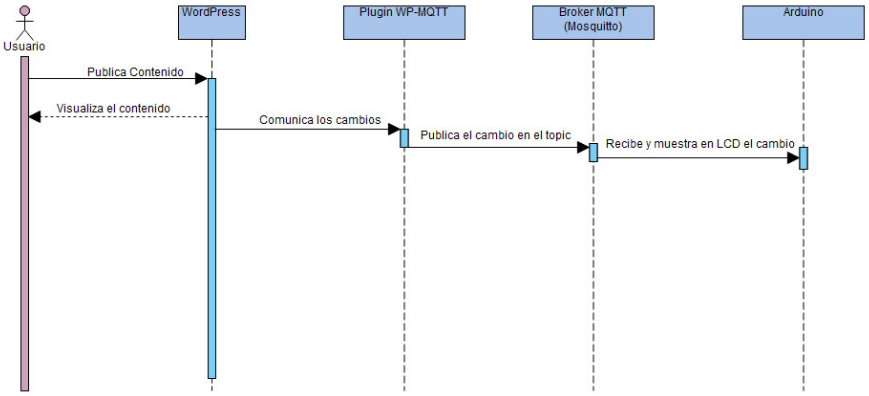


Ilustración 20. Diagrama de secuencia Actividad 2. Blog MQTT a Arduino

Esta actividad se divide en dos sistemas, por una parte, el blog WordPress montado en una máquina virtual sobre el sistema operativo CentOS 7 y, por otra parte, la codificación del Arduino.

Cuando se produce uno de los siguientes eventos en el blog:

- Inicio sesión de usuarios
- Fallo al iniciar sesión
- Nuevo post
- Nueva página
- Nuevo comentario

el complemento WP-MQTT que hemos instalado en el blog, realiza una nueva publicación en un bróker MQTT que configuramos. En el caso de la actividad se ha decidido usar el bróker gratuito Mosquitto y usando como *topic* de publicación "blogTFMAntoni"

La imagen para la virtualización del sistema se puede encontrar en el siguiente enlace: https://upvedues.sharepoint.com/:u:/s/TFMAntoniGimnez/EcQN8LQFX_JFjKxnFegRvtMB6_Q6F8vaLoIU-3HxhcctKQ?e=07NasG

NOTA: el fichero con la virtualización solo es accesible desde una cuenta de la Universitat Politècnica de València.

En la parte Arduino, se ha escrito el código necesario para suscribirse al *topic* mencionado anteriormente y mostrar en la pantalla LCD el contenido de dichos mensajes.

El código está accesible en el enlace que sigue a continuación: https://github.com/antonigimenezrodriguez/Arduino-IoT-33-MQTT.git

Finalizaremos con la **Actividad 3. Dashboard**. El diagrama de flujo para esta actividad es el siguiente:

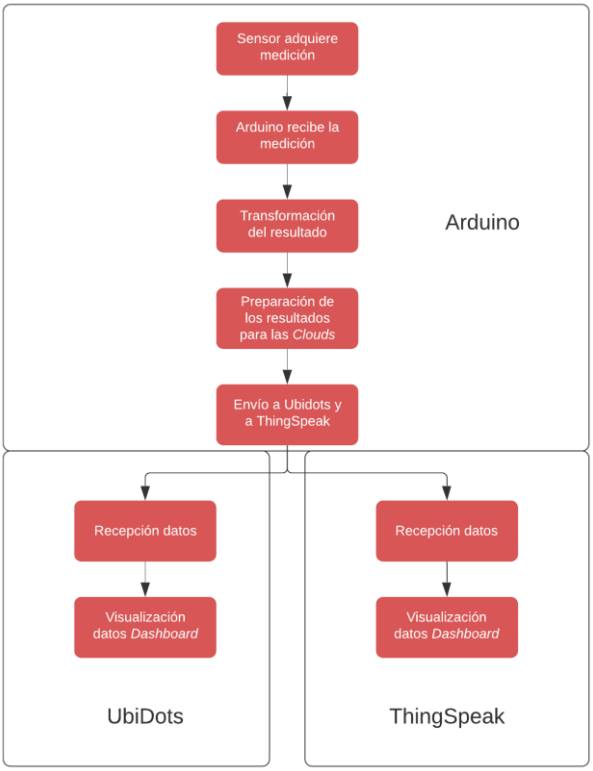


Ilustración 21. Diagrama de flujo Actividad 3. Dashboard

A continuación, se observa el diagrama de secuencia de la actividad:

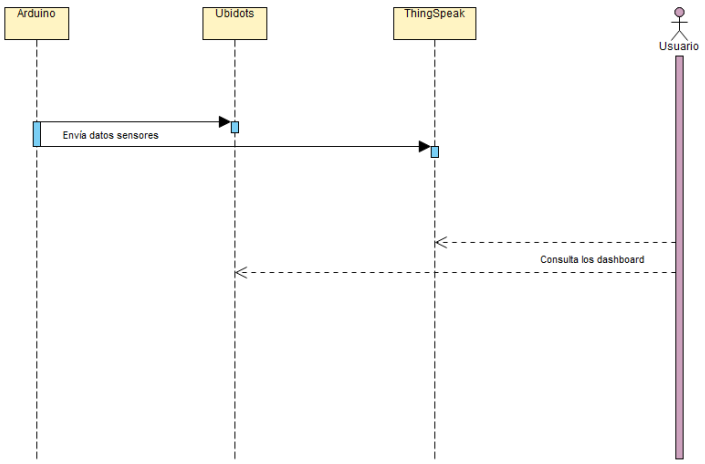


Ilustración 22. Diagrama de secuencia Actividad 3. Dashboard

En esta actividad, tenemos un montaje en una *protoboard* con cuatro sensores (sensor de agua, sensor de distancia, sensor de temperatura y humedad, DHT-11, y un giroscopio) conectados a un Arduino nano IoT 33 WiFi.

El código de Arduino está disponible en este enlace: https://github.com/antonigimenezrodriguez/Arduino-IoT-33-Dashboard.git

Constantemente estamos recogiendo los valores de dichos sensores, adecuando los valores obtenidos (escalado, cambio de unidades…) y preparando una cadena de texto en el formato que precisan las plataformas ThingSpeak y Ubidots.

Una vez realizadas las transformaciones mencionadas anteriormente y preparada la cadena de envío, se envía dicha información a las plataformas. Tanto en ThingSpeak como Ubidots, se ha configurado un *dashboard* que, a partir de los valores recibidos, muestra, en tiempo real, distintos gráficos (de líneas, termómetro, tanque de agua…).

El usuario puede consultar cuando lo desee los *dashboards*, estos están recibiendo y almacenando los datos en segundo plano, aunque el usuario no esté viéndolo en ese momento.

Tecnología utilizada

Para la elaboración de este trabajo se ha requerido de diversas tecnologías y portales. A continuación, se indica una breve descripción:

Hardware

Arduino nano IoT: Es una placa de circuito impreso pequeña, potente y robusta con conectividad WiFi o Bluetooth, junto con una arquitectura de bajo consumo de energía. Es una solución práctica y económica para proyectos de IoT.

Protoboard: Es una placa con orificios que están conectados eléctricamente y consta de un patrón recto en el que se pueden insertar componentes electrónicos y cables para ensamblar y crear prototipos de circuitos electrónicos y sistemas similares.

ADXL345: Es un acelerómetro de 3 ejes pequeño, delgado y de potencia ultra baja, con medición de alta resolución de hasta ± 16 g. Los datos de salida digital están formateados como complemento a dos de 16 bits y son accesibles a través de una interfaz digital SPI o I2C.

HC-SR04: Es un sensor de distancia ultrasónico de baja precisión. Su uso está indicado para medir distancias fácilmente y de una manera rápida.

Sensor de agua: Es un módulo de detección de la profundidad del agua cuyo componente principal es un circuito formado por transistores y cables metálicos en la placa.

DHT11: Es un sensor digital para medir temperatura y humedad con Arduino.

Pantalla LCD 20x4: Es un componente para mostrar en pantalla caracteres alfanuméricos y otros símbolos en un formato de 20 caracteres por línea, en 4 líneas.

Software

Microsoft Visual Studio: Es un IDE para Windows y macOS. Compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que con entornos de desarrollo web, como ASP.NET MVC y Django entre otros.

.NET Core: Es un *framework* de código abierto gratuito para los sistemas operativos Windows, Linux y MacOS. Es el sucesor multiplataforma de .NET Framework.

C#: Es un lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft como parte de la plataforma .NET.

Microsoft Teams: Es una plataforma de comunicación y colaboración que combina un chat fluido, reuniones de videollamadas, almacenamiento y administración de archivos remotos e integración de aplicaciones.

Git: Es un software diseñado para un control de versiones eficiente y confiable. Su propósito es realizar un seguimiento de los cambios y compartir archivos en el repositorio de código para que el trabajo se pueda coordinar entre varias personas.

GitHub: Es un portal que facilita un espacio en la nube para subir repositorios propios y compartir el código con otras personas, de tal forma que sea accesible desde internet. Permite ver fácilmente proyectos de otros usuarios, valorarlos y proponer mejoras en el código.

SourceTree: Es un cliente de Git gratuito para Windows y Mac. Simplifica la forma de interactuar con los repositorios Git para centrarse en la codificación.

Firebase: Es una plataforma de desarrollo de aplicaciones web y móviles. Está en la nube, integrado con Google Cloud Platform y utiliza un conjunto de herramientas para crear y sincronizar proyectos.

Android Studio: Es el entorno de desarrollo nativo y oficial para el desarrollo de aplicaciones en Android.

Arduino: Es un entorno de desarrollo integrado multiplataforma utilizado para programar y cargar código en una placa compatible con Arduino.

Desarrollo de la solución propuesta

En esta sección, vamos a mostrar una serie de fotos y vídeos de las implementaciones finales.

Para descargar la base de datos de Firebase podemos encontrar un vídeo demostrativo en el siguiente enlace: https://www.youtube.com/watch?v=bq27FyusN-Q.

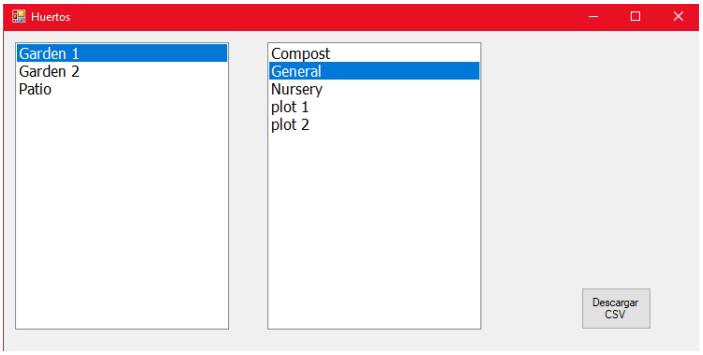


Ilustración 23. Interfaz aplicación descarga Base de Datos de Firebase

La ilustración 23 muestra un ejemplo de la interfaz de la aplicación desarrollada.

A continuación, podemos observar en la ilustración 24, un ejemplo de notificación recibida en el dispositivo Android, en este caso, cuando el sensor de CO2 ha superado el límite.

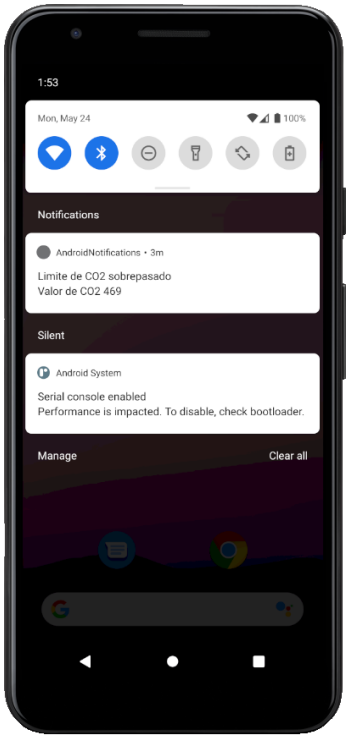


Ilustración 24. Ejemplo Notificación Firebase Cloud Messaging

En el siguiente enlace se puede observar una demo del funcionamiento: https://www.youtube.com/watch?v=LEQ9MU_Ac2w.

Seguiremos con la implementación de la Actividad 2.

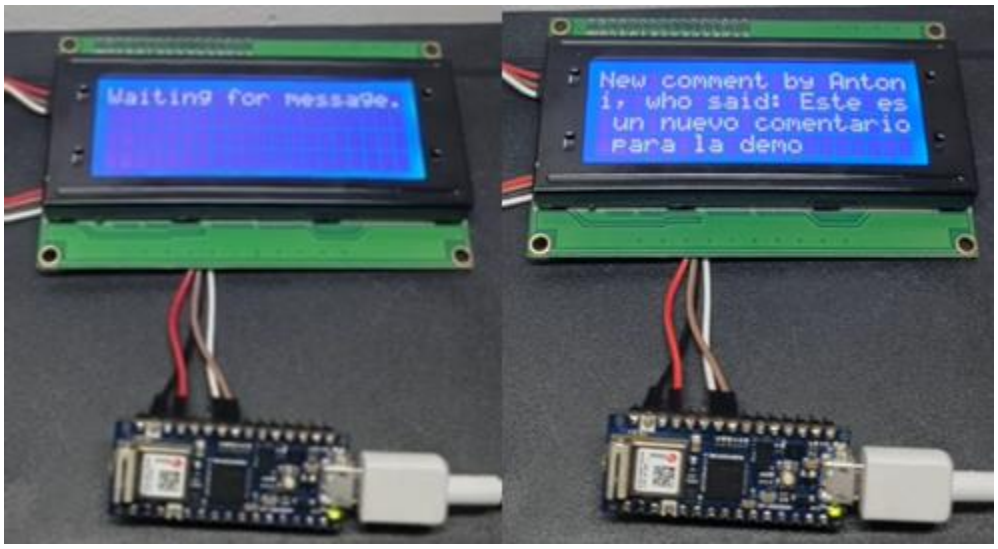


Ilustración 25. Implementación Actividad 2

En la ilustración 25 podemos observar el montaje hardware del Arduino junto con la pantalla LCD. La parte izquierda muestra cuando el Arduino se encuentra esperando un mensaje, mientras que en la parte derecha cuando se ha recibido un mensaje.

Podemos ver una demo del funcionamiento completo en el siguiente vídeo: https://www.youtube.com/watch?v=VqdyaLoTHI8.

Finalmente, observaremos el resultado de la actividad 3.

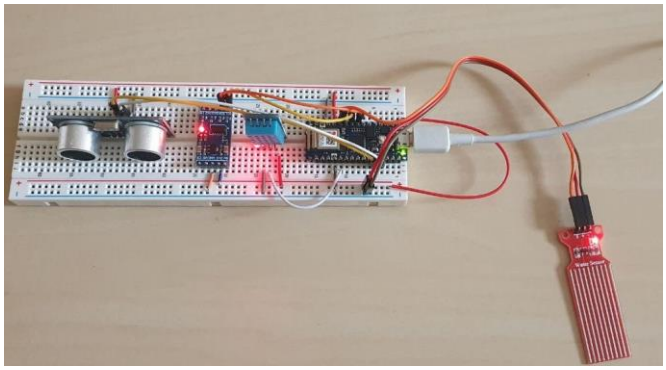


Ilustración 26. Implementación Actividad 3

La ilustración 26 muestra el montaje hardware de los 4 sensores, Sensor de Agua, HC-SR04, DHT11 y ADXL-345, junto con la placa *protoboard* y el Arduino.

En el siguiente enlace: https://www.youtube.com/watch?v=jV6E6JMHLQE encontraremos una video demostración de dicha actividad.

5. Implantación

Este proyecto se encuentra en modo piloto y aún no ha podido ser implantado en un entorno real. Dado que las actividades están ambientadas en un entorno académico, la puesta en marcha de las actividades se debe realizar en un aula con alumnos de ESO y Bachillerato.

Como se menciona en la introducción de este trabajo, durante el curso académico 2021-2022 tengo previsto realizar el Máster de Profesorado, donde espero, poder implementar estas actividades durante la realización de las prácticas en el centro educativo.

En caso de no poder aplicar las actividades, trataré de realizar la programación didáctica para las futuras oposiciones, basándome en estas actividades.

Pruebas

Dado que nuestro proyecto, en la gran mayoría de su desarrollo está centrado en Arduino, no podemos realizar pruebas de caja blanca ni caja negra. Por esto mismo, nos hemos centrado en realizar pruebas de tiempo. Todas las pruebas de tiempos se han realizado con el siguiente hardware:

Equipo: Ordenador portátil HP Probook 450 G8
Procesador: i7-1165G7 2.80 GHz 4 núcleos/8 hilos
Memoria RAM: 32 Gb DDR4-3200 (1600 MHz)
Tarjeta gráfica: Intel Iris Xe Graphics
Sistema Operativo: Windows 10 Pro 64-bits versión 21H1

Hemos creado unas tablas para representar los tiempos de carga de las distintas actividades.

Descarga de la base de datos eSGarden:

Tabla 3. Toma de tiempos aplicación escritorio

Tarea	Número de Repeticiones	Tiempo medio (s)	Tiempo máximo (s)	Tiempo mínimo (s)	RAM (Mb)	CPU (%)
Carga de huertos	1.000	1,27	2,02	0,86	5	11
Carga de campos	1.000	0,70	0,95	0,65	3	2
Descarga de datos	1.000	0,59	0,72	0,50	8	2
Generación de Excel	300	11,45	13.25	9,05	24	40

La Tabla 3 corresponde a la toma de datos realizada para la aplicación de descarga de la base de datos.

Como podemos apreciar en la tabla, los tiempos de carga son relativamente cortos, a excepción de la generación del fichero Excel. Esto se debe a que se tienen que recorrer uno a uno todos los datos de Firebase y situarlos uno a uno en las distintas celdas de la hoja Excel.

NOTA: Para el consumo de RAM y el porcentaje de CPU, se han tenido en cuenta los picos obtenidos durante el proceso.

Actividad 1:

Tabla 4. Toma de tiempos Actividad 1

Repeticiones	Tiempo medio (s)	Tiempo mínimo (s)	Tiempo máximo (s)
500	3,31	2,98	4,23

La Tabla 4 resume el tiempo en mandar una notificación a los terminales que tienen instalada nuestra APP.

Actividad 2:

Tabla 5. Toma de tiempos actividad 2

Número de caracteres	Número de repeticiones	Tiempo medio (s)	Tiempo mínimo (s)	Tiempo máximo (s)
1	500	0,10	0,10	0,10
2	500	0,20	0,20	0,21
5	500	0,51	0,51	0,52
10	500	1,03	1,02	1,03
20	500	2,05	2,05	2,06
40	500	4,10	4,10	4,11
80	500	8,21	8,20	8,22

La Tabla 5 muestra los tiempos de carga para recuperar los datos del bróker MQTT y su escritura en el LCD. Como se puede observar, cada carácter que se muestra en el LCD incrementa 103 milisegundos (0.1 segundos) de media.

Observamos que este incremento es lineal con los distintos números de caracteres. Esto se debe, principalmente, a que en cada escritura de carácter, estamos haciendo una espera de 100 milisegundos para crear el efecto de escritura retardada.

Actividad 3:

Tabla 6. Toma de tiempos actividad 3

Plataforma	Número de repeticiones	Tiempo medio (s)	Tiempo mínimo (s)	Tiempo máximo (s)
Ubidots	500	3,61	3,125	4,650
ThingSpeak	500	0,64	0,601	0,750

En la Tabla 6 se observa el tiempo necesario para enviar los datos a las distintas *clouds*. En este caso nos llama la atención la gran diferencia que existen entre ellas. Investigando la posible causa, se ha llegado a la conclusión que para insertar datos en Ubidots se hace mediante una llamada POST a un API REST, mientras que para realizar la inserción en ThingSpeak, existe una librería específica que optimiza todo el proceso.

6. Conclusiones

A modo de conclusión, podemos afirmar que se ha completado el desarrollo de las tres actividades propuestas inicialmente, así como la aplicación para la descarga de datos.

En cuanto a la primera actividad se refiere, se ha diseñado y desarrollado un sistema capaz de transmitir en tiempo real a dispositivos móviles Android valores anómalos de un sistema de sensorización en Arduino.

En la segunda actividad, se ha desarrollado un sistema complejo que comprende desde la creación de un blog WordPress bajo un servidor propio virtual con VirtualBox hasta la visualización de datos en un LCD usando las tecnologías MQTT y Arduino.

En la tercera actividad, se ha completado con éxito la realización de una composición de diversos sensores ampliable según necesidades. En este caso, los sensores que se han elegido, han sido seleccionados por la forma en la que entregan los datos (Conexión I2C, valores positivos y negativos, valores decimales, analógicos con calibración manual…) y se muestra en tiempo real sus valores y variaciones en un portal con distintas gráficas de visualización.

Finalmente, en lo que respecta a la aplicación, se ha diseñado un sistema capaz de facilitar la descarga de datos de Firebase. En este caso, se ha cumplido con todos los requisitos iniciales. A pesar de ser una aplicación sencilla, se han aplicado principios de código limpio, patrón repositorio y el principio de responsabilidad única de los métodos y clases, facilitando el mantenimiento de esta, así como la potencial ampliación de funcionalidades.

Durante todos los desarrollos de código, hemos usado el sistema Git para mantener un histórico de los cambios, al tiempo que nos permite disponer de un repositorio donde alojar y compartir nuestro código.

Tal y como se ha descrito en capítulos anteriores, podemos concluir que hemos logrado todos los objetivos que se propusieron al inicio del proyecto. Además, hemos usado tecnologías actuales para el IoT, como es Arduino y que, además, se encuentran muy extendidas en el ámbito educativo.

En general, se trata de un amplio proyecto en el que se plantean distintos enfoques, según la actividad, así como distintas tecnologías. Es por eso que, durante el proyecto, se han realizado diversas tareas de gestión, análisis e investigación de tecnologías, análisis de viabilidad, desarrollo y validaciones, con el fin de cumplir todos los objetivos marcados desde un inicio.

A nivel personal, la realización de dicho proyecto ha supuesto un desafío muy grande, ya que ha sido mi primera toma de contacto con placas Arduino, una tecnología en la que quería introducirme desde hace años. La he afrontado con muchas ganas, entusiasmo y motivación resultando ser una experiencia totalmente positiva y enriquecedora, por lo que considero que he completado satisfactoriamente mi trabajo.

Para finalizar, solo queda combinarlo con los próximos estudios que voy a cursar (Máster en formación del profesorado de secundaria) y poder aplicarlo en las aulas con alumnos.

Relación del trabajo desarrollado con los estudios cursados

Muchos de los aspectos que se han tratado en este proyecto están relacionados con los estudios de Grado en Ingeniería Informática, y, sobre todo, con el Máster Universitario en Ingeniería Informática.

Durante los seis años de estudios universitarios he adquirido diversos conocimientos, habilidades y competencias que me han servido para la realización de este proyecto. Para la realización de este trabajo, me ha sido especialmente útil el aprendizaje en:

Gestión de proyectos: He cursado diversas asignaturas, tanto en el grado como en el máster, en las que he aprendido a gestionar los diversos factores de un proyecto tales como la gestión del tiempo, la parte económica y los recursos humanos. Cabe destacar la realización de un diagrama de Gantt para gestionar de forma adecuada los recursos y el tiempo.

Programación: Dado que en el grado cursé la especialidad de Ingeniería de software, me ha proporcionado el conocimiento adecuado para realizar software de calidad, reutilizable y que contenga el mínimo número de fallos posibles. También aprendimos el uso de herramientas como Git para poder gestionar de una forma profesional nuestro código.

UML: Una de las consideraciones más importante a la hora de llevar a cabo un proyecto, sobre todo si incluye parte software, es la realización de diagramas UML. Dentro de la gran diversidad de diagramas aprendidos, hemos aplicado los diagramas de flujo, de secuencia, casos de uso, así como las figuras de las distintas actividades.

Didáctica: Aparte de las competencias mencionadas anteriormente, encuadraríamos este proyecto dentro de la didáctica aplicada a la informática, donde gracias a las actividades realizadas en la asignatura e-Learning y Redes Sociales, hemos tenido el conocimiento suficiente para poder crear y adaptar actividades, en una primera aproximación, dentro del marco educativo de la Generalitat Valenciana y del marco europeo para las competencias digitales DigComp.

Tal y como se puede observar durante toda la memoria del proyecto, he aplicado también otras competencias transversales adquiridas durante mi etapa universitaria en la realización del mismo. Por todo ello, considero que este trabajo me permite culminar una etapa de mi formación.

Trabajos futuros

Con la presentación de este trabajo de fin de máster, el proyecto no termina. Existen diversas líneas en las que se puede profundizar más en detalle.

Actualmente, nos encontramos preparando una contribución que podamos presentar en un congreso de ámbito internacional en Tecnología Educativa. Los congresos en los que tenemos la intención de participar son: “Congreso Internacional de Tecnología Educativa (EDUTEC)” y “El Simposio Internacional de Informática Educativa (SIIE)”.

La primera línea de trabajo será preparar estas actividades para poder relacionarlas con mi formación en el Máster de Profesorado y preparar la documentación necesaria para poder aplicarlas en mis prácticas como docente.

Otra línea de trabajo sería, partiendo de la recopilación de los datos de los sensores, abordar un análisis más profundo de los mismos. Con este propósito, se decidió usar la plataforma ThingSpeak, ya que forma parte de MatLab (MathWorks - Descripción de MATLAB). Como base, se podría partir de la aproximación para construir un sistema de análisis que nos ofrece Mathworks en su web oficial. (MathWorks - Developing an IoT Analytics System with MATLAB).

Referencias

CARRETERO, S., VUORIKARI, R. and PUNIE, Y., 2017. EUR 28558 EN The Digital Competence Framework for Citizens With eight proficiency levels and examples of use. . S.l.:

FERRARI, A., PUNIE, Y. and BREČKO, B.N., 2013. DIGCOMP: A Framework for Developing and Understanding Digital Competence in Europe. [en línea], DOI 10.2788/52966. Disponible en: http://europa.eu/.

FTACNIK, M., SVEDA, D. and KIRES, M., 2020. Digital transformation of education in Slovakia within the context of European documents. *ICETA 2020 - 18th IEEE International Conference on Emerging eLearning Technologies and Applications, Proceedings*. S.l.: Institute of Electrical and Electronics Engineers Inc., pp. 113–118. ISBN 9780738123660. DOI 10.1109/ICETA51985.2020.9379154.

GENERALITAT VALENCIANA. CONSELLERIA DE HACIENDA Y MODELO ECONÓMICO, 2019. *DOGV Num. 8676 / 13.11.2019 48772*. 2019. S.l.: s.n.

GENERALITAT VALENCIANA. CONSELLERÍA D'EDUCACIÓ, cultura i esport, [sin fecha]. *Currículum Informática ESO Comunitat Valenciana* [en línea]. S.l.: s.n. [Consulta: 31 May 2021]. Disponible en: https://ceice.gva.es/documents/162640733/162655311/Inform%C3%A1tica/9b1b36c1-35b4-4e4c-b95e-06237243a764.

GRINDEI, L., BLANC, S. and BENLLOCH-DUALDE, J.V., 2019. ESGarden-Implementing a school-university collaboration project for inclusive and equitable education through technology. *29th Annual Conference of the European Association for Education in Electrical and Information Engineering, EAEEIE 2019 - Proceedings*. S.l.: Institute of Electrical and Electronics Engineers Inc., ISBN 9781728132228. DOI 10.1109/EAEEIE46886.2019.9000448.

Las grandes estadísticas del Internet de las Cosas (IoT) ~ IoT World Online. [en línea], 2020. [Consulta: 5 June 2021]. Disponible en: https://www.iotworldonline.es/las-grandes-estadisticas-del-internet-de-las-cosas-iot/.

MATHWORKS, [sin fecha]. Descripción del producto MATLAB - MATLAB & Simulink - MathWorks España. [en línea]. [Consulta: 28 May 2021 a]. Disponible en: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html.

MATHWORKS, [sin fecha]. Developing an IoT Analytics System with MATLAB, Machine Learning, and ThingSpeak - MATLAB & Simulink. [en línea]. [Consulta: 31 May 2021 b]. Disponible en: https://es.mathworks.com/company/newsletters/articles/developing-an-iot-analytics-system-with-matlab-machine-learning-and-thingspeak.html.

Model Creation – UrsaLeo. [en línea], [sin fecha]. [Consulta: 11 June 2021]. Disponible en: https://ursaleo.com/model-creation/.

Privacy Policy of our APP | School Gardens for Future Citizens. [en línea], 2020. [Consulta: 30 June 2021]. Disponible en: https://esgarden.blogs.upv.es/privacy-policy/.

RECOMMENDATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. , 2006.

REDECKER, C. and PUNIE, Y., 2017. Digital Competence of Educators DigCompEdu. . S.l.:

The Making of Arduino - IEEE Spectrum. [en línea], 2011. [Consulta: 5 June 2021]. Disponible en: https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino.

VUORIKARI, Riina., PUNIE, Yves., CARRETERO, Stephanie., BRANDE, L.V. den. and EUROPEAN COMMISSION. JOINT RESEARCH CENTRE., 2016. *DigComp 2.0 : the digital competence framework for citizens*. S.l.: Publications Office. ISBN 9789279588761.

Glosario

A

Access Point (AP): Es un hardware WiFi que permite conectarse a la red LAN a través de la conexión inalámbrica.

Analógico: Algo que puede variar continuamente respecto del tiempo y de la magnitud.

Android: Es un sistema operativo diseñado para terminales móviles.

Apache: Es un servidor web HTTP multiplataforma de código abierto.

Application Programming Interface **(API)**: Es la interfaz utilizada para interactuar con un software. Por ejemplo, la API de una biblioteca Arduino son las funciones públicas que están a disposición del usuario.

B

Biblioteca: Es una extensión de software de la API de Arduino que amplía la funcionalidad de un programa.

C

Community ENTerprise Operating System **(CentOS)**: Es una distribución Linux que se trata de un *fork* de GNU/Linux Red Hat Enterprise Linux (RHEL).

D

Depuración: Es el proceso de identificar errores y corregirlos hasta que se logra el comportamiento esperado.

Digital: Es un sistema que se ocupa de valores discretos.

F

Firewall: Es un sistema informático o parte de una red que está diseñado para no permitir el acceso no autorizado. A su vez, se puede configurar para permitir comunicaciones autorizadas.

Firmware: Es similar al software, pero se almacena en una memoria interna ROM no volátil. Se utiliza en sistemas embebidos y se compone de instrucciones de código de máquina. El firmware suele ser actualizable.

Función: Es una serie de líneas de código que ejecuta una tarea concreta.

I

Integrated Development Environmen **(IDE)**: Es el lugar donde se desarrolla el software. En el caso del IDE de Arduino, se usa para escribir código y para cargarlo en una placa Arduino.

Inter Integrated Circuit **(IIC/I2C)**: Es un bus síncrono serie de dos cables que proporciona comunicación entre dos circuitos integrados. Puede conectar diferentes tarjetas simplemente conectando dos cables llamados SDA (línea de datos) y SCL (línea de reloj).

Internet Protocol **(IP)**: Es un conjunto de números que se utilizan para identificar la interfaz de red del dispositivo que utiliza el protocolo, correspondiente al nivel de red del modelo TCP / IP.

ISO: Es un archivo informático que contiene una copia exacta o imágenes de un sistema de archivos.

J

JavaScript Object Notation **(JSON):** Un formato de texto para el intercambio de datos. Este es un subconjunto de la notación de objetos literales de JavaScript, pero debido a que se usa ampliamente como una alternativa a XML, se considera un formato independiente del lenguaje.

M

Máquina Virtual: Es un software que emula un sistema operativo y permite que los programas se ejecuten como si fueran un ordenador físico real.

Message Queuing Telemetry Transport **(MQTT)**: Es un protocolo de red de suscripción/publicación ligero para transmitir mensajes entre dispositivos. El protocolo normalmente se ejecuta sobre TCP / IP.**Microcontrolador:** Es un pequeño ordenador que se programa para recibir, procesar y mostrar información.

Monitor serie: Es una herramienta integrada en el IDE de Arduino, que puede enviar y recibir datos en serie desde y hacia una placa conectada.

My Structured Query Languaje **(MySQL)**: Es Un sistema de gestión de bases de datos relacionales. Se considera una de las bases de datos de código abierto más populares del mundo y es una de las bases de datos más populares.

P

Hypertext Procesor (PHP): Es un lenguaje de programación de uso general adecuado para el desarrollo web.

Plug-in: Es una miniaplicación que se agrega a otros para agregar nuevas funciones. Este complemento lo ejecuta la aplicación principal e interactúa con él a través de la interfaz de programación de la aplicación.

R

Random Acces Memory (RAM): Es una memoria de acceso aleatorio volátil utilizada como memoria de trabajo por los sistemas operativos, los programas y la mayoría del software.

S

Sistema Operativo (SO): Es el software principal de un sistema informático, administra los recursos de hardware y brinda servicios para las aplicaciones de software, se ejecuta en modo privilegiado con respecto a otras aplicaciones.

***Sketch*:** Es el término dado a los programas escritos en el IDE de Arduino.

***Software Development Kit* (SDK):** Es un conjunto de herramientas para el desarrollar software.

***Service Set Identifier* (SSID)**: Es el identificador, o nombre, de una red inalámbrica.

T

Tierra o *Ground* (GND): Es el punto de un circuito donde no hay energía eléctrica. Sin la toma de tierra, la corriente no tendría un lugar para fluir en un circuito.

U

Unified Modeling Language (UML): Es el lenguaje de modelado de software más utilizado y conocido. Son gráficos para visualización, especificaciones y documentación del sistema.

***Universal Serial Bus* (USB):** Es un puerto que es estándar en la mayoría de los sistemas informáticos.

V

V_{CC}: Es el pin utilizado para suministrar el voltaje positivo de la alimentación a un circuito, un dispositivo o una placa.

W

Wi-Fi: es una tecnología para la conexión inalámbrica de dispositivos.

WordPress: Es un sistema de gestión de contenidos enfocado a la creación de todo tipo de sitios web.

***Wi-Fi Protected Acces* (WPA)**: Es un protocolo de cifrado para mantener la transferencia inalámbrica de datos a salvo de posibles escuchas.

Anexos

Anexo I. IDE Arduino

Para realizar las partes correspondientes a la programación de la placa Arduino NANO IoT 33 WiFi, vamos a usar el IDE de escritorio de Arduino, dicho software está disponible en la URL oficial de Arduino:

https://www.arduino.cc/en/software

Una vez instalado, tenemos que configurar el entorno para que funcione con nuestra placa Arduino. Por lo tanto, en este anexo vamos a tratar la configuración para la placa Arduino NANO IoT 33 WiFi. Los pasos para seguir son:

Abrimos el IDE de Arduino, conectamos la placa Arduino nano IoT 33 y observaremos un mensaje como el siguiente:

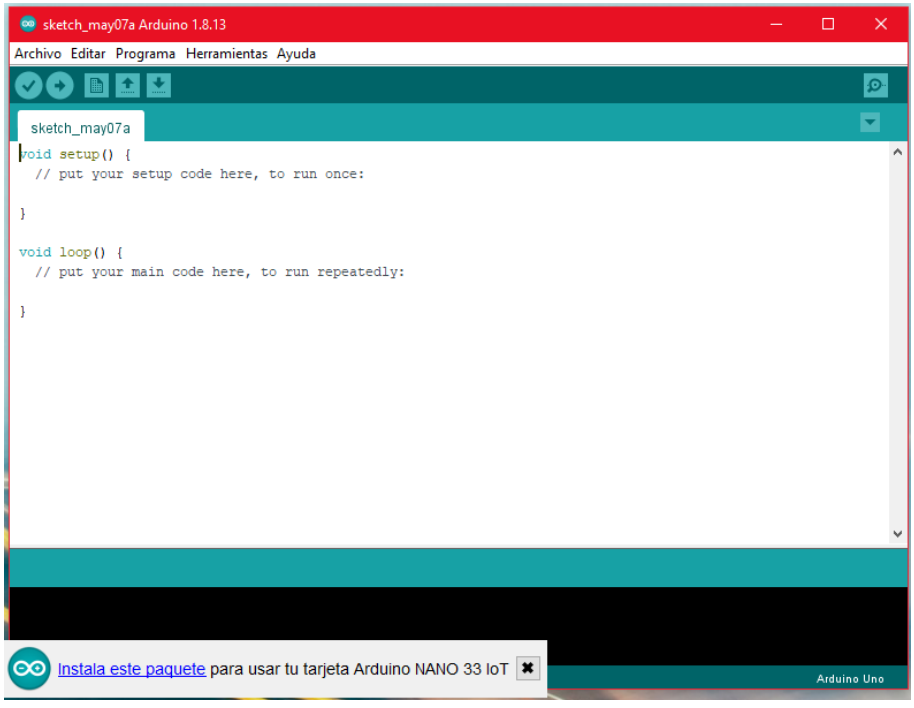


Ilustración 27. IDE Arduino

Pulsaremos en “Instalar paquete” y se abrirá al gestor de tarjetas con el paquete que tenemos que instalar prefijado.

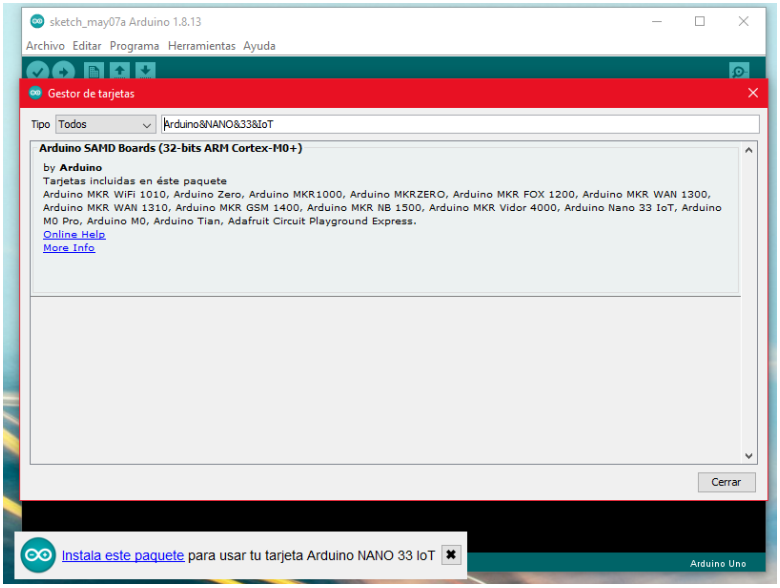


Ilustración 28. Gestor de tarjetas

En caso de no aparecernos esta ventana emergente, accederemos al gestor de tarjetas desde el menú / herramientas / placa / Gestor de tarjetas.

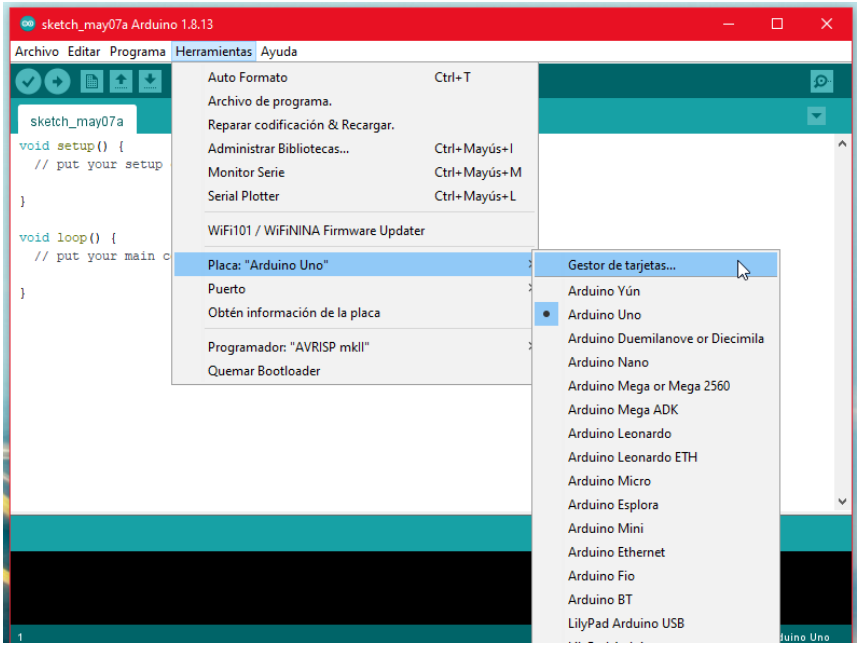


Ilustración 29. Menú gestor de tarjetas

A continuación, se nos mostrará un cuadro diálogo para filtrar por tarjetas, escribiremos el siguiente filtro: "**Arduino&NANO&33&IoT**" e instalaremos el paquete: "**Arduino SAMD Boards (32-bits ARM Cortex-M0+)**"

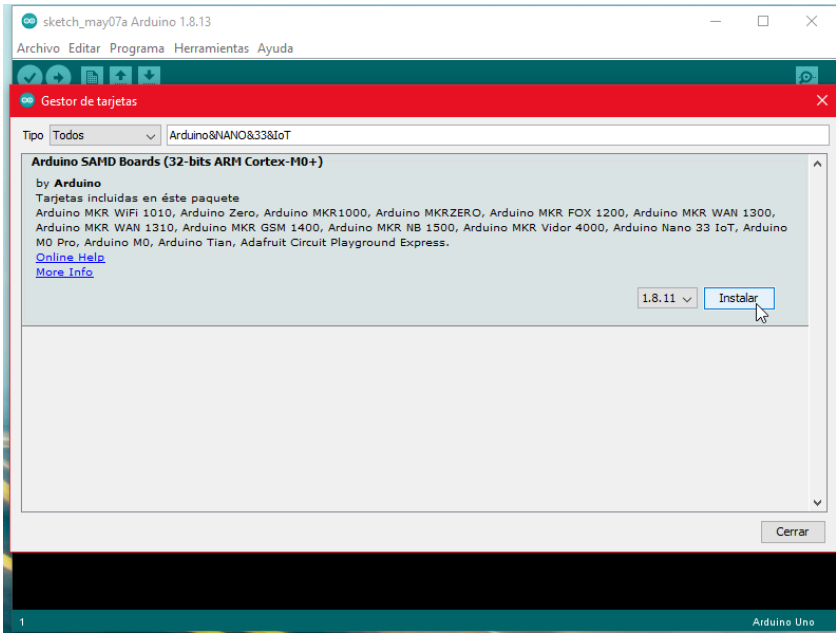


Ilustración 30. Paquete Arduino SAMD Boards

Una vez instalada la tarjeta, seleccionaremos nuestra placa en el menú Herramientas / Placa / Arduino SAMD (32-bits ARM Cortex-M0+) y seleccionar Arduino NANO 33 IoT.

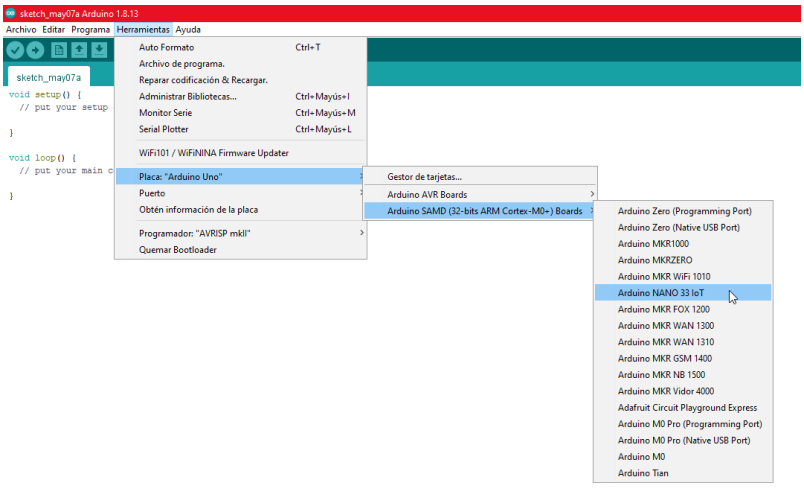


Ilustración 31. Menú selección placa Arduino

A continuación, especificaremos el puerto COM en el que se encuentra conectada la placa. Para ello, accederemos al Administrador de Dispositivos de Windows, buscando la opción "Puertos (COM y LPT)" y anotaremos el número COM del dispositivo llamado: Arduino NANO 33 IoT.

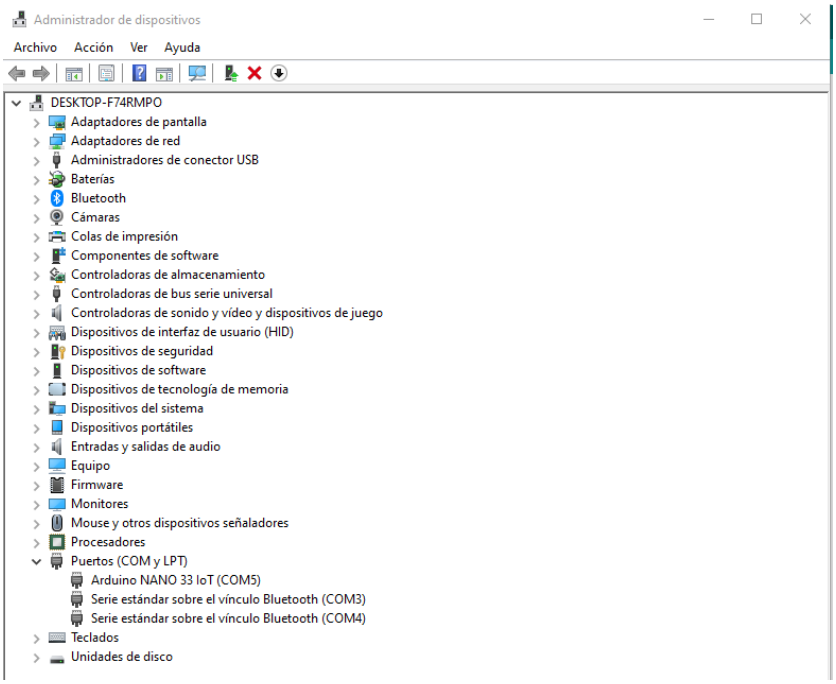


Ilustración 32. Panel Administrador de dispositivos de Windows

Una vez anotado el puerto COM, accederemos al menú Herramientas / Puerto y seleccionaremos el número del puerto COM, en el caso del ejemplo, COM5.

Una vez configurada la placa, procederemos a comprobar que el IDE de Arduino es capaz de ejecutar código en la placa Arduino. Para ello, cargaremos el ejemplo "Blink" que se encuentra en el menú Archivo / Ejemplos / 01.Basics / Blink.

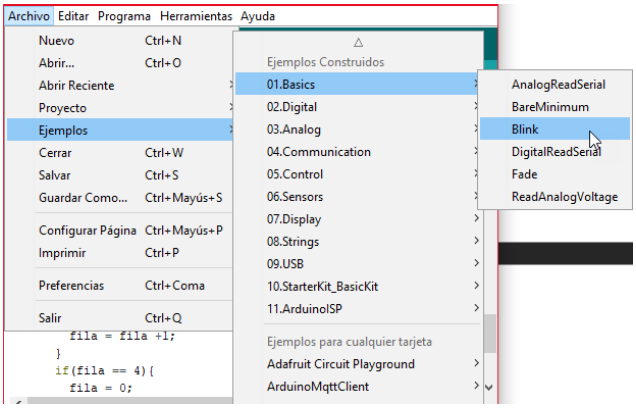


Ilustración 33. Menú ejemplo blink

A continuación, ejecutaremos el código en el Arduino y si el LED naranja empieza a parpadear, significa que todo ha ido correctamente.

El siguiente paso, será la instalación de la librería WiFi. Para ello, accederemos al gestor de librerías mediante la opción de menú Herramientas / Administrar bibliotecas e instalaremos la librería WiFi NINA.

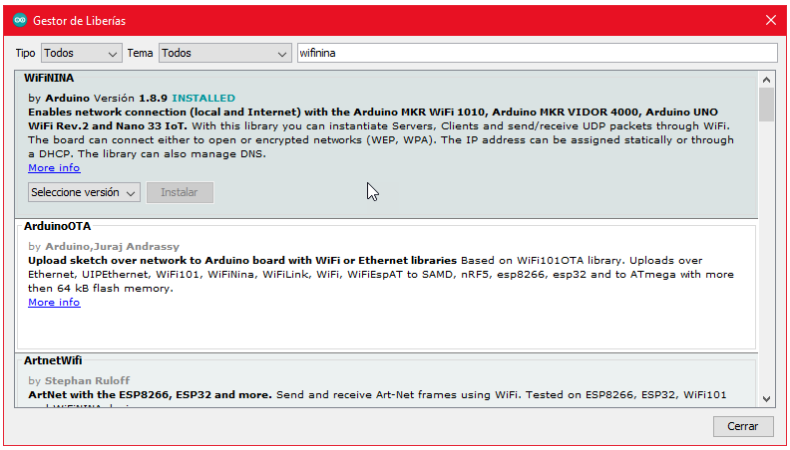


Ilustración 34. Librería WiFi NINA

Una vez realizados los pasos anteriores, ya tenemos configurado el entorno del IDE de Arduino.

Anexo II. Firebase Cloud Messaging

Este anexo se compone de 2 partes. En primer lugar, se explica cómo crear y configurar una aplicación Android usando Android Studio para que pueda recibir las notificaciones de Firebase. En segundo lugar, el guardado de valores de distintos sensores y el envío de una notificación siguiendo unos criterios, en este caso, que el valor recibido se encuentre fuera de un rango

Aplicación Android

El código de la implementación final puede encontrarse en el siguiente repositorio de GitHub:
https://github.com/antonigimenezrodriguez/AndroidFirebase_Notifications.git

Para la realización de la aplicación de Android vamos a usar el SDK Android Studio en su última versión, a la hora de elaborar este trabajo, la última versión es la 4.1.2 que se puede descargar en el siguiente enlace:

https://developer.android.com/studio?hl=es

Crear el proyecto.

Firebase

En primer lugar, tenemos que crear un proyecto en Firebase. Para ello entramos en la consola: https://console.firebase.google.com/ y pulsaremos en Añadir proyecto

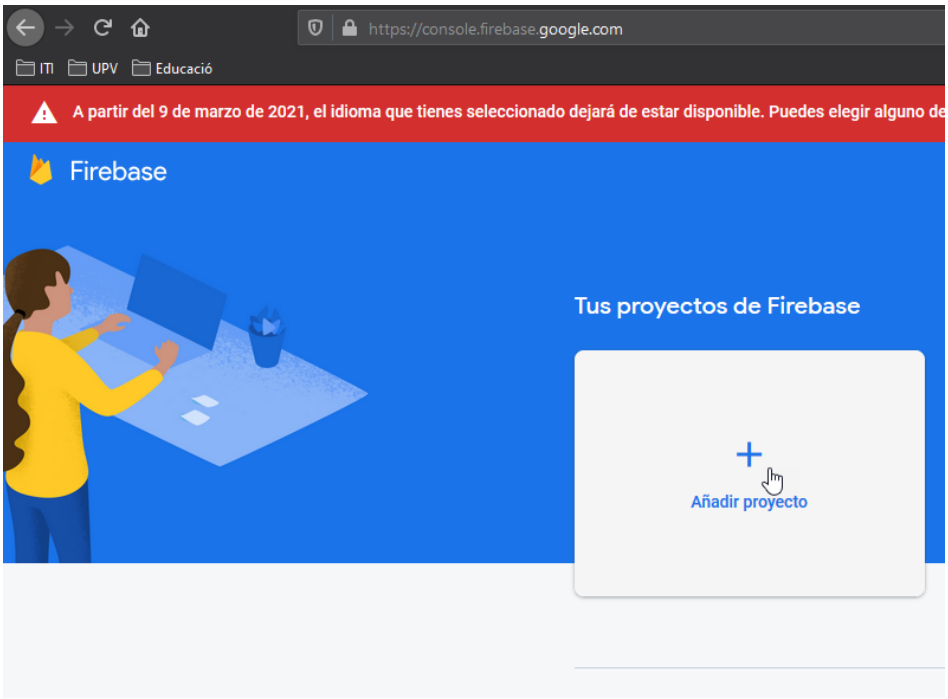


Ilustración 35. Consola de Firebase

Insertamos el nombre del proyecto que queramos y le pulsamos en continuar:

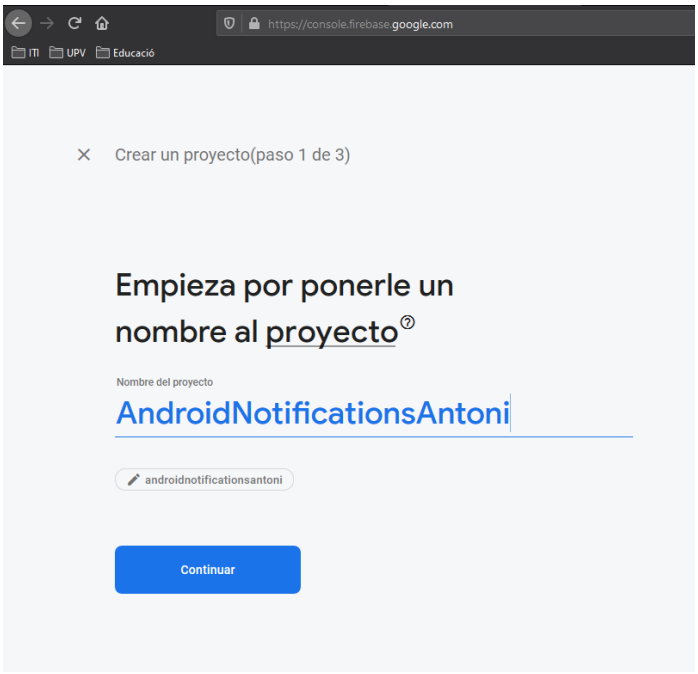


Ilustración 36. Creación proyecto Firebase

Android Studio

A continuación, crearemos un proyecto en Android Studio pulsando en "Create New Project"

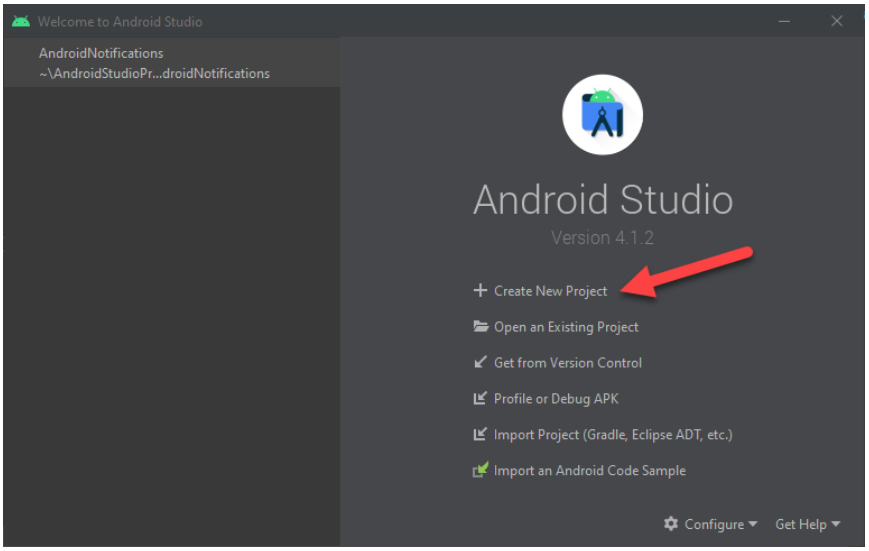


Ilustración 37. Ventana inicio Android Studio

En la siguiente Ventana, seleccionamos “Empty Activity”

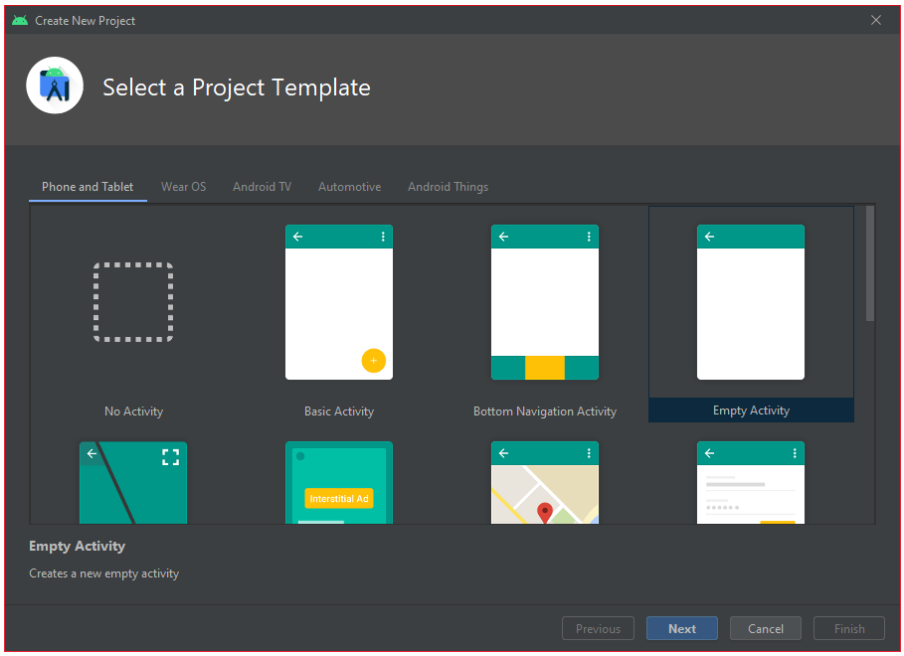


Ilustración 38. Plantilla Proyecto Android Studio

En la siguiente ventana nos va a pedir rellenar los datos básicos del proyecto:

Name: Nombre del proyecto
Package Name: nombre del paquete
Save Location: Dirección donde queremos guardar nuestro proyecto
Languaje: Lenguaje que vamos a utilizar, en nuestro caso Java
Minimum SKD: Versión mínima de Android, vamos a usar API 16

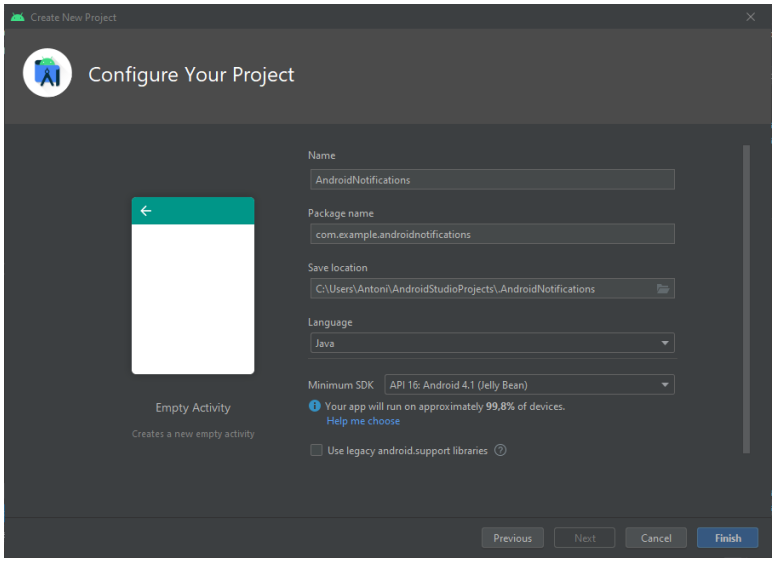


Ilustración 39. Configuración proyecto Android Studio

Cuando hayamos rellenado todos los campos, pulsaremos el botón “finish”. Seguidamente, nos cargará el proyecto que hemos creado en Android Studio.

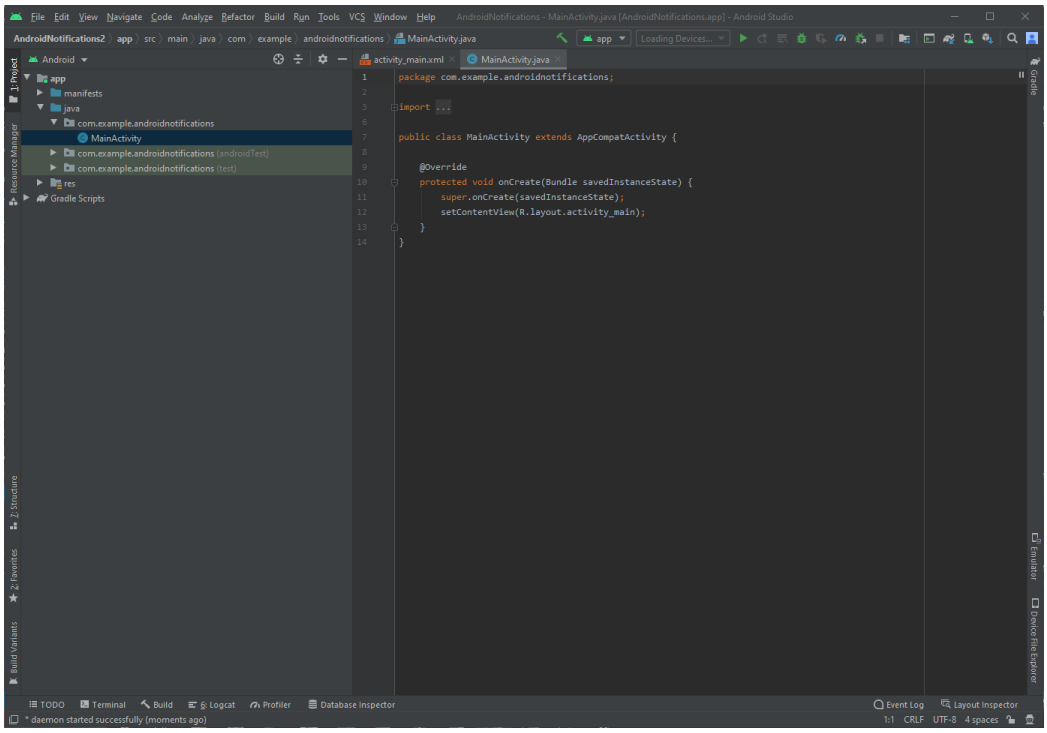


Ilustración 40. Nuevo Proyecto Android Studio

Una vez realizado los pasos anteriores, ya tenemos creados los 2 proyectos necesarios.

Crear una notificación

En primer lugar,

Para crear una notificación necesitamos tener 3 componentes en nuestra APP:

1. Canal de notificación
2. Generador de notificaciones
3. administrador de notificaciones

Para ello, vamos a añadir las siguientes 3 líneas de código en la clase “MainActivity.java”

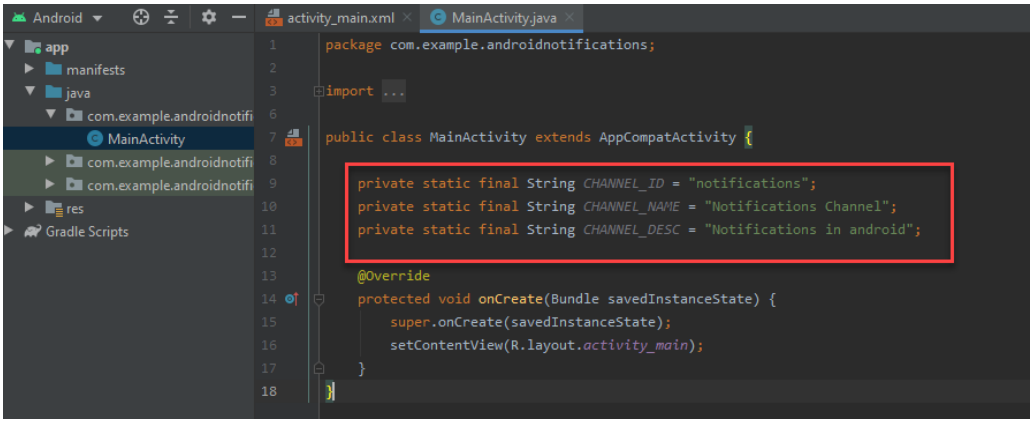


Ilustración 41. Código añadido MainActivity.java

El siguiente paso será crear el método que mostrará dicha notificación. En primer lugar, vamos a añadir el icono de la visitando la siguiente web: https://www.flaticon.es/ y descargaremos el icono que más nos guste en formato SVG. Una vez tengamos descargado el icono en formato SVG, vamos a añadirlo a la librería. Para ello, desplegaremos la carpeta “res” y dentro de “res” buscamos la carpeta “drawable”

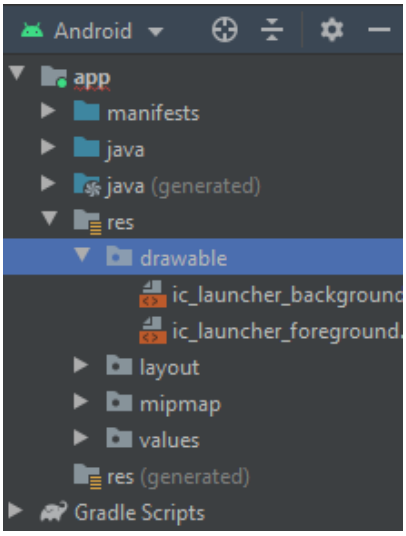


Ilustración 42. Ubicación carpeta "drawable"

Ahora pulsaremos el botón de la derecha del ratón encima de la carpeta “drawable” new / Vector Asset

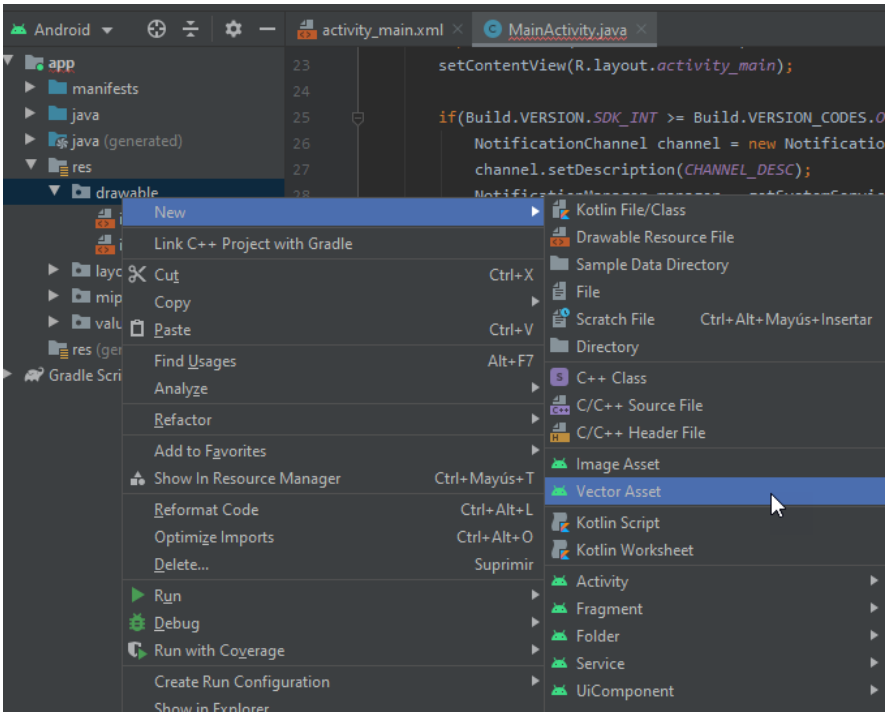


Ilustración 43. Menú inserción Vector Asset

En la nueva ventana, seleccionaremos “Local file (SVG, PSD)”, le daremos un nombre y buscamos la ruta donde nos hemos descargado el archivo

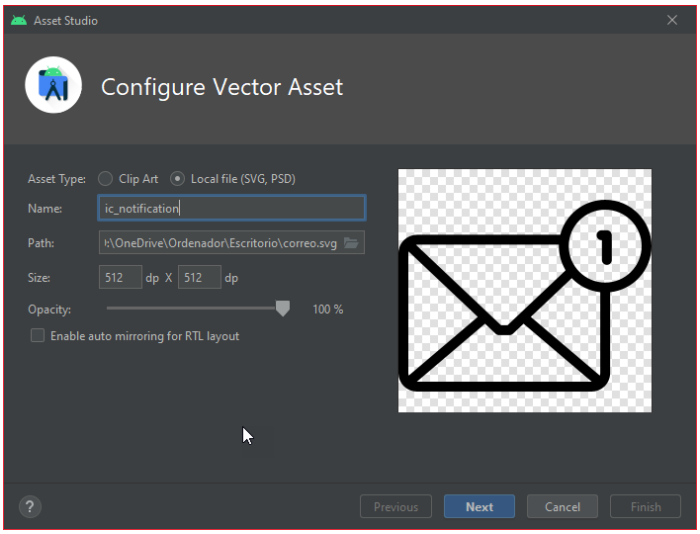


Ilustración 44. Ventana configuración Vector Asset

Una vez tengamos el icono en la biblioteca, ya podemos escribir el método que mostrará nuestra notificación

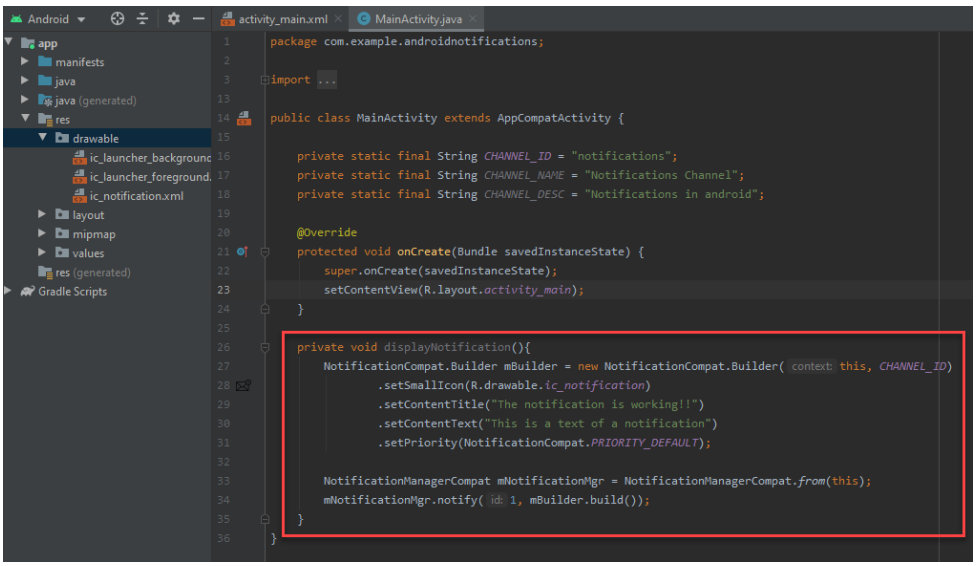


Ilustración 45. Código mostrar notificación

El siguiente paso es cambiar la vista (activity_main.xml), para ello vamos a simplificar la vista por defecto, para cambiar el *layout* de tipo: "androidx.constraintlayout.widget.ConstraintLayout" a RelativeLayout. Quitaremos el TextView actual y añadiremos un botón. Cuyo resultado final de nuestra vista es el siguiente:

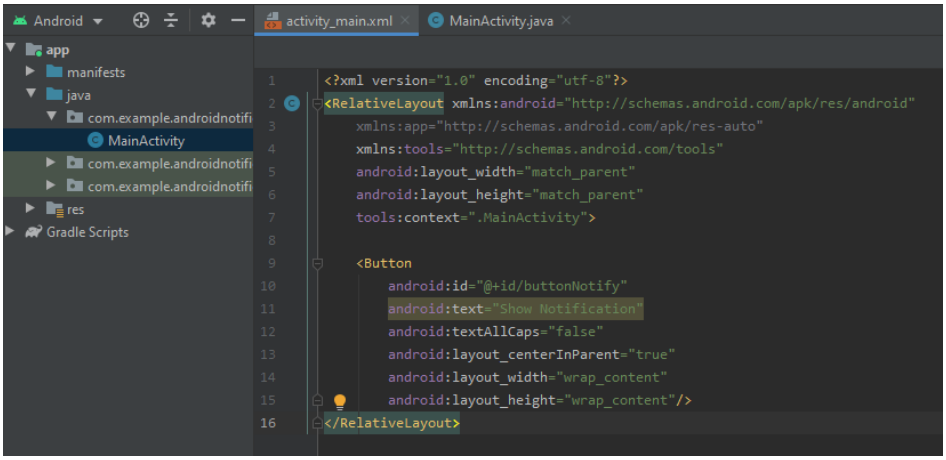


Ilustración 46. Código vista activity_main.xml

Seguidamente, volveremos a la parte de código y añadiremos, en el método "onCreate" del "MainActivity.java", la creación del canal de notificaciones y un *listener* de tipo "OnClickListener" para el botón que acabamos de definir.

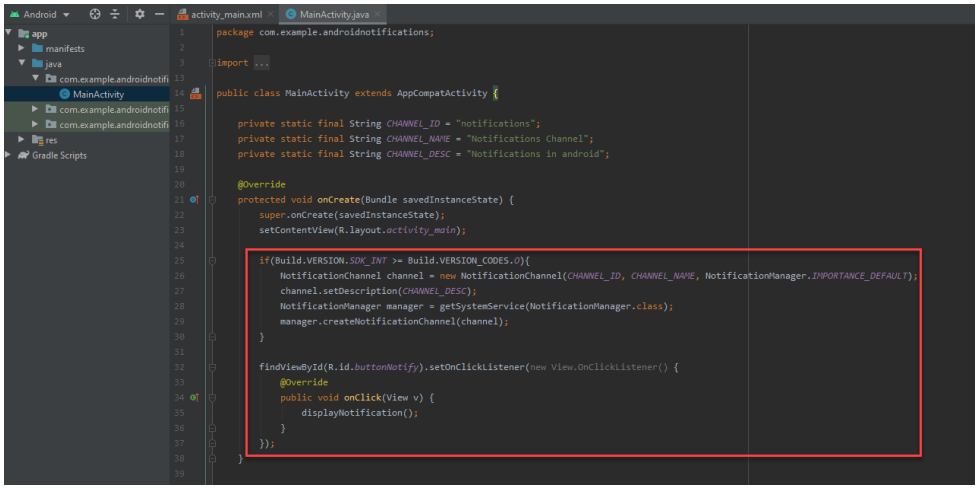


Ilustración 47. Código creación del canal y listener del nuevo botón

El siguiente paso será comprobar que funciona nuestra aplicación y se muestra la notificación que hemos creado. Para ello vamos a ejecutar la APP en el emulador que trae por defecto Android Studio

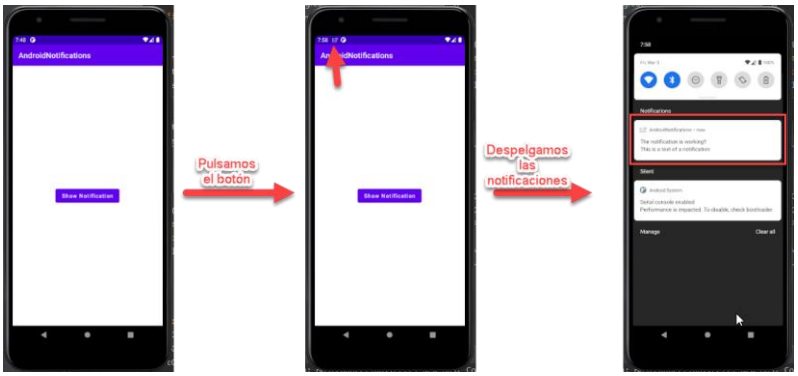


Ilustración 48. Ejemplo funcionamiento notificación en Smartphone

Ya podemos visualizar la primera notificación generada directamente desde la Aplicación Android. El siguiente paso será la realización de las configuraciones necesarias para poder recibir notificaciones de forma remota a través de Firebase Cloud Messaging.

Token Firebase Cloud Messaging

Para mandar notificaciones de forma remota, necesitamos saber el código único (Token) que Firebase le ha asignado a cada Smartphone. Dicho token es accesible desde la aplicación Android, es por ello, que cada vez que un dispositivo instale la aplicación vamos a consultar dicho Token y a guardarlo en la base de datos de Firebase

Primero prepararemos la interfaz, para ello modificaremos el “activity_main.xml” y lo dejaremos de la siguiente forma, quitando el botón y sustituyéndolo por un “TextView”

activity_main.xml | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewToken"
        android:textAllCaps="false"
        android:layout_centerInParent="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</RelativeLayout>
```

Ilustración 49. Código vista activity_main.xml

A continuación, eliminaremos la búsqueda del botón y su *listener*:

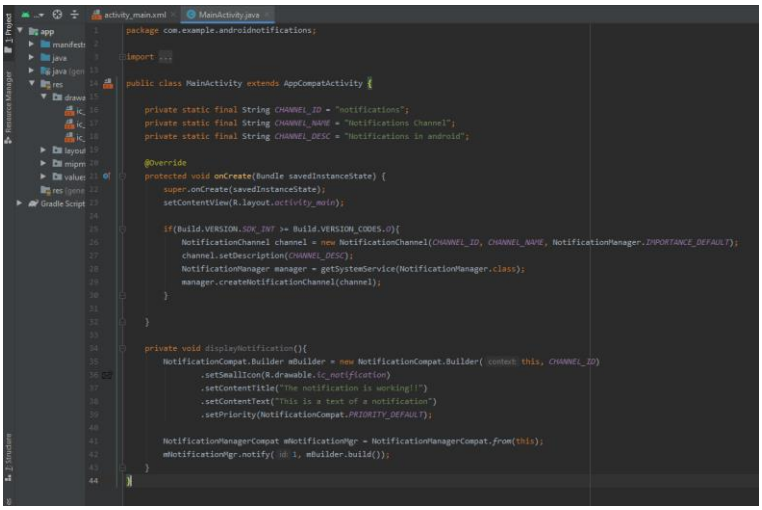


Ilustración 50. Código MainActivity.java

Una vez realizadas las modificaciones en la vista y en el código, obtendremos el token de registro de Firebase. Para ello, tendremos que registrar la aplicación en Firebase siguiendo los siguientes pasos:

Pinchamos en el menú Tools / Firebase

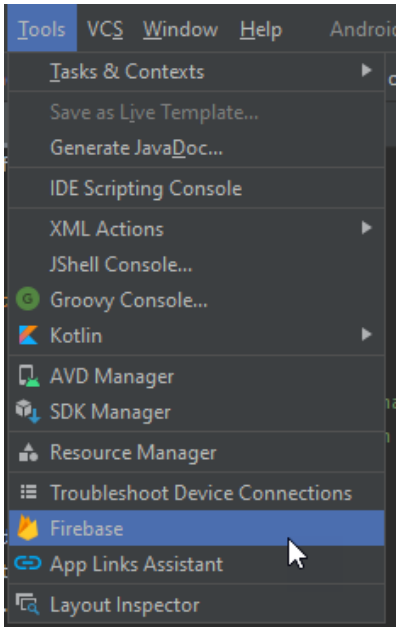


Ilustración 51. Menú Firebase

Y se nos desplegará otro menú a la derecha de la pantalla

NOTA: Antes de continuar, debemos de asegurarnos que hemos iniciado sesión en Android Studio con la misma cuenta que hemos creado el proyecto en Firebase.

En el nuevo menú de Firebase que ha aparecido a la derecha, buscaremos la opción Cloud Messaging y pincharemos en “Set up Firebase Cloud Messaging”

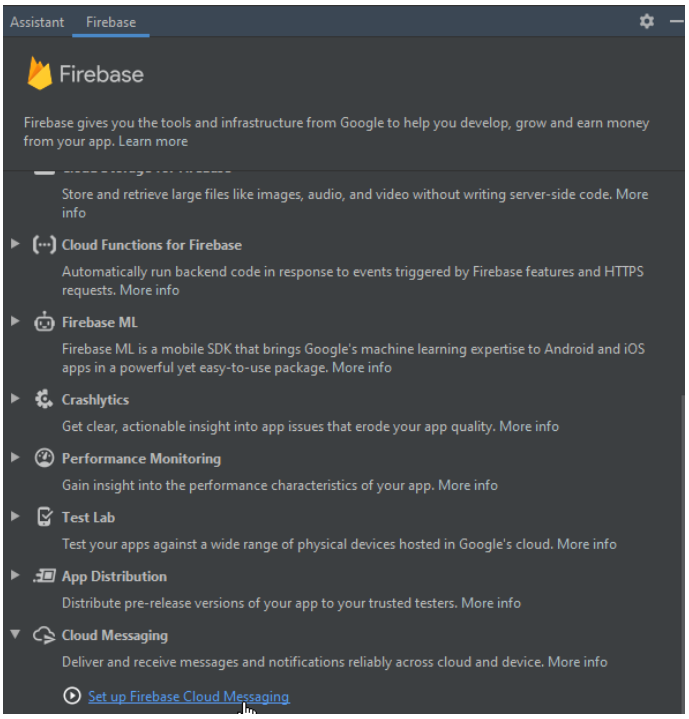


Ilustración 52. Menú Firebase Cloud Messaging

Seguidamente nos cambiará el menú y pulsaremos en “Connecto to Firebase”

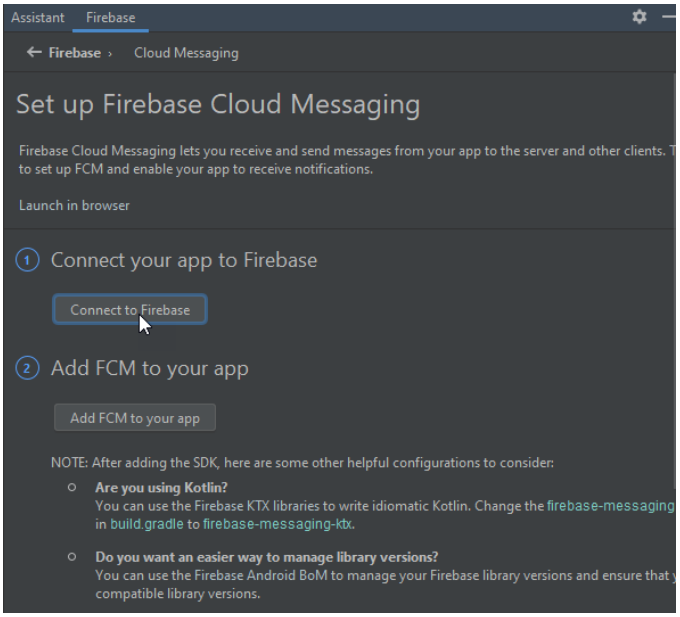


Ilustración 53. Conectar la aplicación a Firebase

Al pinchar en el botón, abrirá un navegador de internet en la dirección de Firebase. En dicha página, Firebase pregunta que proyecto queremos conectar, seleccionaremos el proyecto creado anteriormente (o tenemos la posibilidad de crear uno nuevo).

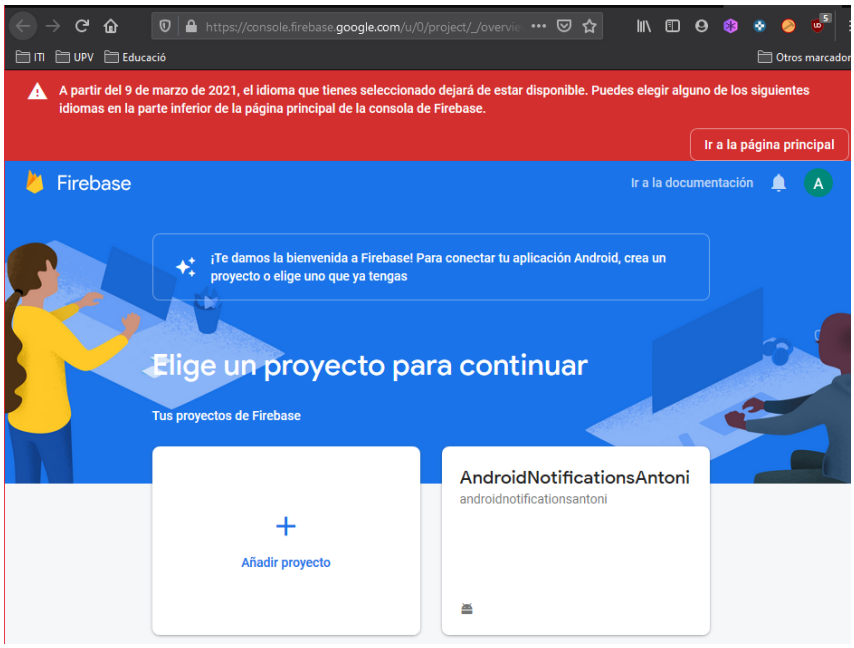


Ilustración 54. Selección de proyecto de Firebase

Una vez hemos seleccionado el proyecto, nos sale una ventana nueva con un botón de Conectar

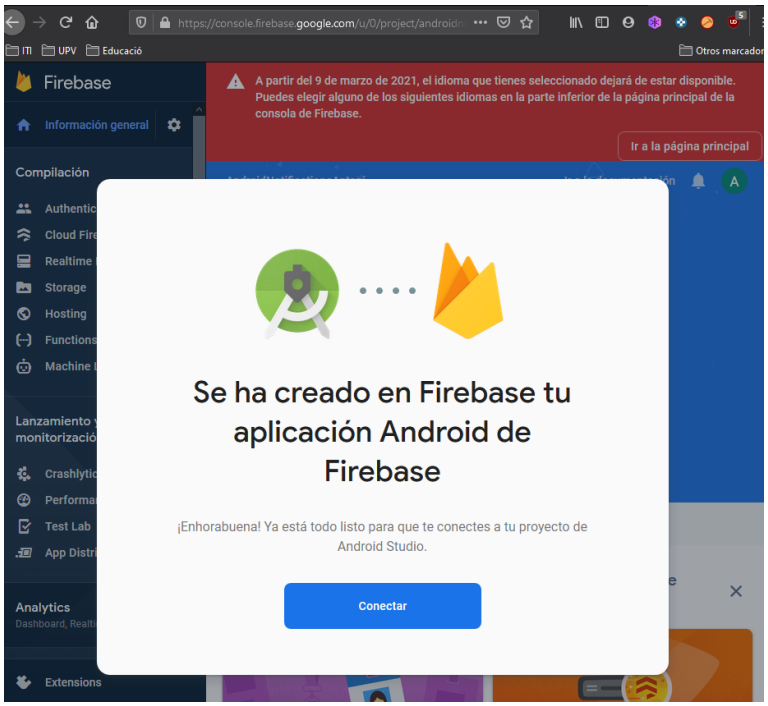


Ilustración 55. Ventana confirmación creación aplicación Android en Firebase

Al pulsar en conectar, nos saldrá un mensaje de que nuestra aplicación de Android se ha conectado correctamente a Firebase

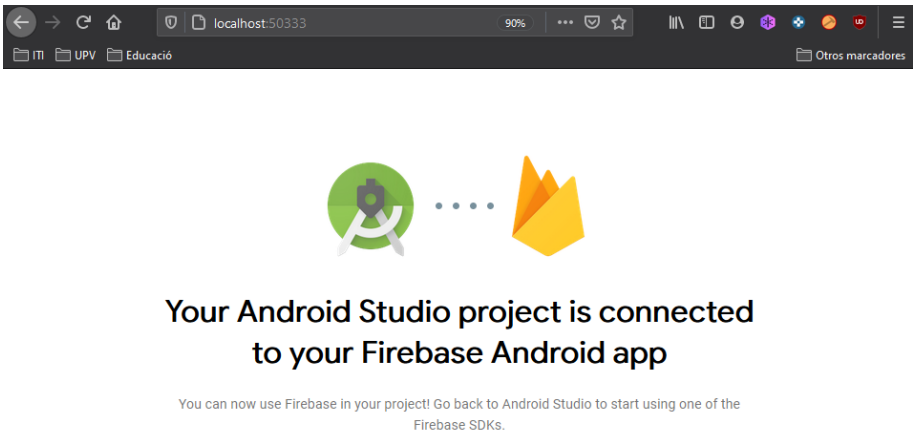


Ilustración 56. Ventana confirmación conexión Android Studio con Firebase

Ahora, en el menú de la derecha de Android Studio, observaremos que el punto 1 nos sale como conectado

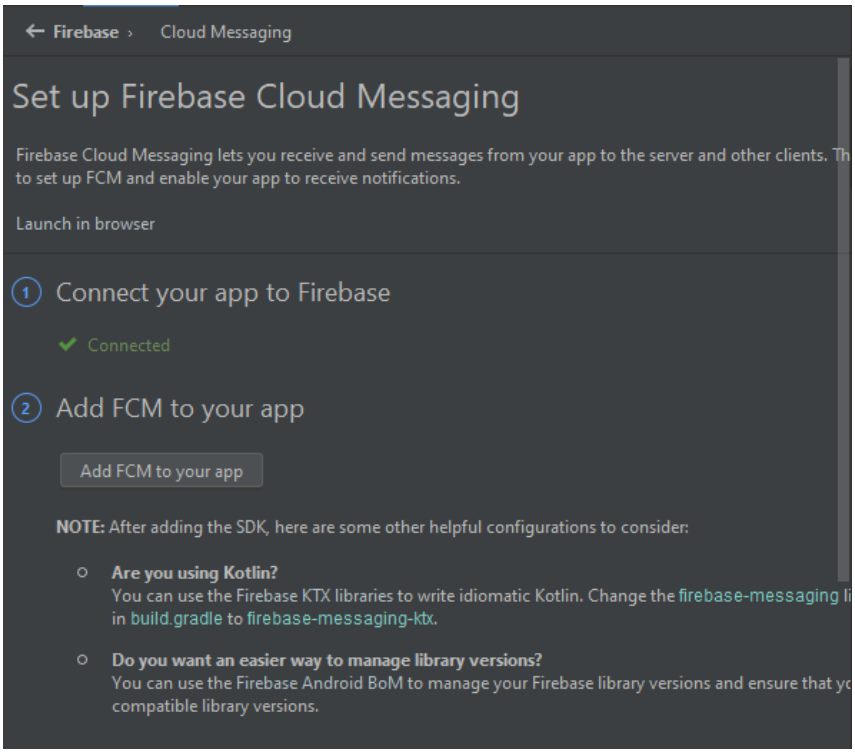


Ilustración 57. Ventana Firebase conectado correctamente

El siguiente paso es Añadir las dependencias de Firebase Cloud Messaging a nuestro proyecto, para ello, tenemos que pulsar el botón existente en el paso 2 “Add FCM to your app”

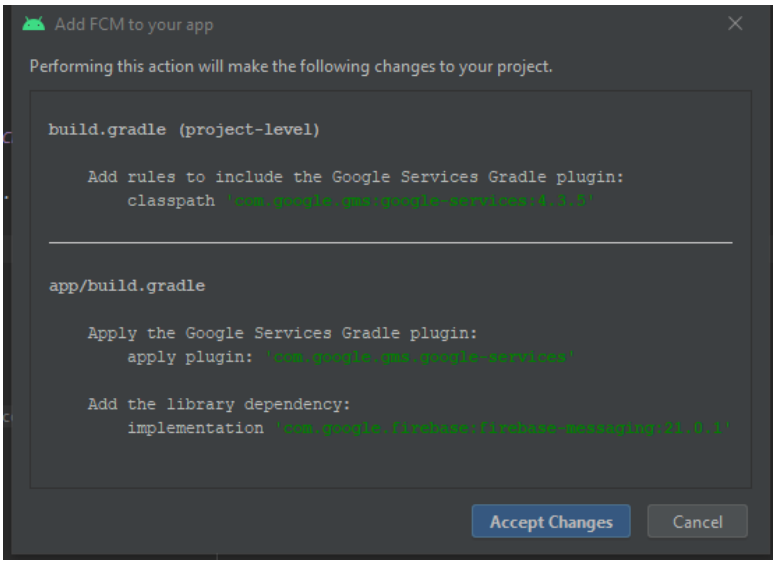


Ilustración 58. Ventana Add FCM to your app

Podemos observar una vista previa de las modificaciones que se van a producir en nuestra aplicación, añadir la regla y las dependencias necesarias para poder realizar la conexión entre ambos servicios. Aceptamos los cambios y ya tendremos la aplicación conectada con Firebase Cloud Messaging.

Una vez hemos conectado Firebase con nuestra aplicación. Vamos a obtener el token de registro y a guardarlo en la base de datos. Para ello, vamos a obtener el token de la siguiente forma:

primero tenemos que definir el “TextView” de la siguiente forma:

Ilustración 59. Añadir TextView Main Activity

Después editaremos el método “onCreate” para Recuperar la instancia de Firebase y mostrar el resultado en el “textView” que hemos añadido anteriormente:

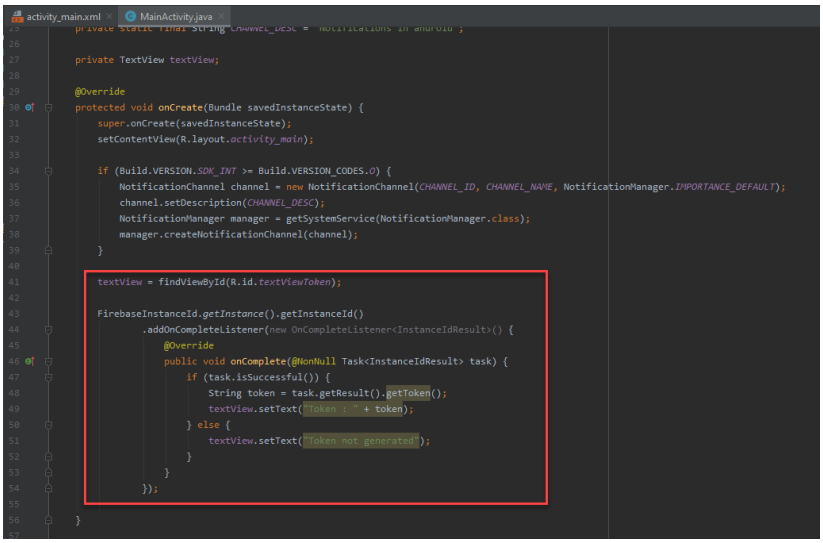


Ilustración 60. Código consulta Token FCM

Ahora vamos a poner en marcha la aplicación y observamos el resultado:

Ilustración 61. Visualización Ttoken FCM en pantalla

Como podemos observar, hemos recuperado el Token de Firebase Cloud Messaging y lo mostramos en pantalla. Dado que se trata de información interna y no le aporta información al usuario, vamos a realizar una serie de modificaciones para no mostrar dicho token.

Firebase Authentication

A continuación, vamos a implementar un sistema de autenticación en nuestra aplicación a través de los servicios de Firebase Authentication.

En primer lugar, eliminaremos el “TextView” que hemos creado anteriormente, ya que ahora no vamos a mostrar la información del token. Para realizarlo tenemos que eliminar las siguientes líneas de código:

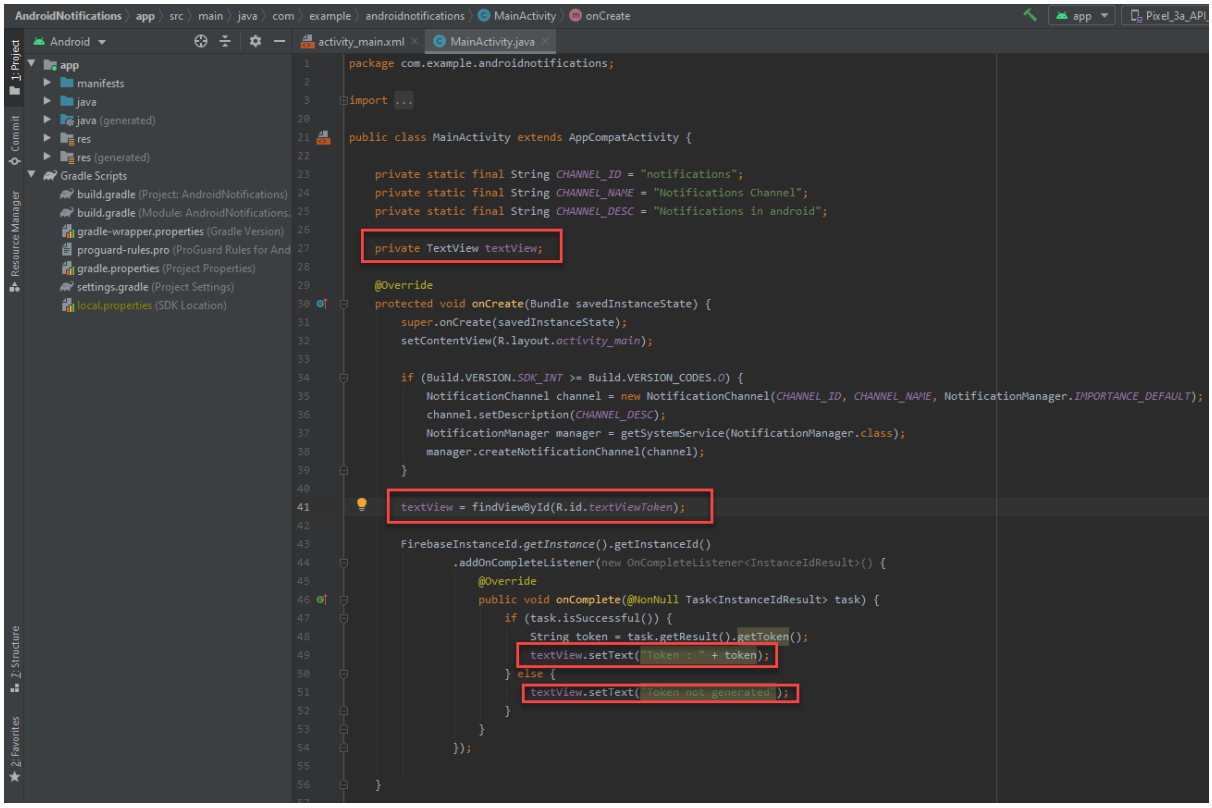


Ilustración 62. Código a eliminar MainActivity.java

Seguidamente, modificaremos el “activity_main.xml” para quitar el “textView” y diseñar una vista de la pantalla de inicio, es por ello, que ahora nuestro “activity_main.xml” va a tener el siguiente contenido

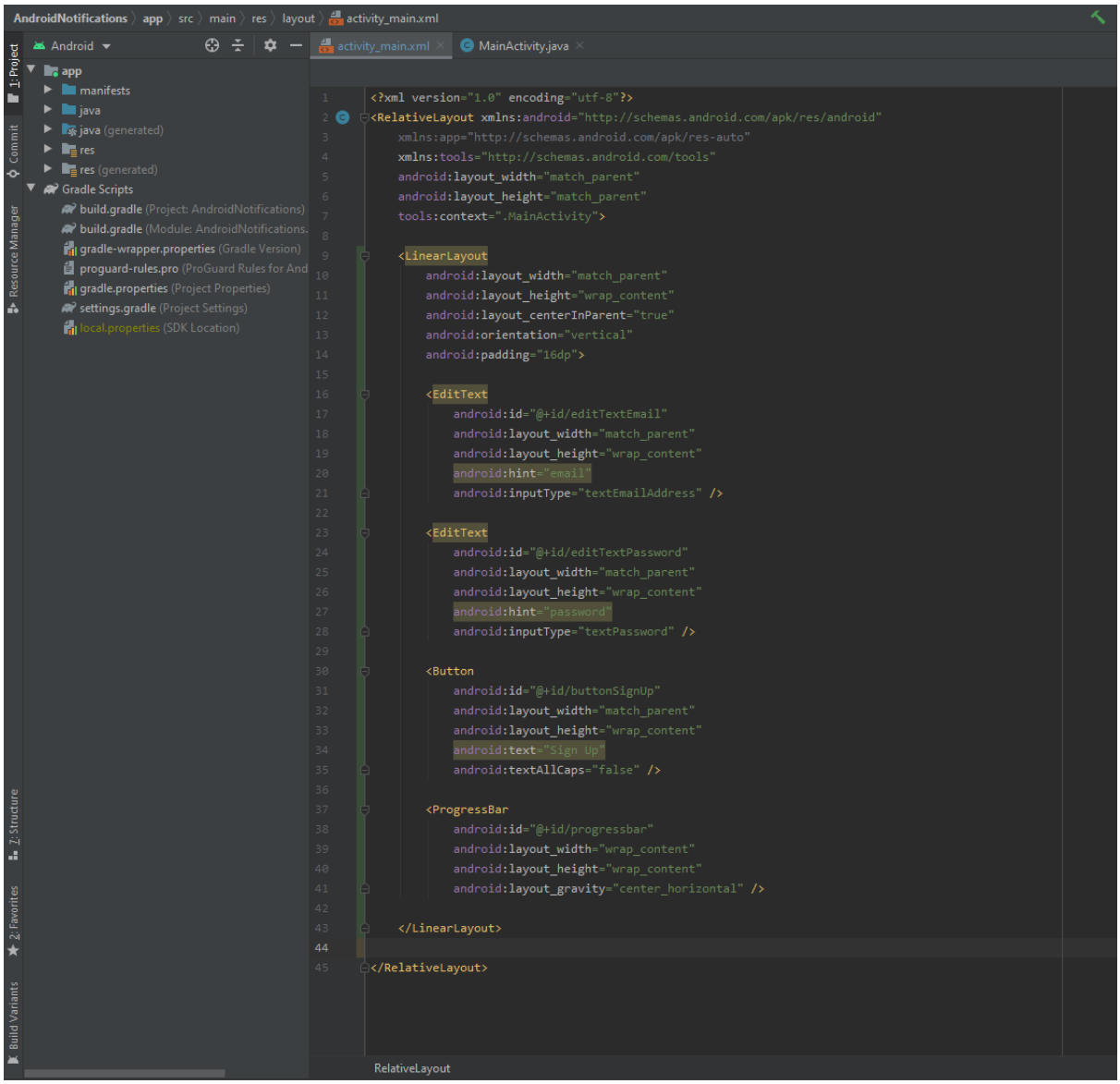


Ilustración 63. Nuevo contenido activity_main.xml

A continuación, declararemos e inicializare los siguientes nuevos componentes:

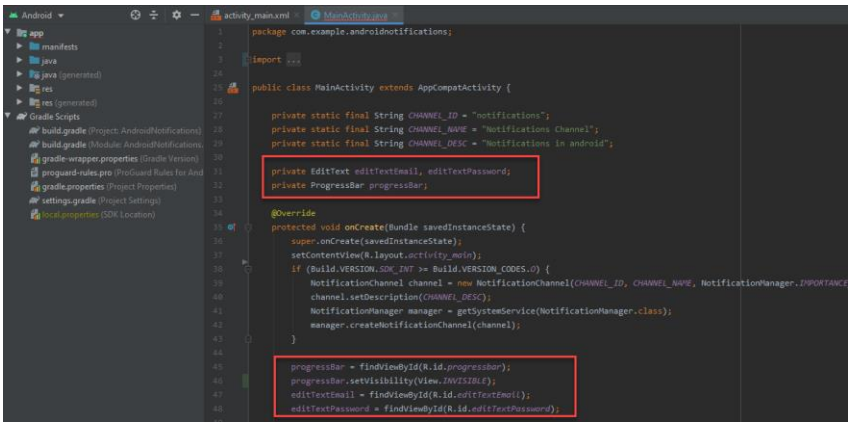


Ilustración 64. Declaración e inicialización nuevos componentes

El siguiente paso será añadir el evento “setOnClickListener” al botón de inicio de sesión:

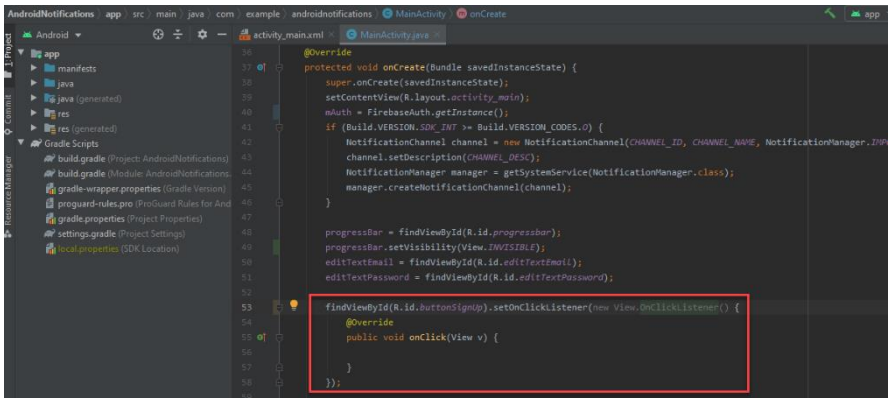


Ilustración 65. Listener botón login

Continuaremos añadiendo la autenticación por Firebase, volvemos al menú “Tools / firebase” y seleccionaremos “Authentication”

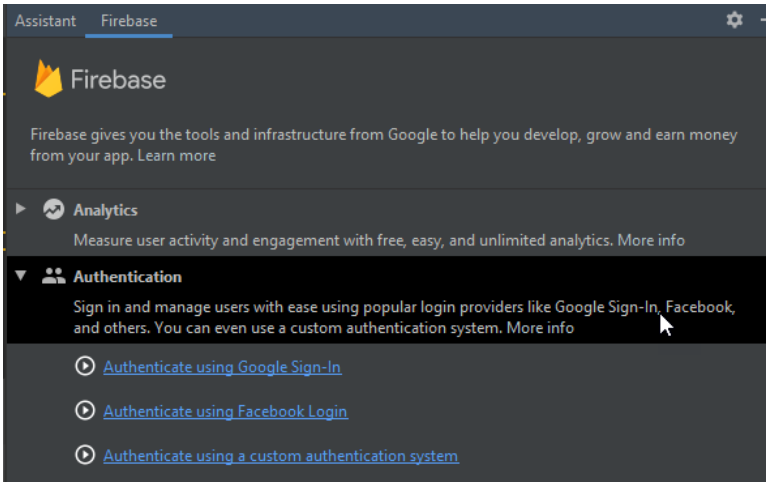


Ilustración 66. Firebase Authentication

En este punto, seleccionaremos la 3ª opción: "Authenticate using a custom authentication System". Una vez hecho esto, podremos observar cómo Firebase ya está conectado, puesto que lo hemos conectado anteriormente

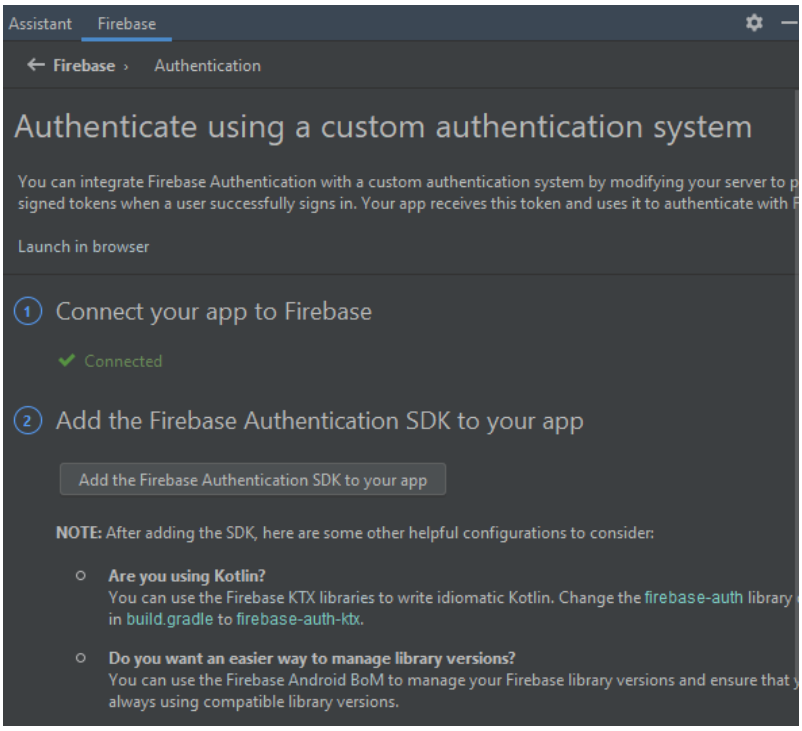


Ilustración 67. Ventana. Autenticación Firebase

Finalmente, tendremos que añadir el SDK a nuestro proyecto, pinchando en el botón del punto 2: "Add the firebase Authentication SDK to your app"

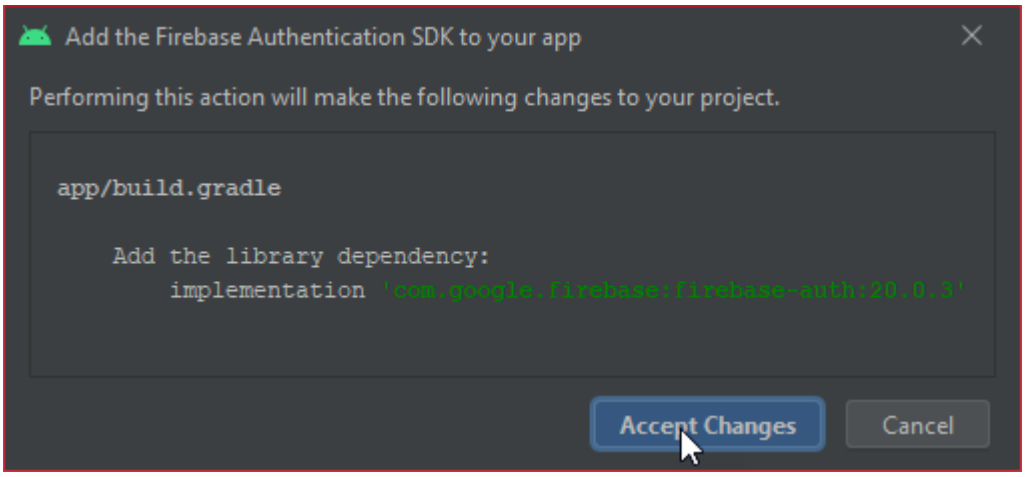


Ilustración 68. Cambios Firebase Atuhentication SDK

Aceptamos los cambios, y ya tendremos todo lo necesario.

Una vez hayamos conectado la autenticación de Firebase con el proyecto. Podemos usar dicha autenticación en nuestra aplicación.

Para poder autenticar correctamente contra Firebase, referenciaremos “FirebaseAuth”, declarando y asignando la instancia de ésta:

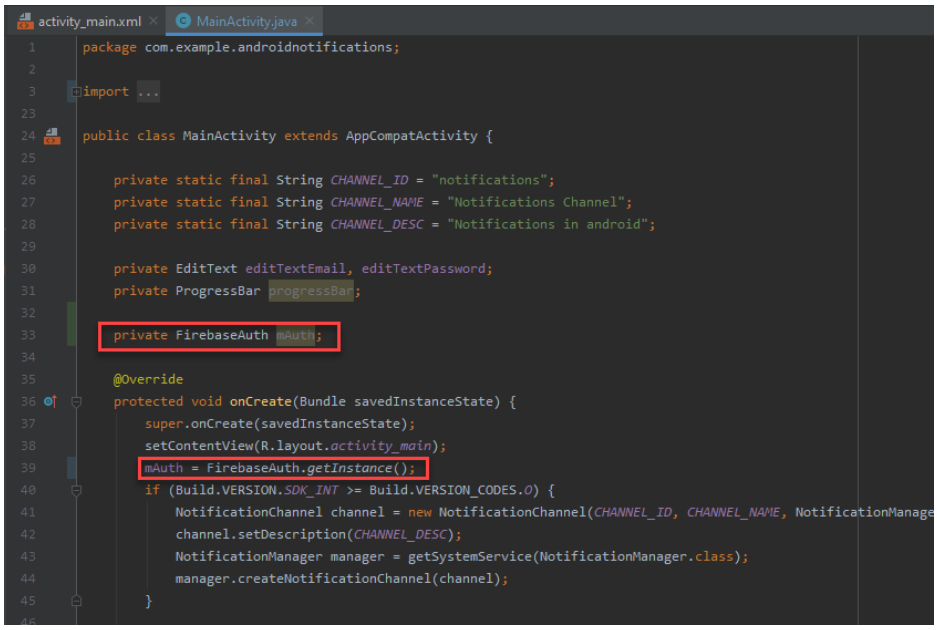


Ilustración 69. Declaración y asignación de FirebaseAuth

Dado que vamos a implementar una autenticación, nuestra aplicación precisa de 2 ventanas, una para el inicio de sesión, y otra para la actividad principal de la aplicación. Crearemos una nueva actividad:

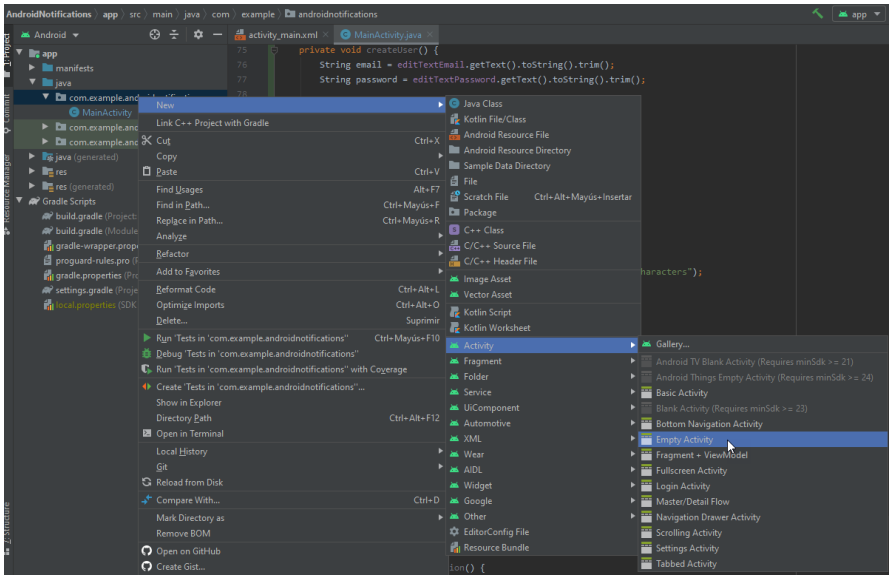


Ilustración 70. Menú creación nueva actividad

El nombre que le daremos será “ProfileActivity”, y las otras opciones las dejaremos tal y como sigue la siguiente captura:

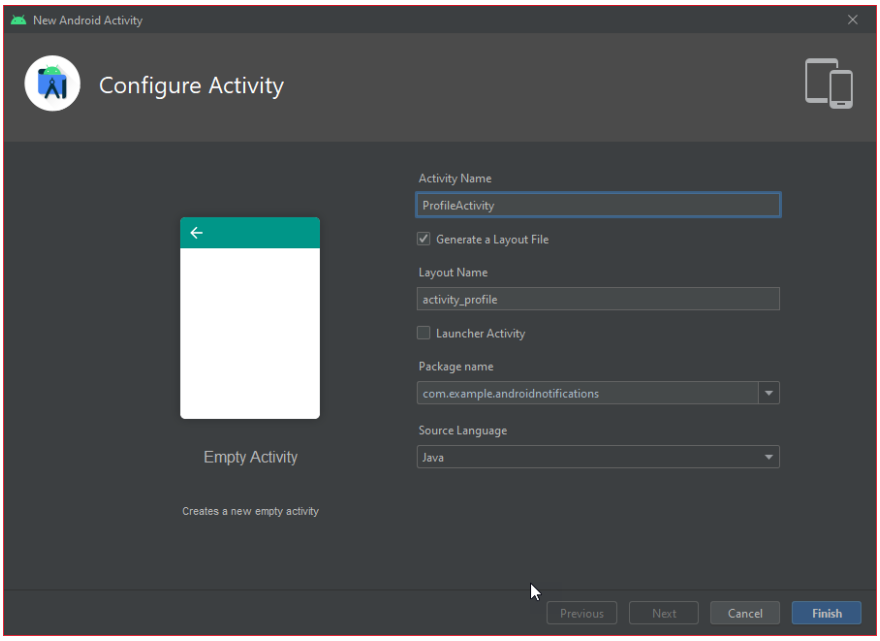


Ilustración 71. Configuración nueva actividad. ProfileActivity

A continuación, crearemos un método para iniciar dicha actividad desde la ventana “MainActivity”:

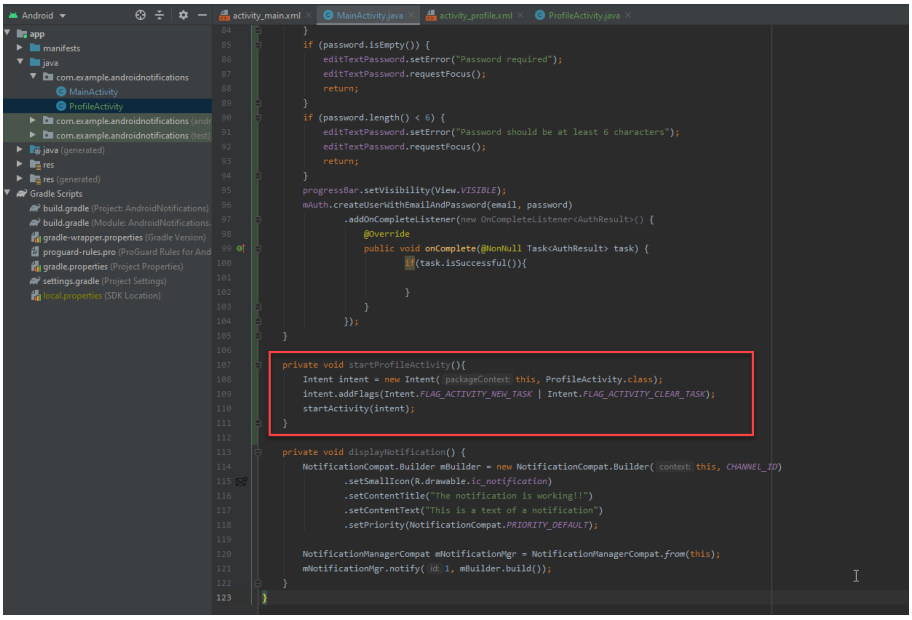
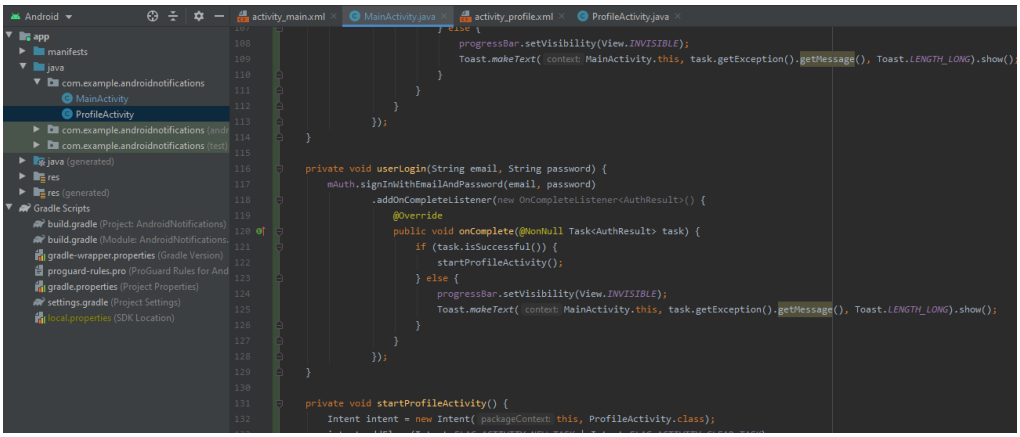


Ilustración 72. Método iniciar nueva actividad

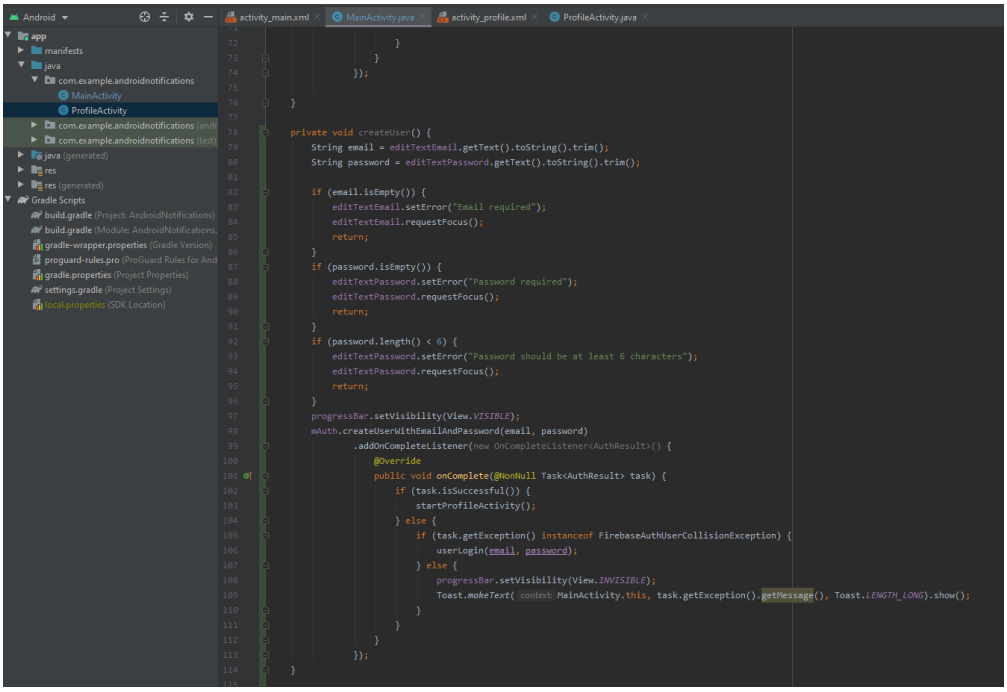
Seguidamente, crearemos el método encargado de la autenticación de los usuarios:



```
private void userLogin(String email, String password) {
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        startProfileActivity();
                    } else {
                        progressBar.setVisibility(View.INVISIBLE);
                        Toast.makeText( context: MainActivity.this, task.getException().getMessage(), Toast.LENGTH_LONG).show();
                    }
                }
            });
}

private void startProfileActivity() {
    Intent intent = new Intent( packageContext: this, ProfileActivity.class);
```

Ilustración 73. Método autenticación usuarios

Ahora crearemos el método encargado de la creación de nuevos usuarios. En dicho método, realizaremos comprobaciones varias sobre los campos, tales como, campos no vacíos, contraseña mayor de 6 caracteres…:



```
private void createUser() {
    String email = editTextEmail.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();

    if (email.isEmpty()) {
        editTextEmail.setError("Email required");
        editTextEmail.requestFocus();
        return;
    }
    if (password.isEmpty()) {
        editTextPassword.setError("Password required");
        editTextPassword.requestFocus();
        return;
    }
    if (password.length() < 6) {
        editTextPassword.setError("Password should be at least 6 characters");
        editTextPassword.requestFocus();
        return;
    }
    progressBar.setVisibility(View.VISIBLE);
    mAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        startProfileActivity();
                    } else {
                        if (task.getException() instanceof FirebaseAuthUserCollisionException) {
                            userLogin(email, password);
                        } else {
                            progressBar.setVisibility(View.INVISIBLE);
                            Toast.makeText( context: MainActivity.this, task.getException().getMessage(), Toast.LENGTH_LONG).show();
                        }
                    }
                }
            });
}
```

Ilustración 74. Método creación de usuarios

En el *listener* del botón que hemos añadido, añadiremos la llamada al nuevo método:

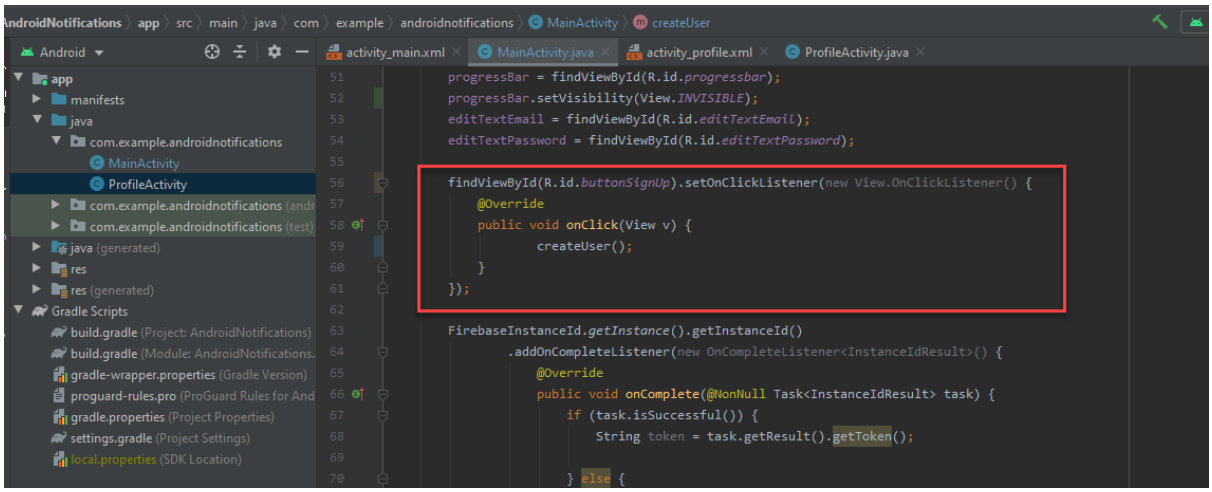


Ilustración 75. Llama createUser en el listener

Antes de proceder a testear los cambios, tenemos permitir la autenticación por usuario y contraseña de Firebase. Para ello, iremos a la consola y entraremos en el menú "Authentication".

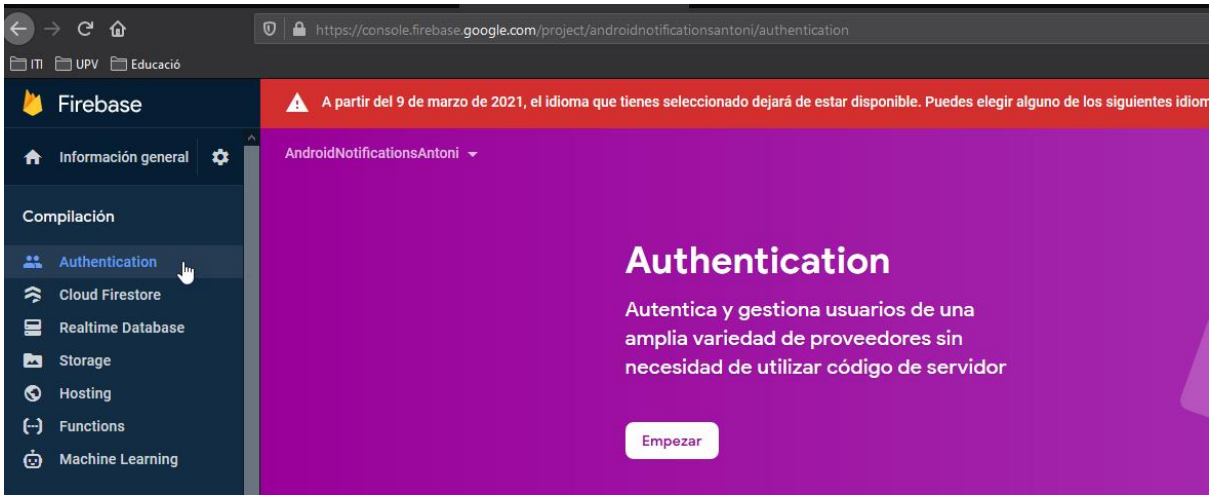


Ilustración 76. Opción Authentication Consola Firebase

Y en la pestaña “Sign-in method”, habilitaremos la opción de correo electrónico/contraseña:

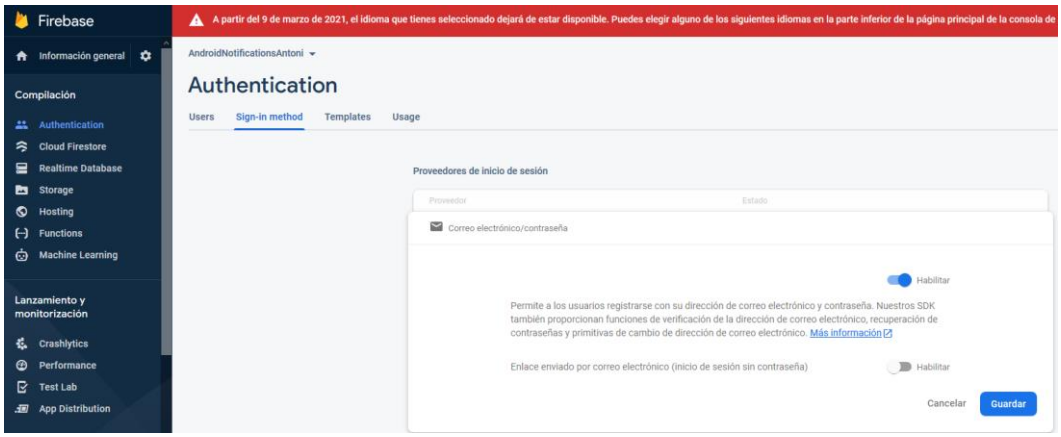


Ilustración 77. Habilitar inicio de sesión por correo electrónico y contraseña Firebase

Una vez realizada dicha configuración, procederemos a testear la aplicación:

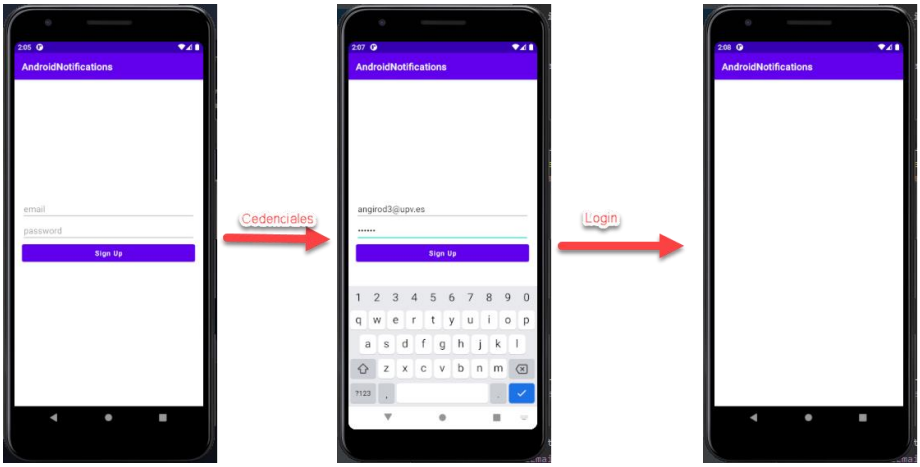


Ilustración 78. Flujo inicio sesión aplicación Android

Para comprobar que el usuario se ha creado correctamente, accederemos a la pestaña “Users” de Firebase:

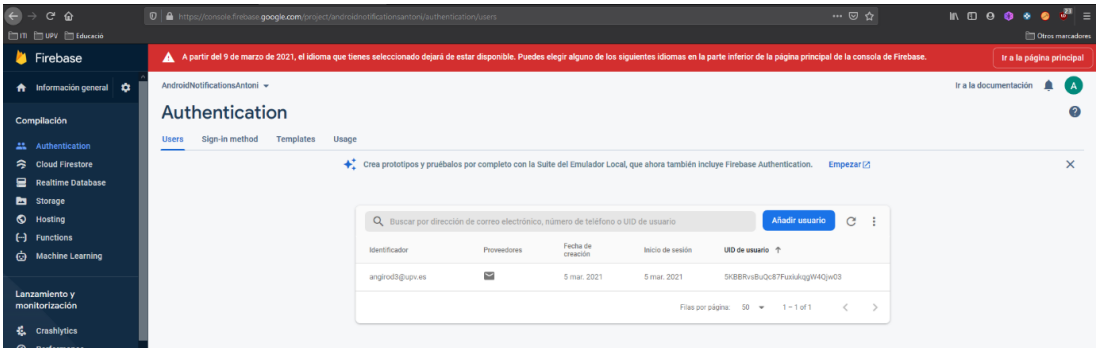


Ilustración 79. Tabla usuarios Firebase con el nuevo usuario

Guardado del Token

Una vez seamos capaces de obtener el token de Firebase, y autenticar el usuario por correo electrónico y contraseña, vamos a guardar dicho token en la base de datos de Firebase:

En primer lugar, tendremos que crear una base de datos de tipo “Realtime” (si no la teníamos creado con anterioridad) accediendo a la pestaña “Realtime Database” de la consola de Firebase:

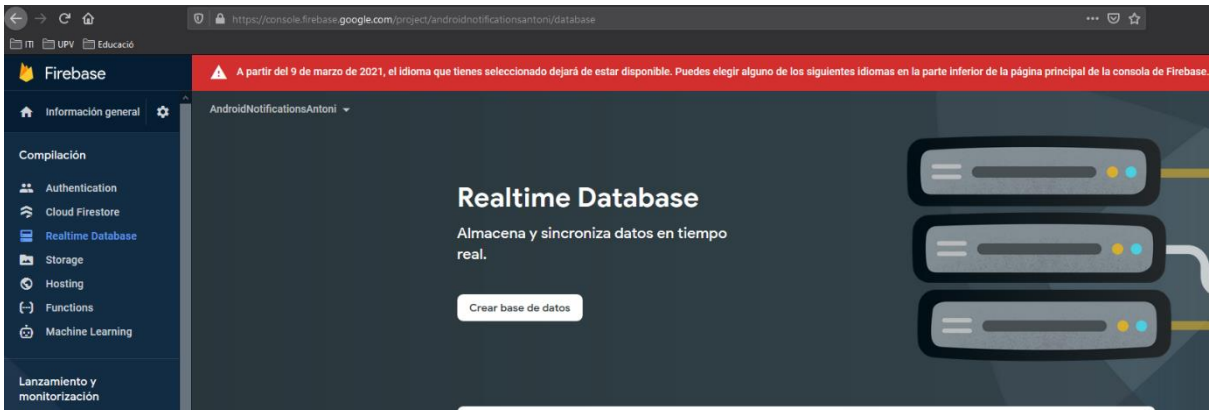


Ilustración 80. Pestaña Realtime Database de Firebase

Y crearemos una base de datos nueva en modo de prueba

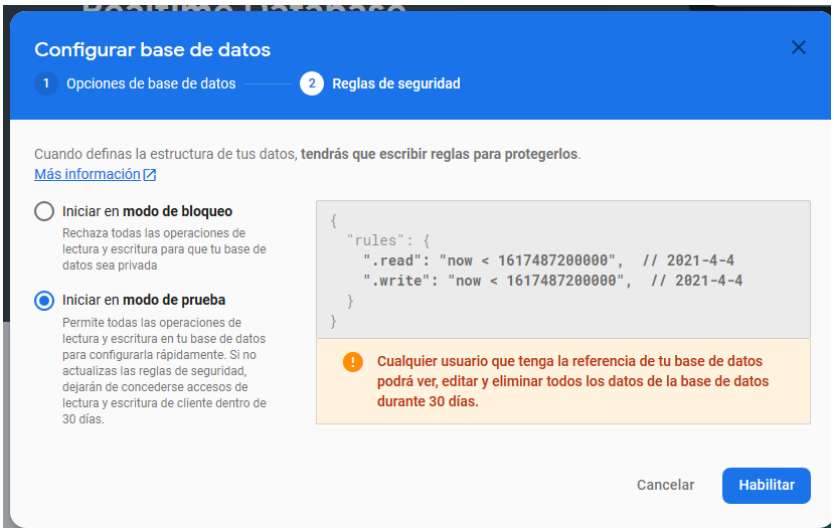


Ilustración 81. Creación Realtime Database en modo prueba

Una vez esté creada la base de datos, vamos a cambiar las reglas para que solo puedan leer y escribir los usuarios autenticados y validados en nuestro proyecto, para ello, accedemos a la pestaña Rules, editamos las reglas dejando el código JSON que se muestra a continuación y pulsamos en publicar.

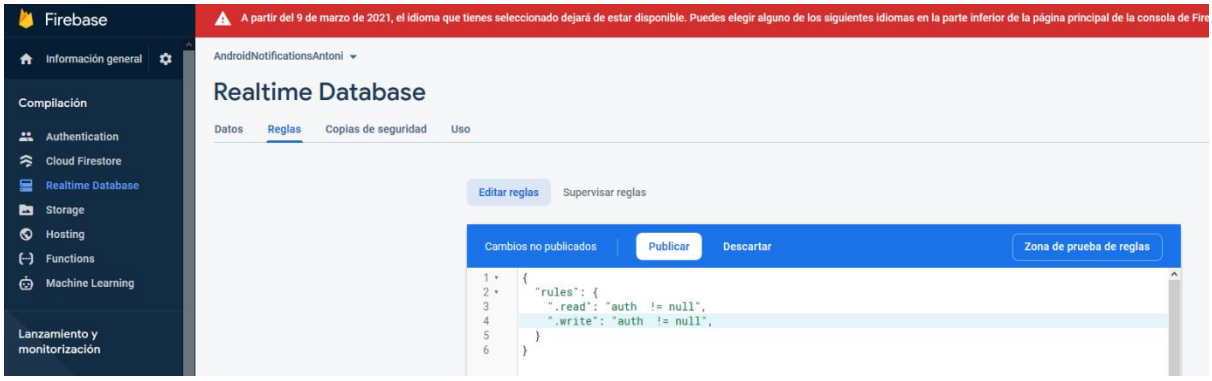


Ilustración 82. Reglas Firebase Realtime Database

Una vez tengamos las reglas creadas, volveremos al proyecto de Android Studio y crearemos una nueva clase en el proyecto llamada "User":

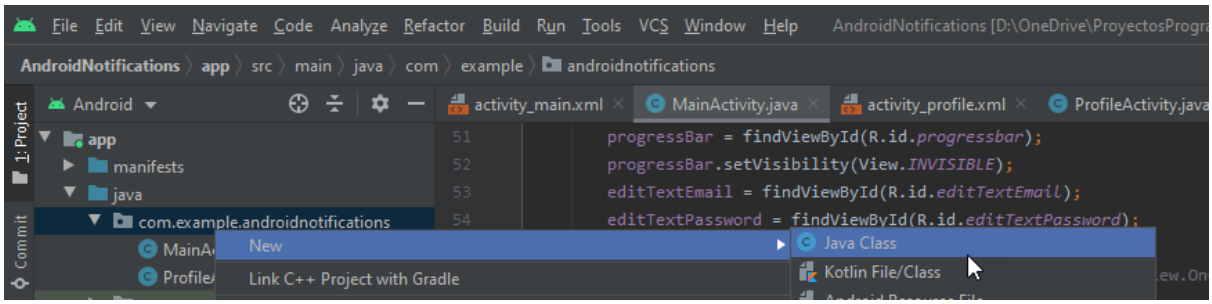


Ilustración 83. Creación nueva clase Android

En esta clase vamos a poner las 2 propiedades que necesitamos para identificar el dispositivo, email y token:

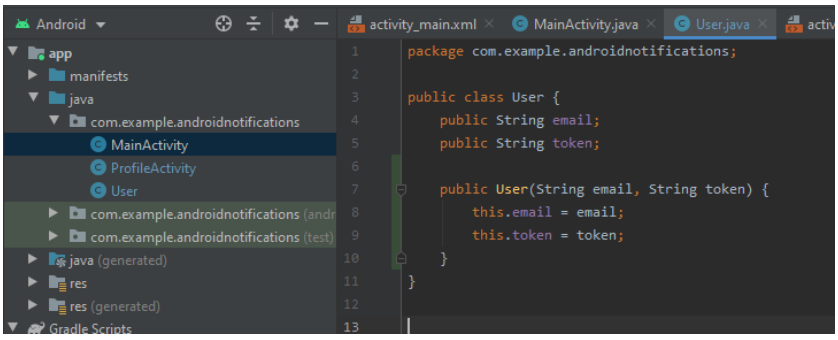


Ilustración 84. Contenido clase User.java

Una vez tengamos la clase “User” creada, cortaremos el código del “MainActivity” donde obteníamos anteriormente el código y pegaremos en el “profileActivity”:

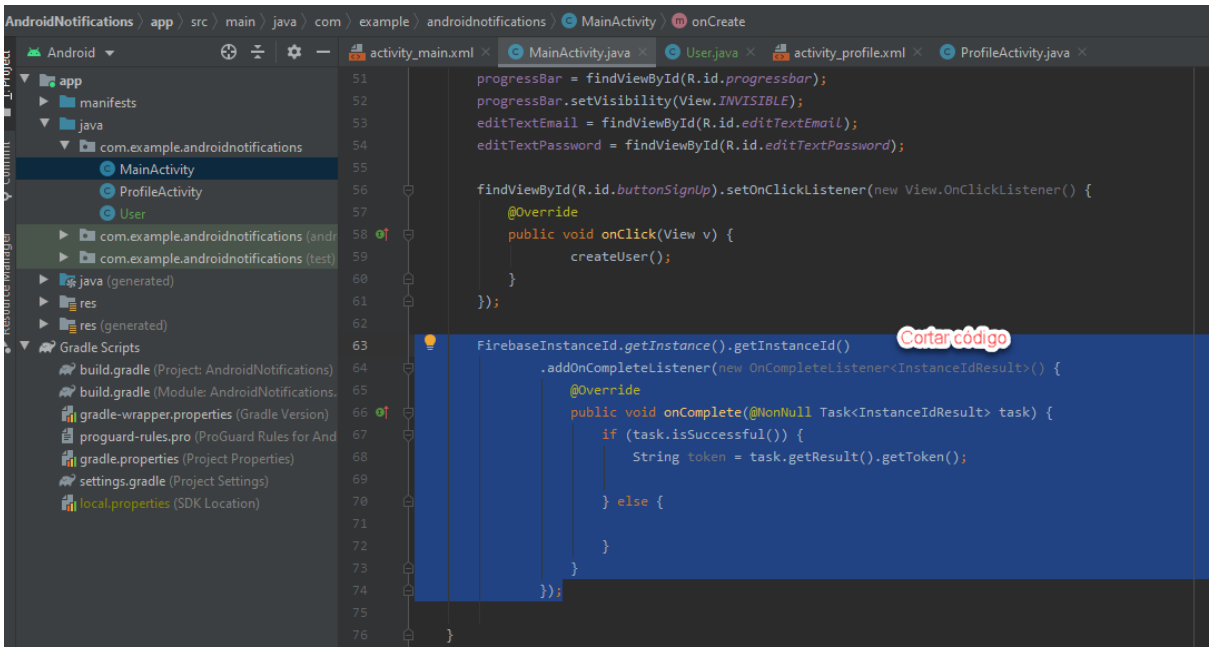


Ilustración 85. Código a cortar MainActivity.java

Y lo pegamos en el “onCreate” del “ProfileActivity”:

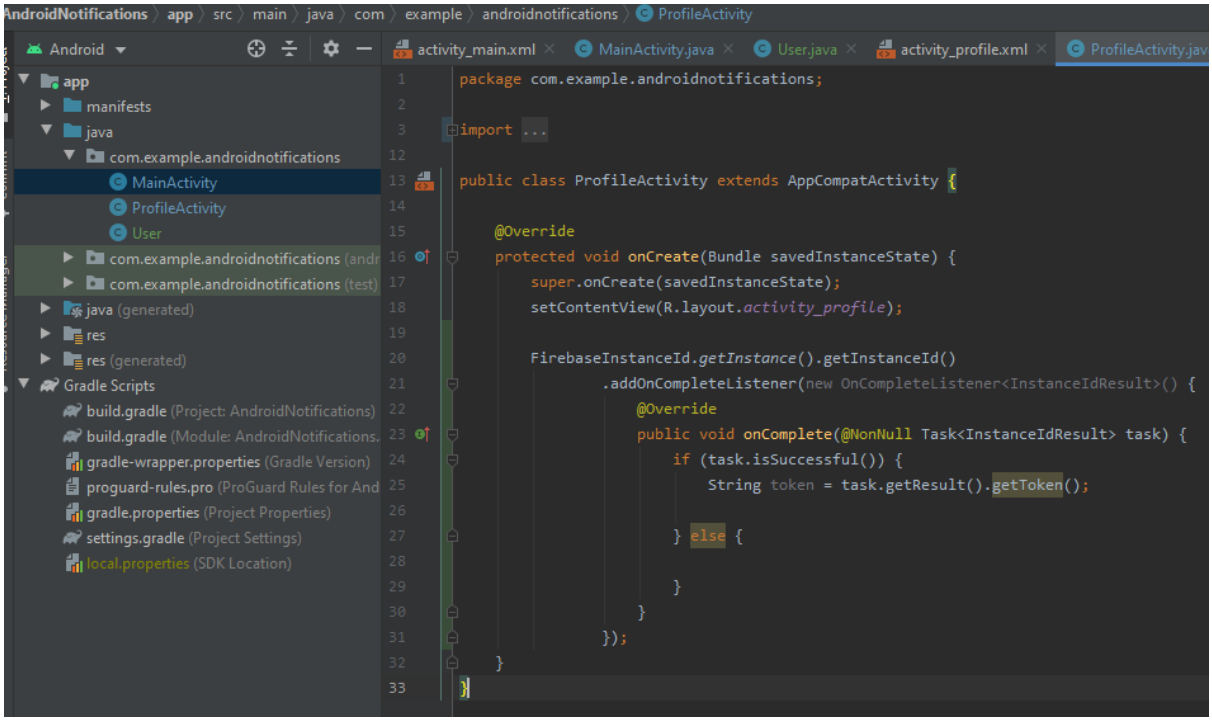


Ilustración 86. Código pegado ProfileActivity.java

El siguiente paso será almacenar este token en la base de datos, para ello, primero tenemos que añadir la referencia de la base de datos *Realtime* a nuestro proyecto.

Volvemos a la opción Tools del menú, Firebase, Realtime Database y pulsamos el enlace:

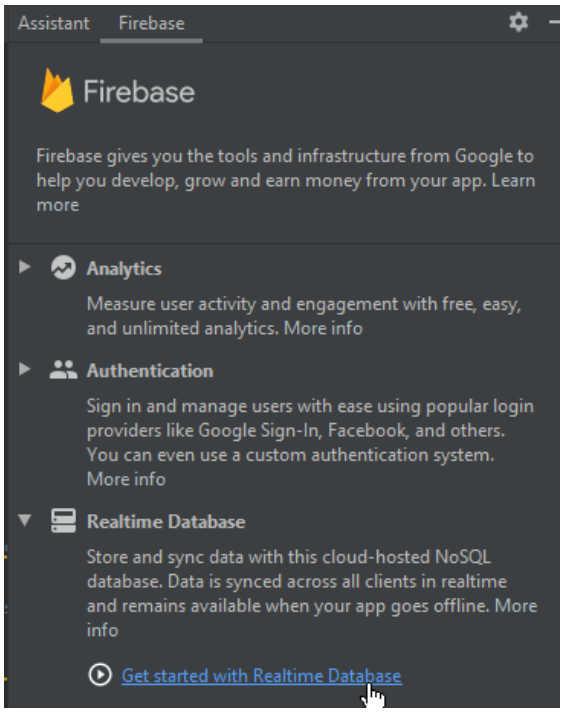


Ilustración 87. Menú añadir Firebase Realtime Database a Android

Ya hemos conectado anteriormente la App con Firebase, con lo que solo tenemos que añadir el SDK a nuestro proyecto:

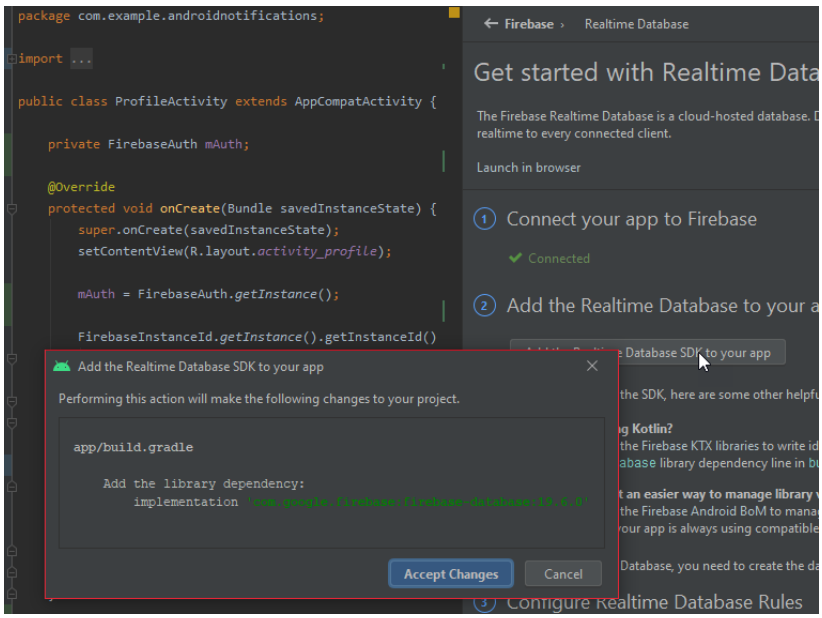


Ilustración 88. Añadir dependencias Firebase Realtime Database al proyecto

Seguidamente, crearemos un método que realice dicha función:

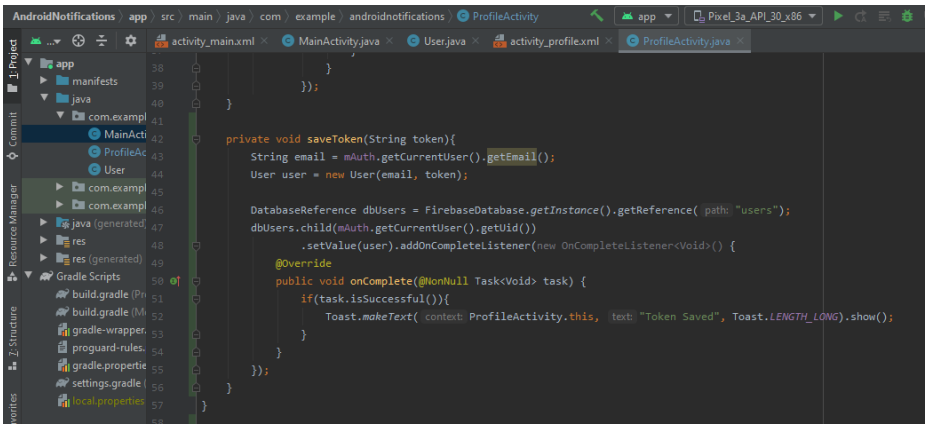


Ilustración 89. Método saveToken

Y lo referenciaremos en el *listener* definido en el método "onCreate" de dicha actividad:

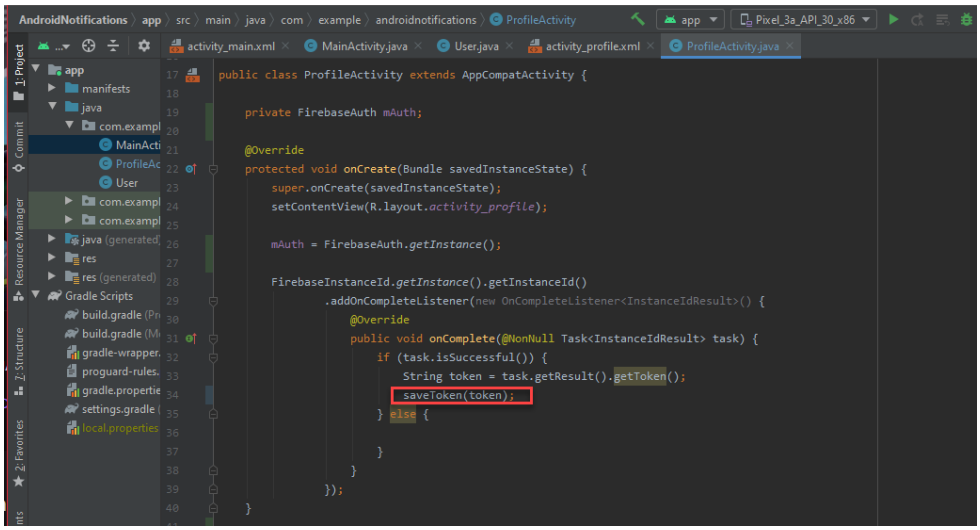


Ilustración 90. Referencia saveToken

Sobrescribimos el método "onStart" del "ProfileActivity" para evitar que alguien pueda acceder a esta ventana sin autenticarse:

Ilustración 91. Comprobación usuario autenticado

Del mismo modo, sobrescribimos el método “onStart” del “MainActivity” para que, si estamos autenticados, nos redirija automáticamente a “profileActivity”:

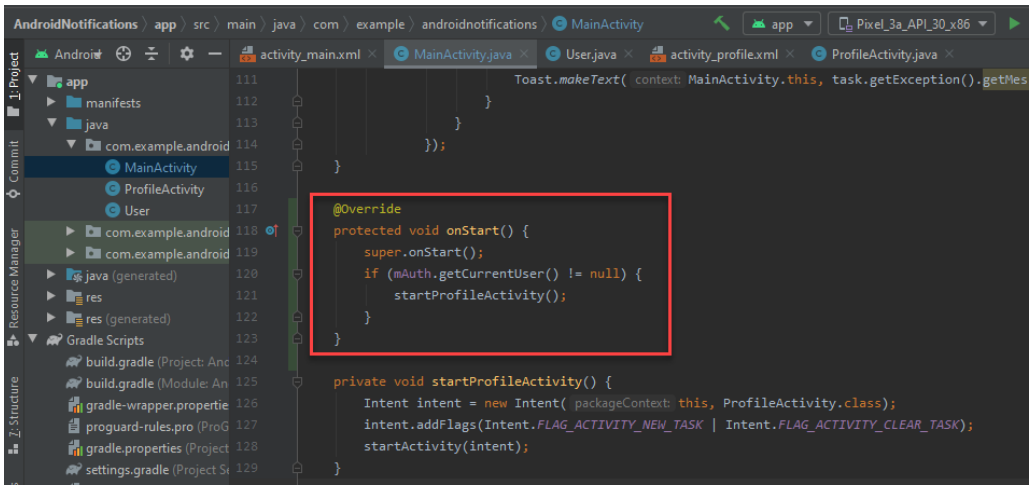


Ilustración 92. Redirigir a profileActivity

Ejecutamos la aplicación y observamos que todo es correcto:

Ilustración 93. Token guardado APP Smartphone

Y en nuestra base de datos de Firebase, observaremos que se ha guardado correctamente dicho token:

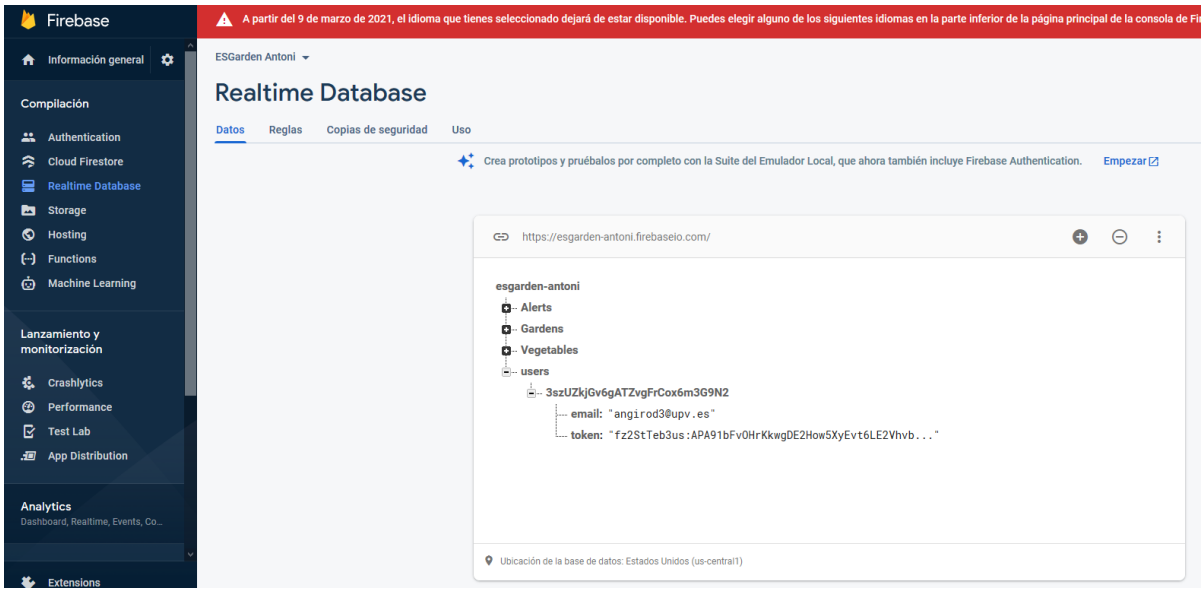
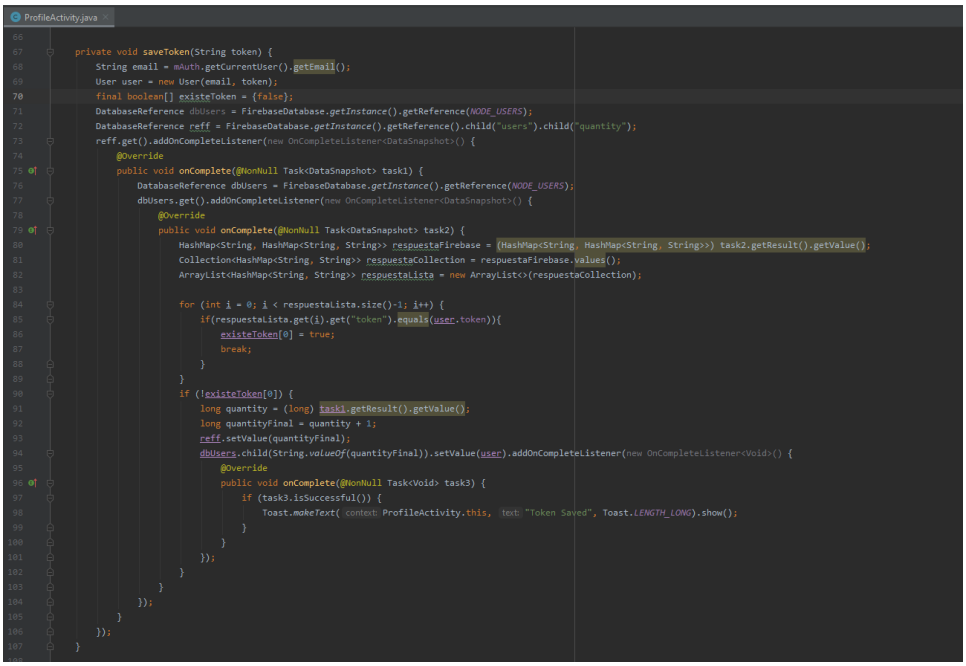


Ilustración 94. Captura del token guardado en la base de datos de Firebase

Dado que en la parte de Arduino tenemos que recuperar la lista de tokens para mandar las notificaciones *push* a los dispositivos que usan nuestra APP. Vamos a modificar el método "saveToken" de la clase "ProfileActivity.java" para que nos genere una estructura como la que observamos a continuación:

Ilustración 95. Estructura definitiva colección users

El método “saveToken” quedará de la siguiente forma, para cumplir con la nueva estructura:



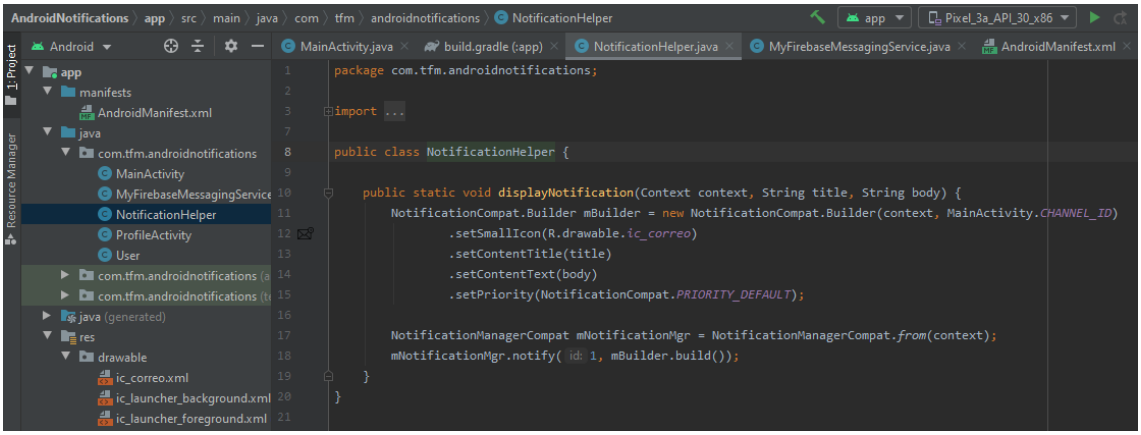
```
private void saveToken(String token) {
    String email = mAuth.getCurrentUser().getEmail();
    User user = new User(email, token);
    final boolean[] existeToken = {false};
    DatabaseReference dbUsers = FirebaseDatabase.getInstance().getReference(NODE_USERS);
    DatabaseReference reff = FirebaseDatabase.getInstance().getReference().child("users").child("quantity");
    reff.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DataSnapshot> task1) {
            DatabaseReference dbUsers = FirebaseDatabase.getInstance().getReference(NODE_USERS);
            dbUsers.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DataSnapshot> task2) {
                    HashMap<String, HashMap<String, String>> respuestaFirebase = (HashMap<String, HashMap<String, String>>) task2.getResult().getValue();
                    Collection<HashMap<String, String>> respuestaCollection = respuestaFirebase.values();
                    ArrayList<HashMap<String, String>> respuestaLista = new ArrayList<>(respuestaCollection);

                    for (int i = 0; i < respuestaLista.size()-1; i++) {
                        if(respuestaLista.get(i).get("token").equals(user.token)){
                            existeToken[0] = true;
                            break;
                        }
                    }
                    if (!existeToken[0]) {
                        long quantity = (long) task1.getResult().getValue();
                        long quantityFinal = quantity + 1;
                        reff.setValue(quantityFinal);
                        dbUsers.child(String.valueOf(quantityFinal)).setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task3) {
                                if (task3.isSuccessful()) {
                                    Toast.makeText( context: ProfileActivity.this,  text: "Token Saved", Toast.LENGTH_LONG).show();
                                }
                            }
                        });
                    }
                }
            });
        }
    });
}
```

Ilustración 96. Modificación método saveToken

Notificaciones *push* desde Firebase

Para lanzar notificaciones desde Firebase y que éstas se visualicen en los terminales Android que contengan la aplicación, debemos realizar una serie de modificaciones:

En primer lugar, creamos una nueva clase llamada “NotificactionHelper”. Y dentro de esta clase vamos a cortar el código del método “displayNotification” que inicialmente creamos en “MainActivity” y haciendo publica la constante “CHANNEL_ID” del “mainActivity”



```
package com.tfm.androidnotifications;

import ...

public class NotificationHelper {

    public static void displayNotification(Context context, String title, String body) {
        NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(context, MainActivity.CHANNEL_ID)
                .setSmallIcon(R.drawable.ic_correo)
                .setContentTitle(title)
                .setContentText(body)
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        NotificationManagerCompat mNotificationMgr = NotificationManagerCompat.from(context);
        mNotificationMgr.notify( id: 1, mBuilder.build());
    }
}
```

Ilustración 97. Contenido clase NotificationHelper

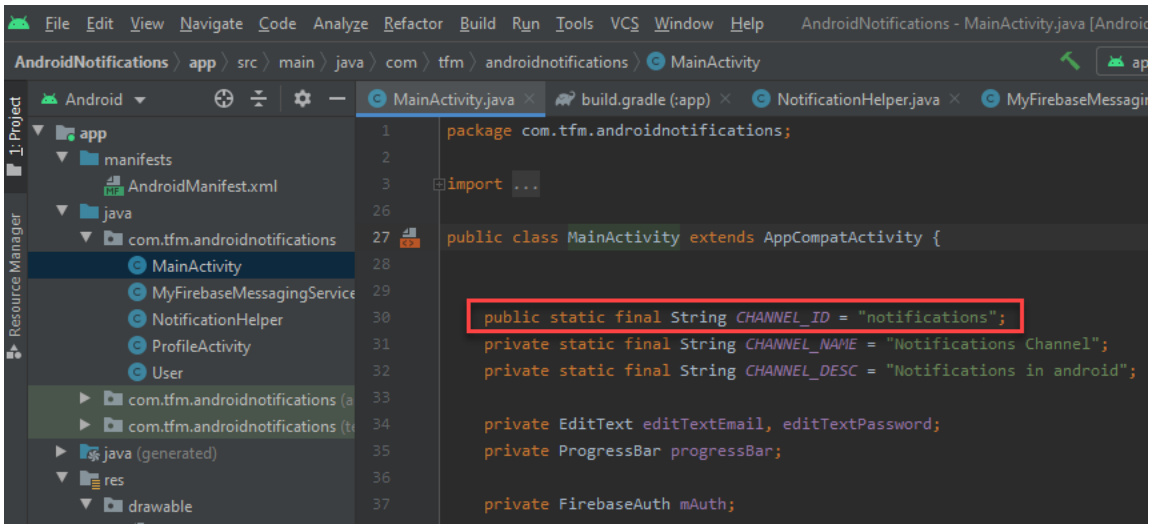


Ilustración 98. Variable CHANNEL_ID pública

Seguidamente, vamos a crear una nueva clase llamada “MyFirebaseMessagingService”, que necesita extender de la clase “FirebaseMessagingService”. Sobrescribiremos el método “onMessageReceived” para mostrar la notificación recibida desde Firebase:

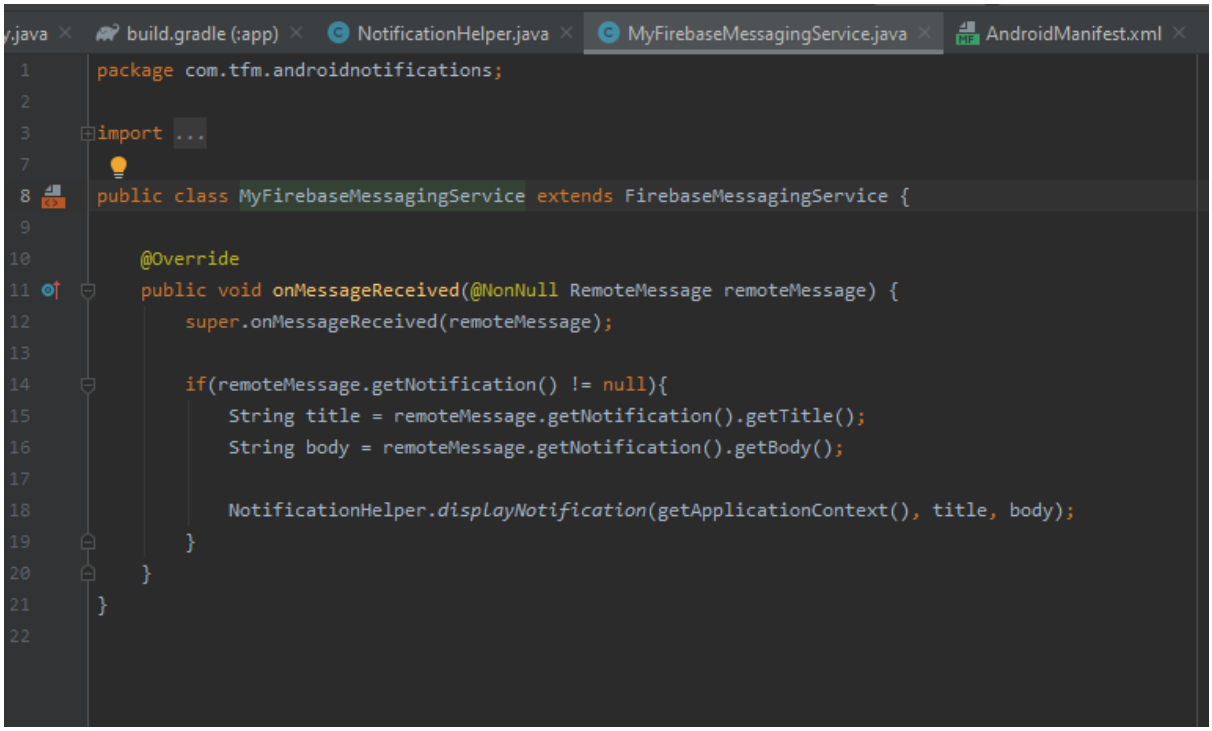


Ilustración 99. Contenido clase MyFirebaseMessagingService.java

En el archivo del manifiesto, AndroidManifest.xml, tenemos que añadir las siguientes líneas:

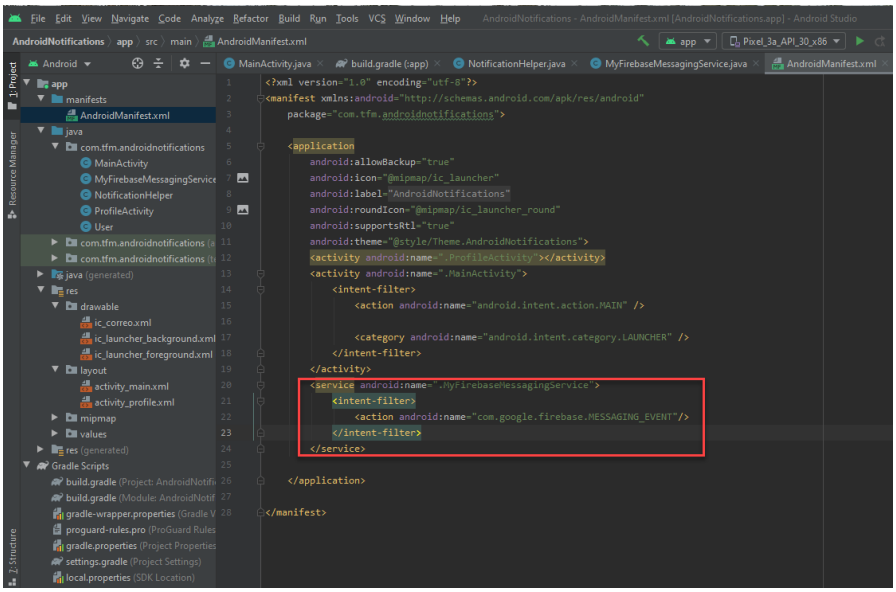


Ilustración 100. Modificación archivo manifiesto

Finalmente testearemos la aplicación, para ello ponemos en marcha la APP, vamos a la consola de Firebase, accedemos a nuestra base de datos y recuperamos el token almacenado:

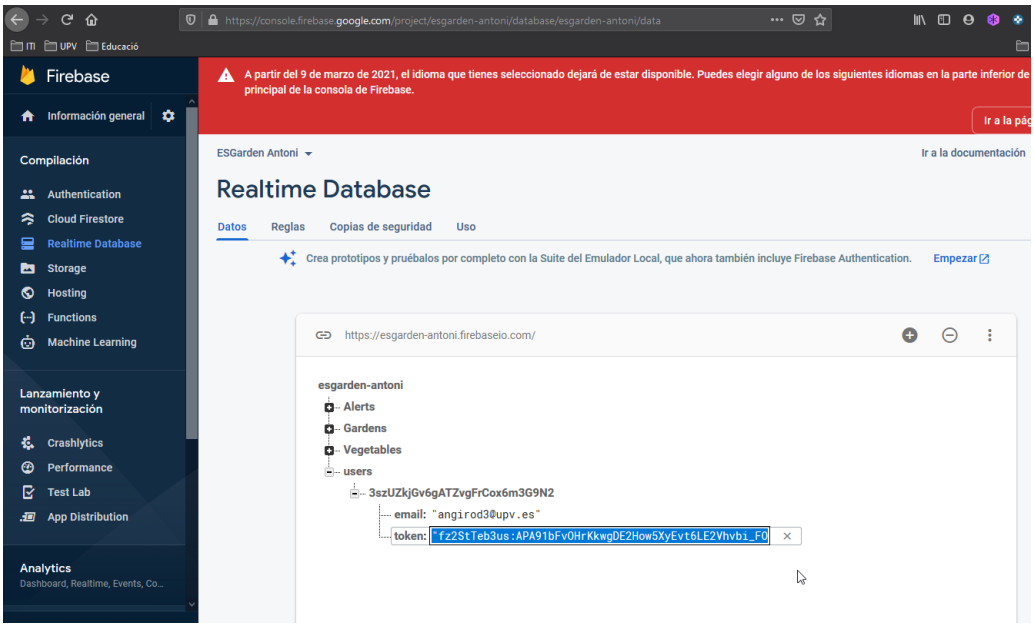


Ilustración 101. Obtención del token

Seguidamente, accederemos a la opción de menú “Cloud messaging”:

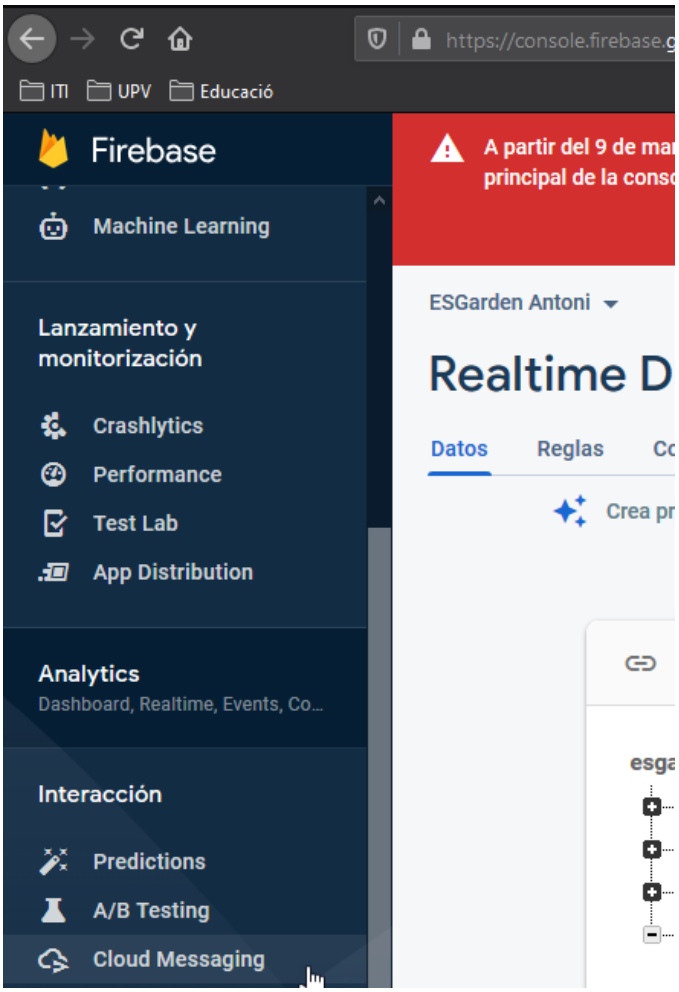


Ilustración 102. Menú Cloud Messaging

Y enviaremos nuestro primer mensaje:

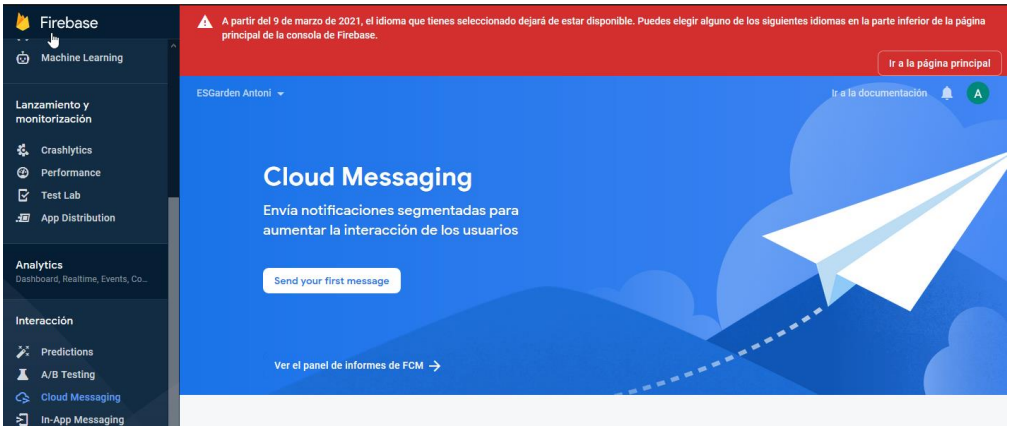


Ilustración 103. Ventana Cloud Messaging de Firebase

Introducimos el título y el cuerpo de la notificación:

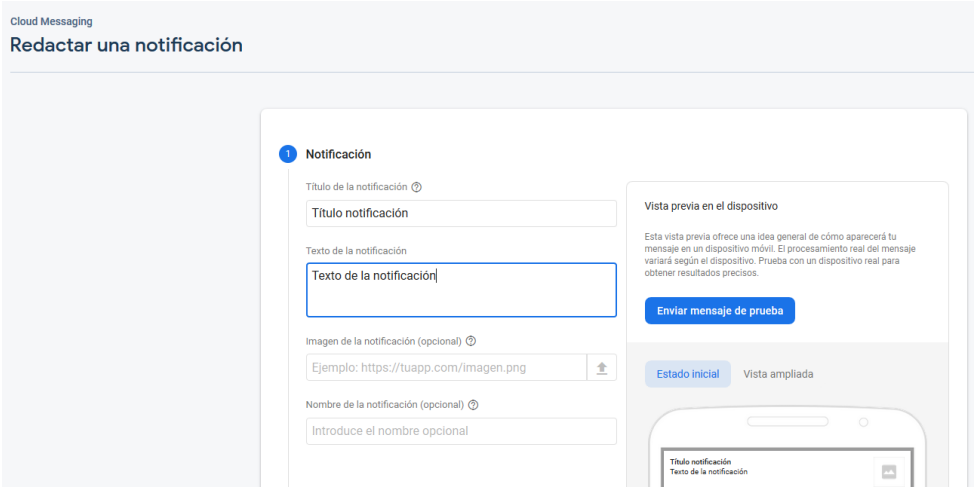


Ilustración 104. Redacción nueva notificación

Al pulsar en "Enviar mensaje de prueba", nos pide el token del dispositivo al que queremos mandar la notificación, pegaremos el token que se encuentra en la base de datos de Firebase:

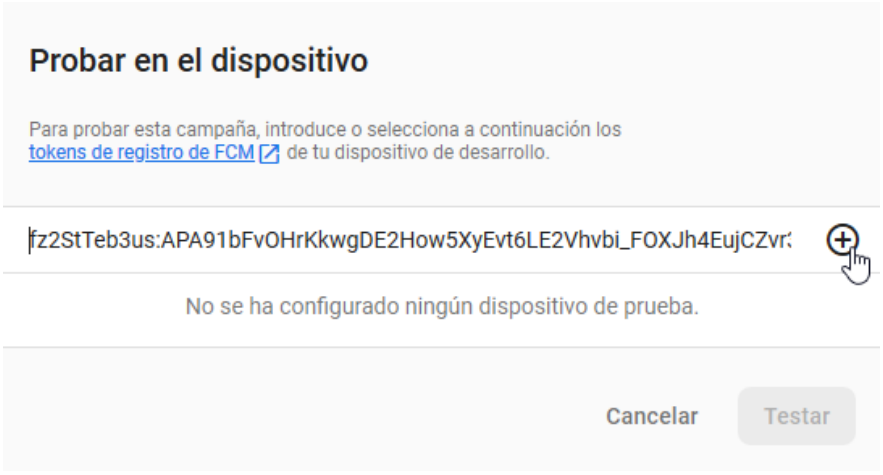


Ilustración 105. Introducción token dispositivo

Agregamos el token pulsando el icono "+" y pulsamos en Testar.

Observaremos, en el emulador, que ha aparecido un icono nuevo, con el icono que definimos anteriormente. Desplegaremos la cortinilla de las notificaciones y observaremos que tenemos una nueva notificación cuyo contenido se corresponde con lo introducido anteriormente.

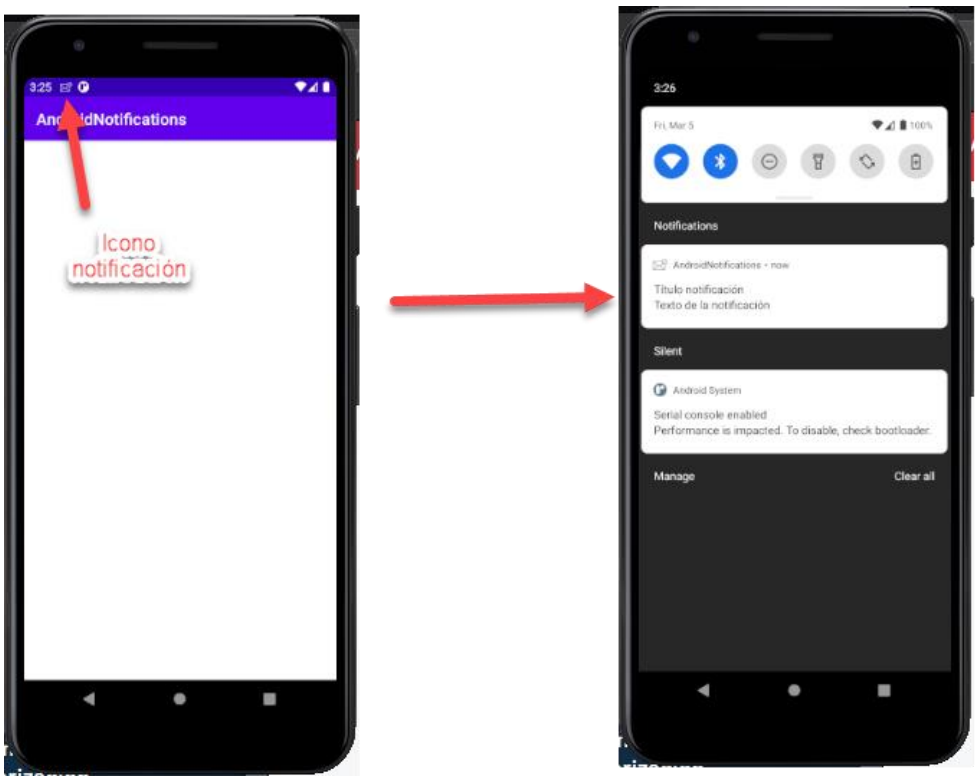


Ilustración 106. Notificación recibida en el dispositivo

Arduino

Una vez tengamos el dispositivo Android correctamente configurado para recibir notificaciones a través del servicio de Firebase Cloud Messaging, procederemos a realizar el proyecto de Arduino encargado de leer los valores de los sensores, si el valor está fuera del rango especificado, mandaremos una notificación al dispositivo Android.

El código final de la implementación se puede encontrar en la siguiente ruta:

https://github.com/antonigimenezrodriguez/Arduino-IoT-33-FCM.git

Para realizar la parte de notificar a través de notificaciones *push* en dispositivos Android cuando un valor supero un cierto umbral, tendremos que haber realizado anteriormente

el Anexo I. Entorno Arduino. Tendremos que instalar las siguientes librerías específicas para esta actividad:

- ArduinoJson:

Ilustración 107. Librería ArduinoJson

- Firebase Arduino based on WiFININA:

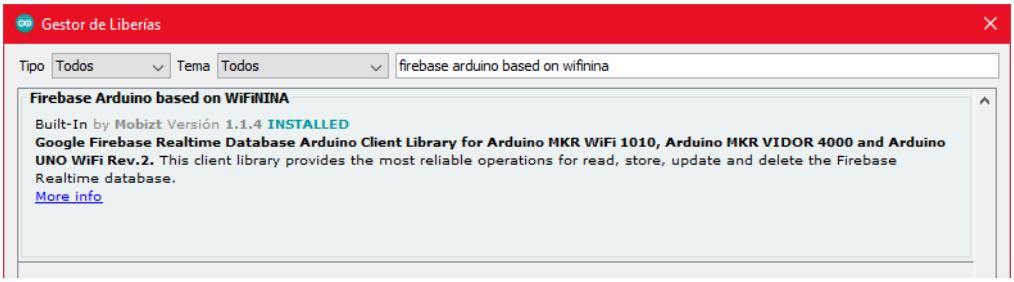


Ilustración 108. Librería Firebase Arduino based on WiFININA

- Memory free:

  Esta librería vamos a descargarla del siguiente enlace: https://github.com/antonigimenezrodriguez/memory-free/blob/main/MemoryFree-master.zip

  Una vez descargado el fichero ZIP, lo descomprimiremos en la ruta de instalación del IDE de Arduino, normalmente en: C/Archivos de programa (x86)/Arduino/libraries

Una vez instaladas las librerías requeridas, tenemos que actualizar el firmware de WiFi NINA y cargar en la placa los certificados de nuestra base de datos de Firebase y del servidor de Firebase Cloud Messaging.

En primer lugar, vamos a cargar el código de actualización de firmware en el Arduino. Menú Archivo / Ejemplos / WiFiNINA / Tools / FirmwareUpdater.

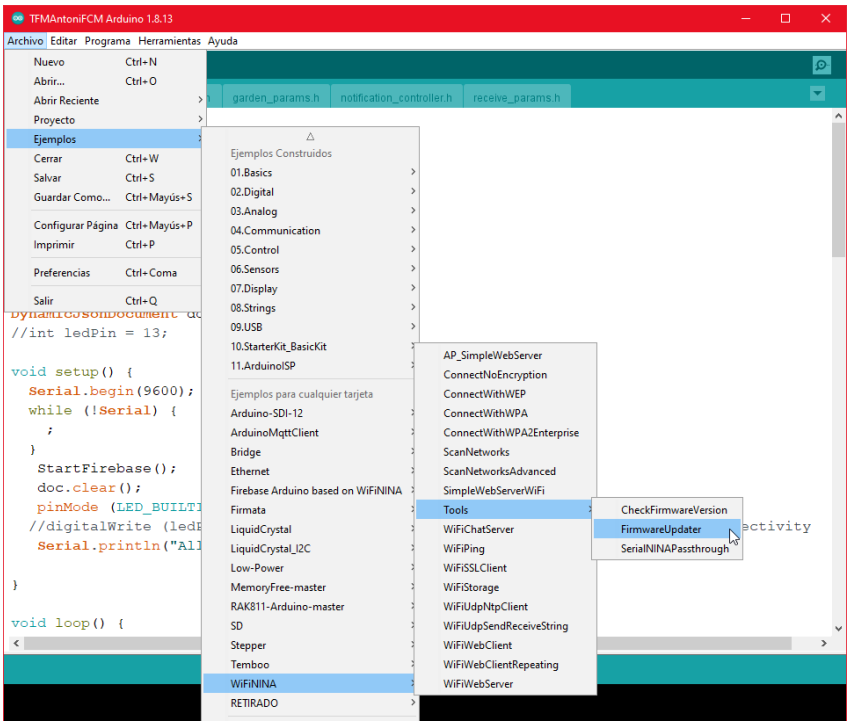


Ilustración 109. Cargar sketch actualziación firmware de WiFiNINA

Subiremos dicho código al Arduino. Seguidamente vamos a ejecutar la siguiente opción: Menú Herramientas / “WiFi101 / WiFiNINA Firmware Updater”

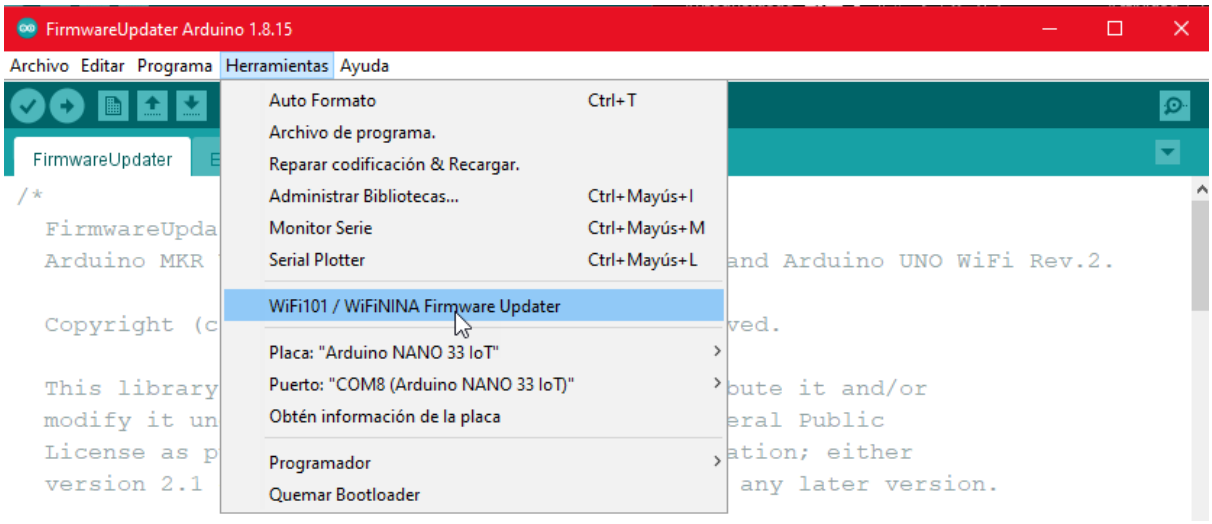


Ilustración 110. Menú actualización firmware WiFiNINA

En la siguiente ventana que se abre, tenemos que seleccionar la placa Arduino NANO 33 IoT con su respectivo puerto COM, y en el punto 2, seleccionaremos la última versión del software:

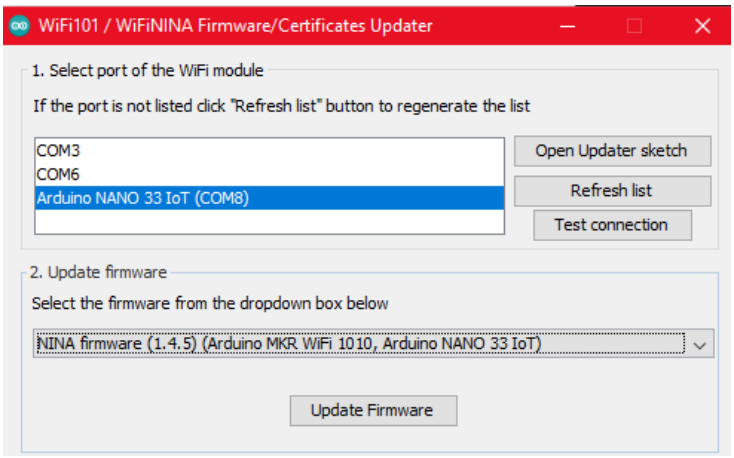


Ilustración 111. Selección de la placa y versión del firmware de NINA

Y actualizamos el firmware mediante el botón “Update Firmware”

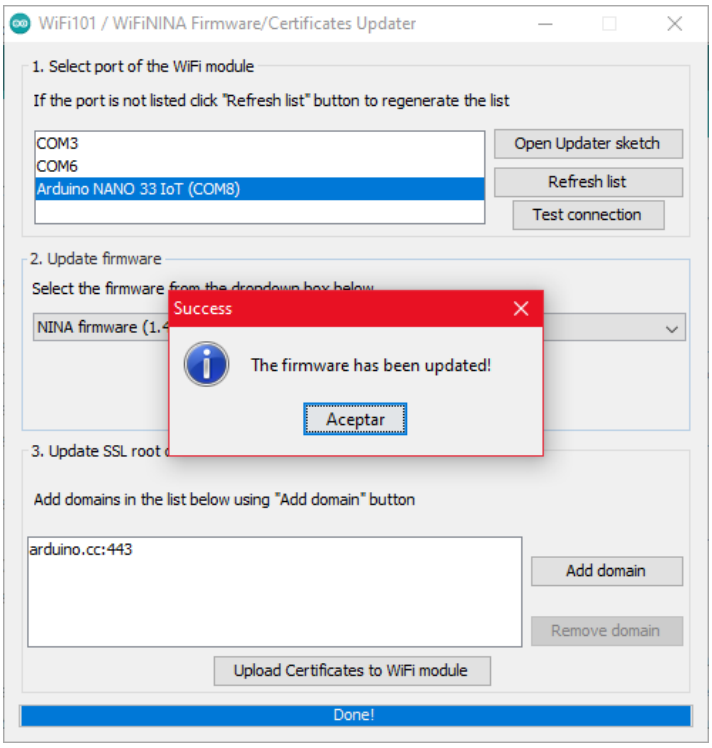


Ilustración 112. Actualización de firmware realizada con éxito

Una vez se haya actualizado el firmware de WiFi NINA, vamos a tener que añadir los certificados SSL, tanto de nuestra base de datos como del servidor de Cloud Messaging,

para ello, iremos al punto 3. “Update SSL root certificates”, pulsaremos en “Add domain” y añadiremos, en primer lugar, la dirección de nuestra base de datos:

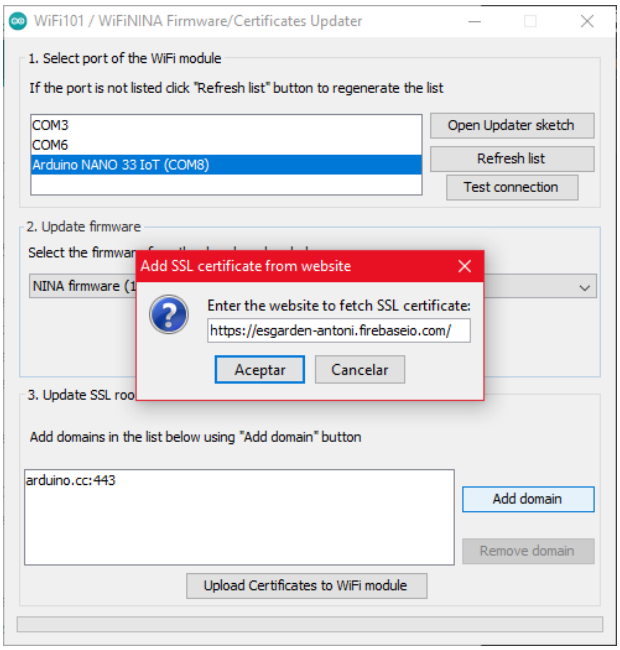


Ilustración 113. Certificado SSL base de datos de Firebase

Y en segundo lugar añadiremos la siguiente dirección: fcm.googleapis.com:443:

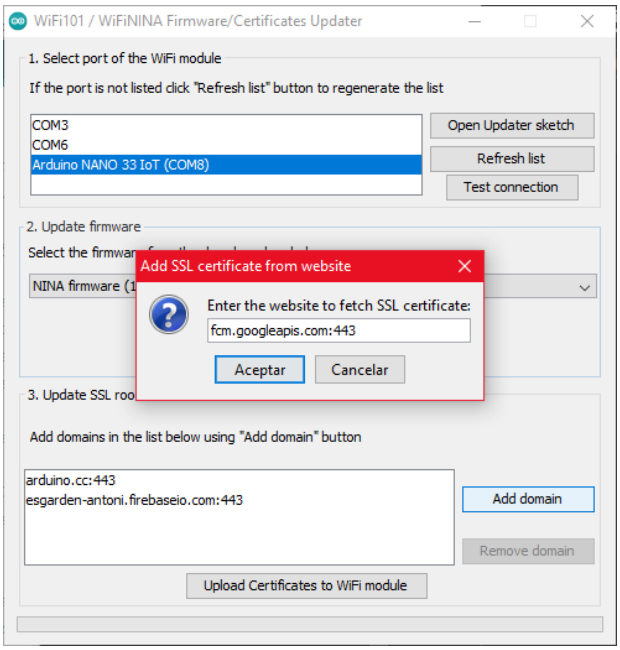


Ilustración 114. Certificado SSL de Firebase Cloud Messaging

Cuando hayamos añadido las 2 URLs, el punto 3 deberemos tenerlo como la siguiente imagen:

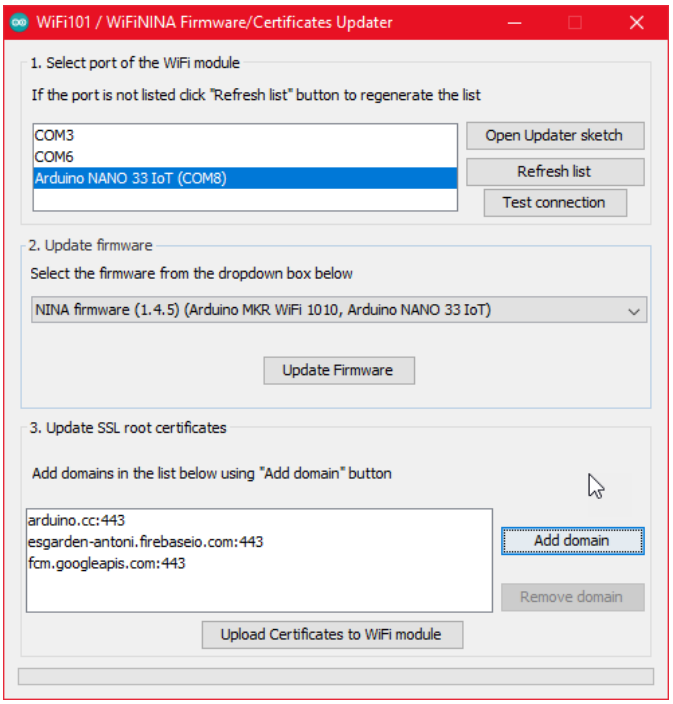


Ilustración 115. Certificados SSL

Finalmente pulsamos en el botón “Upload Certificates to WiFi module” y una vez observado el siguiente mensaje:

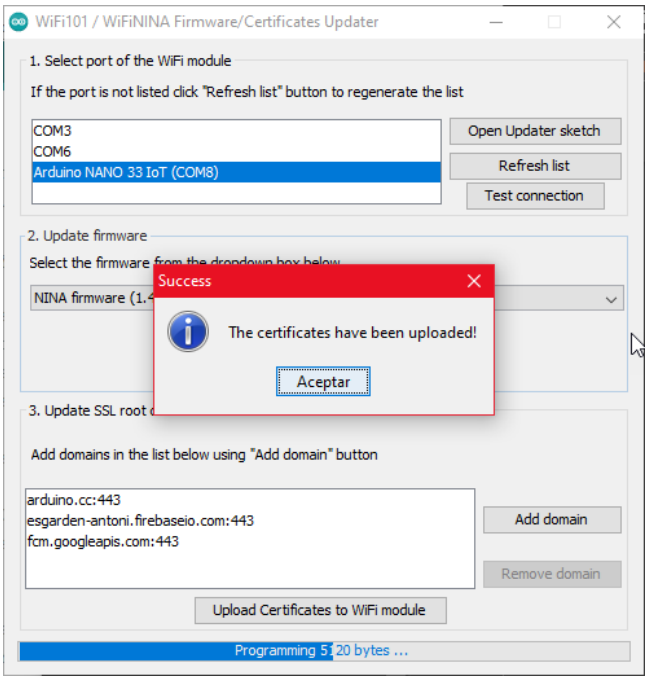


Ilustración 116. Actualización de certificados realizada con éxito

Ya tenemos la placa configurada para poder acceder a los servicios de Firebase.

Sketch Arduino:

Para enviar las notificaciones a través de Firebase Cloud Messaging crearemos un nuevo fichero en el sketch de Arduino que queramos mandar las notificaciones.

En primer lugar, tenemos que recuperar nuestro token de autenticación de Firebase Cloud Message, para ello, accederemos a la consola de Firebase y dentro de la consola, accederemos a la configuración del proyecto:

Ilustración 117. Configuración del proyecto Firebase

Y dentro de la configuración del proyecto, buscaremos la pestaña Cloud Messaging para anotar el Token que observamos:

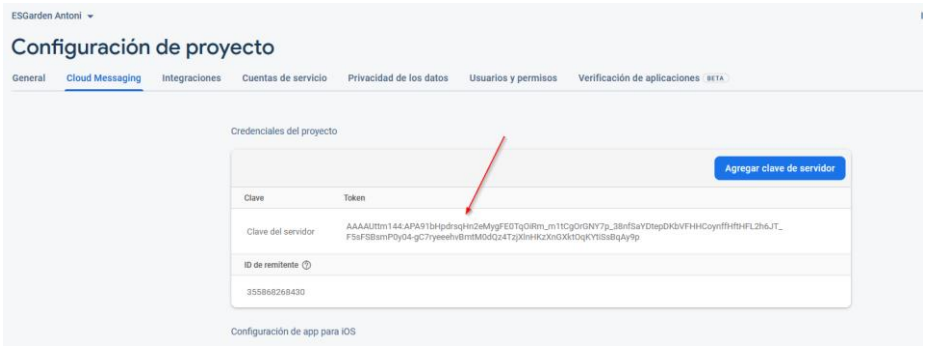


Ilustración 118. Token Cloud Messaging

Una vez tengamos el token anotado, vamos a crear el fichero llamado "notification_controller.h" cuyo contenido será el siguiente:

Librerías, definiciones e inicializaciones:

```
#include <SPI.h>
#define FCM_HOST "fcm.googleapis.com"
#define FCM_AUTH "YOUR_TOKEN "
#define CONTENT_TYPE "application/json"
#define CO2_TYPE 0
#define TEMP_TYPE 1
#define HUM_TYPE 2

WiFiSSLClient client;
void send_notification(uint16_t, int, uint8_t, String);
int getQuantityUsers();
String getTokenUser(int);
```

NOTA: Debemos sustituir la variable FCM_AUTH por el token que hemos anotado anteriormente del panel de control de Firebase.

Funciones auxiliares:

```
int getQuantityUsers() {
  String path = "/users/quantity";
  if (Firebase.getInt(firebaseData, path))
  {
    if (firebaseData.dataType() == "int")
      return firebaseData.intData();
    else if (firebaseData.dataType() == "float")
      return firebaseData.floatData();
  }
  else
  {
    Serial.println("----------Can't get data--------");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println("--------------------------------");
    Serial.println();
  }
}
String getTokenUser(int i) {
  String route = "";
  route = "/users/" + String(i) + "/token";
  Serial.print("Ruta token usuario 1: ");
  Serial.println(route);
  if (Firebase.getString(firebaseData, route))
  {
    if (firebaseData.dataType() == "string")
      return firebaseData.stringData();
    else if (firebaseData.dataType() == "json")
      return firebaseData.jsonData();
  }
  else
  {
    Serial.println("----------Can't get data--------");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println("--------------------------------");
    Serial.println();
  }
}
```

La primera función, “getQuantityUsers()” realiza la función de conectarse a Firebase y recuperar la propiedad “quantity” de la colección usuarios, que nos va a servir para poder iterar y recuperar los tokens de los usuarios.

Función para mandar las notificaciones:

```
void send_notification(uint16_t val, int type, uint8_t isAuto, String sensorName) {

  int quantityUsers = getQuantityUsers();
  delay(10);
  Serial.print("Users: ");
  Serial.println(quantityUsers);
  delay(10);

  for (int i = 1; i <= quantityUsers ; i++) {
    String token = getTokenUser(i);
    Serial.print("Token usuario ");
    Serial.print(i);
    Serial.print(": ");
    Serial.print(token);

    String title, body;
    title = "Limite de " + sensorName + "sobrepasado";
    body = "Valor de " + sensorName + String(val);

    Serial.println("\tEnviando notificacion por sobrepasar limites");
    DynamicJsonDocument params(1024);
    params["to"] = token;
    params["notification"]["title"] = title;
    params["notification"]["body"] = body;
    params["data"]["tipo"] = type;
    params["data"]["value"] = val;
    params["data"]["auto"] = isAuto;

    String data;
    serializeJson(params, data);

    if (client.connect(FCM_HOST, 443)) {
      Serial.println("\tConectando con el servidor...");
      client.println("POST /fcm/send HTTP/1.1");
      client.println("Host: fcm.googleapis.com");
      client.println("Authorization: key=" + String(FCM_AUTH));
      client.println("Content-Type: application/json");
      client.print("Content-Length: ");
      client.println(data.length());
      client.print("\n");
      client.print(data);
    }
    Serial.println("\tDatos enviados");
    client.flush();
    client.stop();
  }
}
```

La función principal, se encarga de recibir cuál es el sensor que se encuentra fuera de los límites, en una primera instancia consulta el número de usuarios registrados para así poder recuperar el token de la aplicación de cada uno para posteriormente enviar una notificación a cada uno de los dispositivos registrados.

Anexo III. Conectar WordPress con Arduino a través de un bróker MQTT

Este anexo explica como recibir, en una pantalla LCD de 20x4 conectada a un Arduino, distintos eventos que se producen en un Blog de WordPress.

En primer lugar, aprenderemos a crear una máquina virtual con Sistema Operativo CentOS7. Seguidamente, instalaremos las utilidades Linux, Apache, MySQL, PHP (LAMP) así como las dependencias necesarias. Posteriormente, procederemos a la instalación y configuración de WordPress y el complemento WP-MQTT (encargado de publicar en el bróker MQTT).

Finalmente, descubriremos como suscribirnos al bróker MQTT y como escribiremos ese resultado en el panel LCD.

Instalar Máquina virtual CentOS

Requisitos previos

Descargaremos e instalaremos la última versión de VirtualBox de la siguiente dirección: https://www.virtualbox.org/wiki/Downloads

Descargaremos CentOS 7 minimal, en este caso, vamos a usar la versión 2009:

http://mirror.uv.es/mirror/CentOS/7.9.2009/isos/x86_64/CentOS-7-x86_64-Minimal-2009.iso

Máquina virtual

Una vez hayamos descargado VirtualBox y la ISO de CentOS7, así como haber realizado una instalación típica de VirtualBox, procederemos a realizar los siguientes pasos:

Abriremos Virtual box y crearemos una nueva máquina virtual:

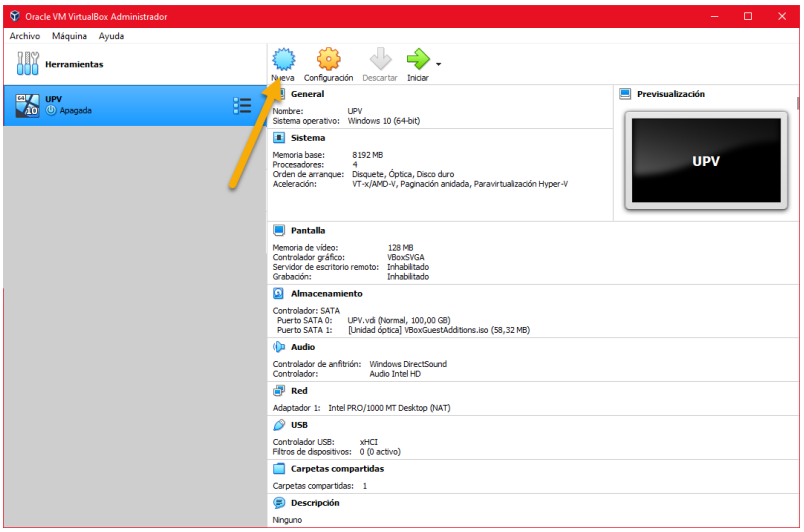


Ilustración 119. Interfaz de VirtualBox

En el siguiente paso, seleccionaremos Tipo: Linux y Versión: Red Hat (64-bit)

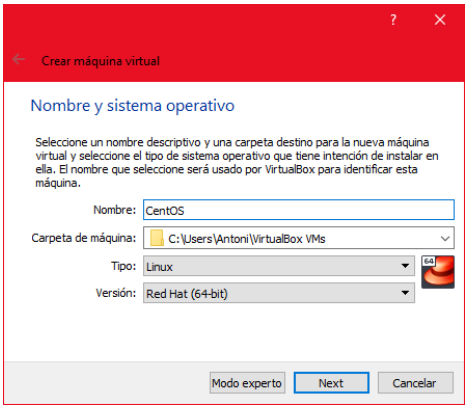


Ilustración 120. Ventana creación nueva máquina virtual

En el siguiente paso, escogeremos 2048MB de memoria como mínimo.

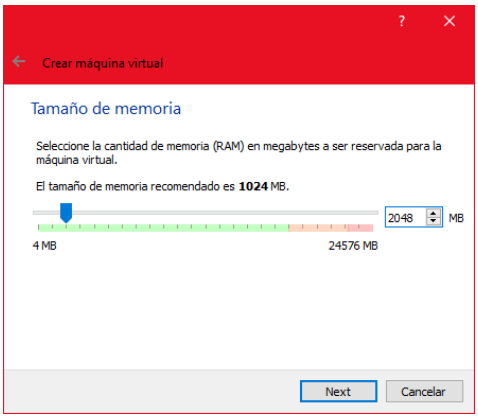


Ilustración 121. Ventana selección memoria RAM

Seguidamente, crearemos un disco duro virtual nuevo ahora.

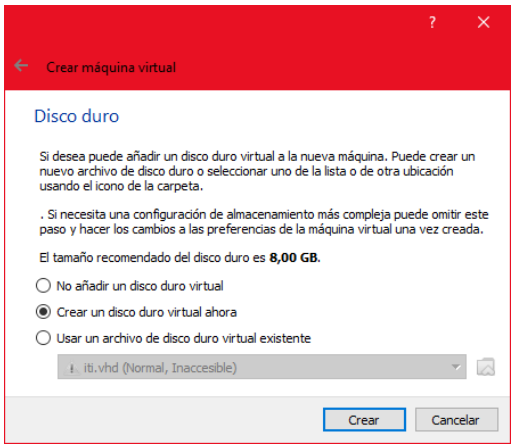


Ilustración 122. Ventana creación disco duro

Para el tipo de disco duro seleccionaremos VDI (VirtualBox Disk Image)

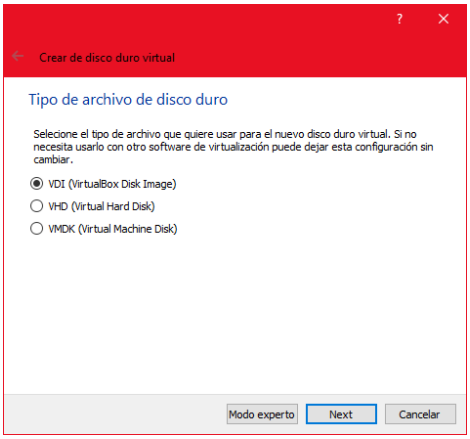


Ilustración 123. Ventana creación tipo disco duro

Marcaremos como el tamaño como Reservado dinámicamente

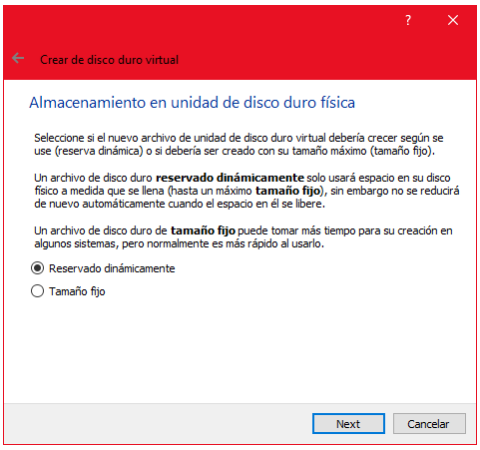


Ilustración 124. Ventana tipo tamaño disco duro

Seleccionamos la ruta de la máquina virtual y seleccionaremos 60Gb de disco duro

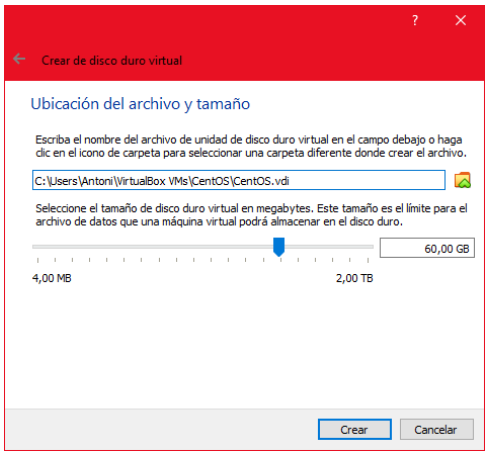


Ilustración 125. Ventana tamaño disco duro

Una vez realizados los pasos anteriores, tendremos nuestra máquina virtual creada. El siguiente paso, consistirá en la instalación del Sistema Operativo, por tanto, iremos a configuración, luego a Almacenamiento, Controlador: IDE y añadimos la ISO de CentOS7 que hemos descargado previamente.

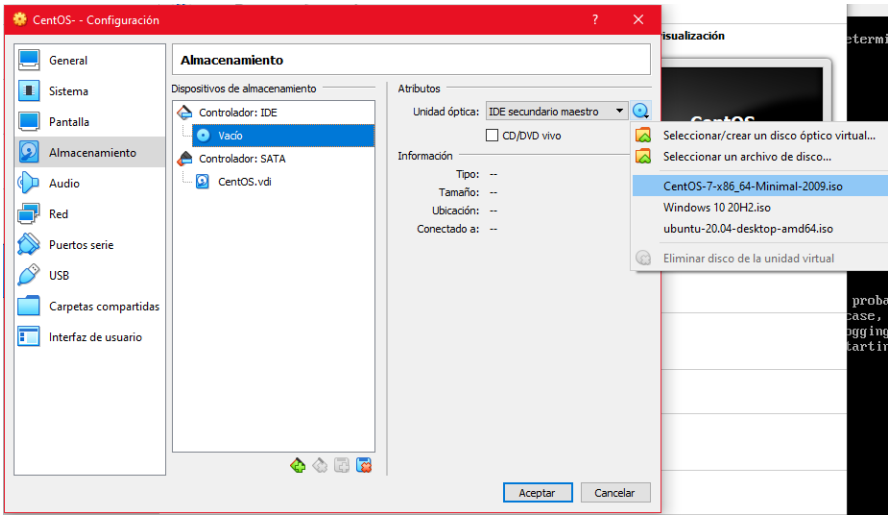


Ilustración 126. Inserción Sistema Operativo

Iniciamos la máquina pulsando el botón de iniciar, con ello empezará a arrancar la máquina virtual y empezará la instalación del Sistema Operativo contenido en la imagen ISO cargada.

En la siguiente ventana, nos preguntará si queremos instalar el sistema, o simplemente probarlo sin instalarlo. En nuestro caso vamos a seleccionar “Install CentOS7”, ya que, si no lo instalamos, al apagar o reiniciar la máquina, vamos a perder todos los cambios realizados.

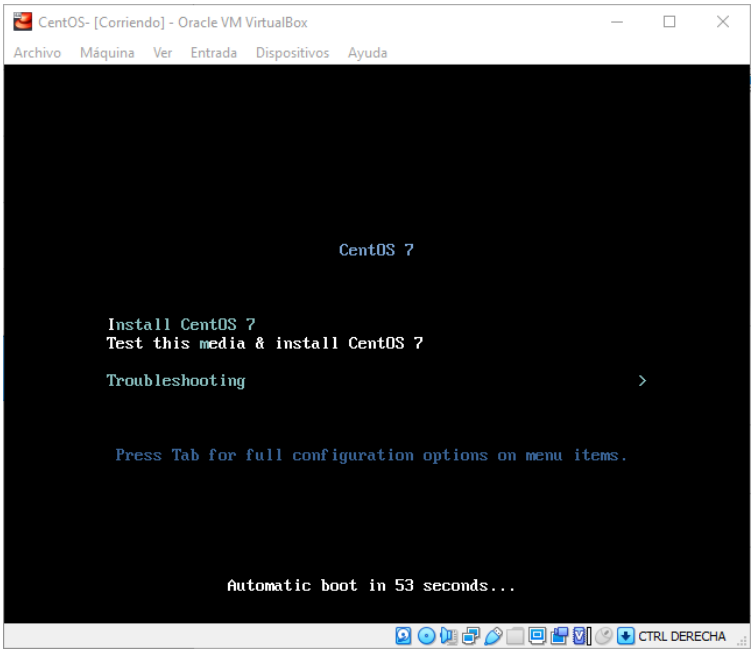


Ilustración 127. Primer arranque Sistema Operativo

El siguiente paso es elegir el idioma, en nuestro caso vamos a elegir español de España.

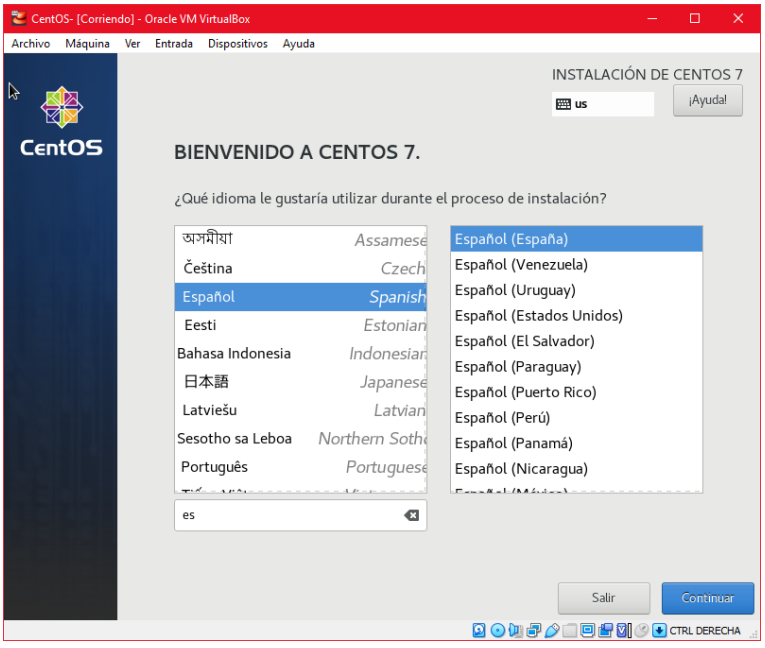


Ilustración 128. Ventana selección idioma

Una vez configurado el idioma de nuestro sistema, tenemos que especificar donde queremos que se realice dicha instalación, seleccionaremos “Destino de la instalación”.

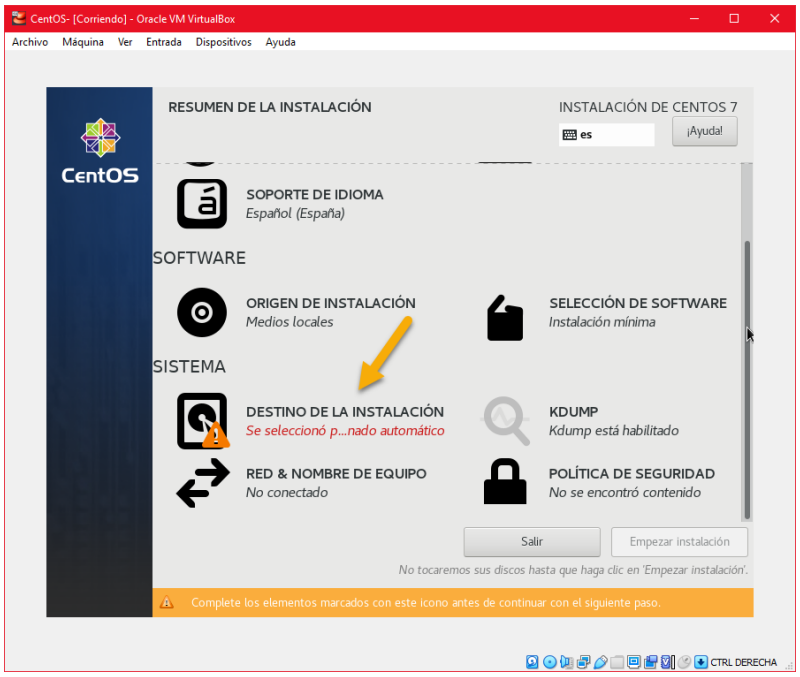


Ilustración 129. Ventana resumen de la instalación

Una vez hayamos seleccionado el disco que creamos en la creación de la máquina virtual, pulsaremos el botón “Listo”.

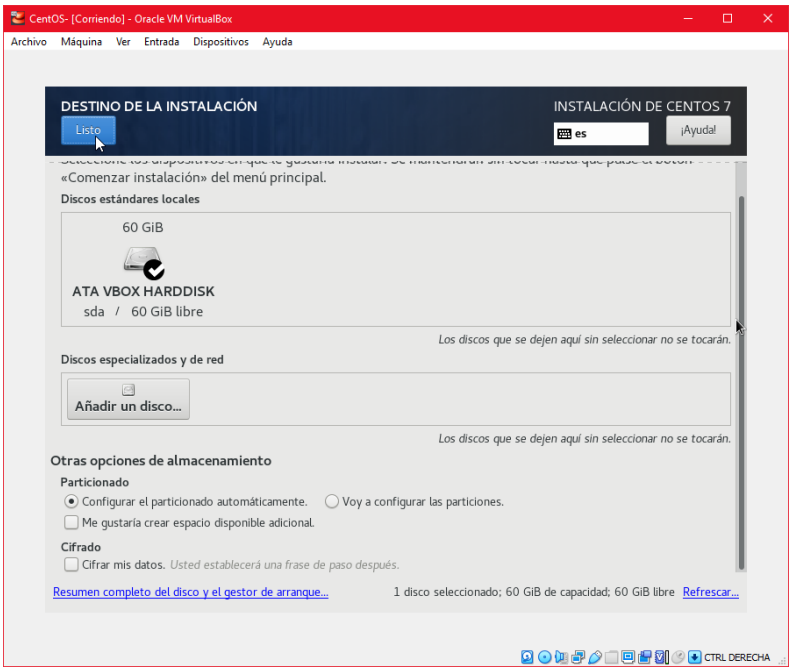


Ilustración 130. Ventana destino de la instalación

Seguidamente cambiaremos la configuración de Red & Nombre de equipo pulsando dicho botón.

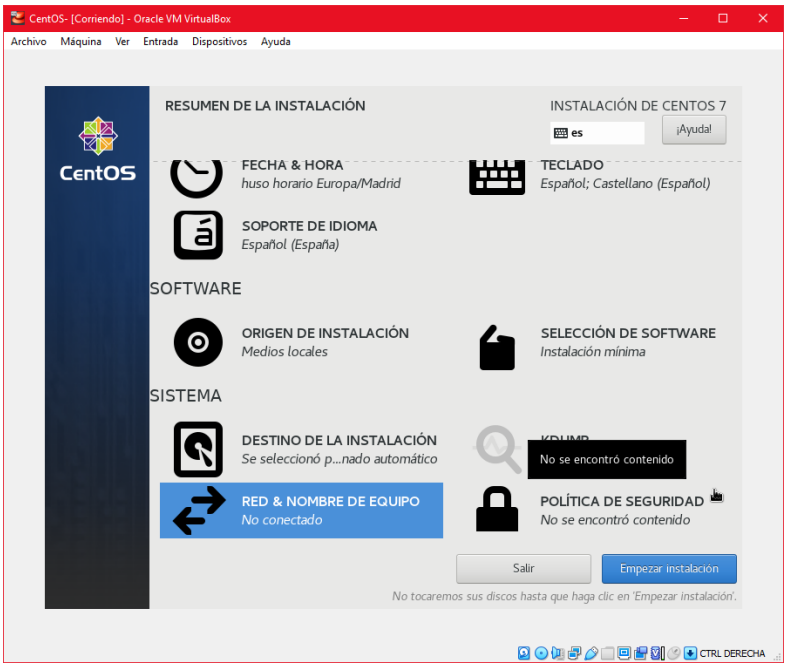


Ilustración 131. Ventana resumen de la instalación

Cambiaremos el interruptor que aparece a conectado y pulsaremos en listo.

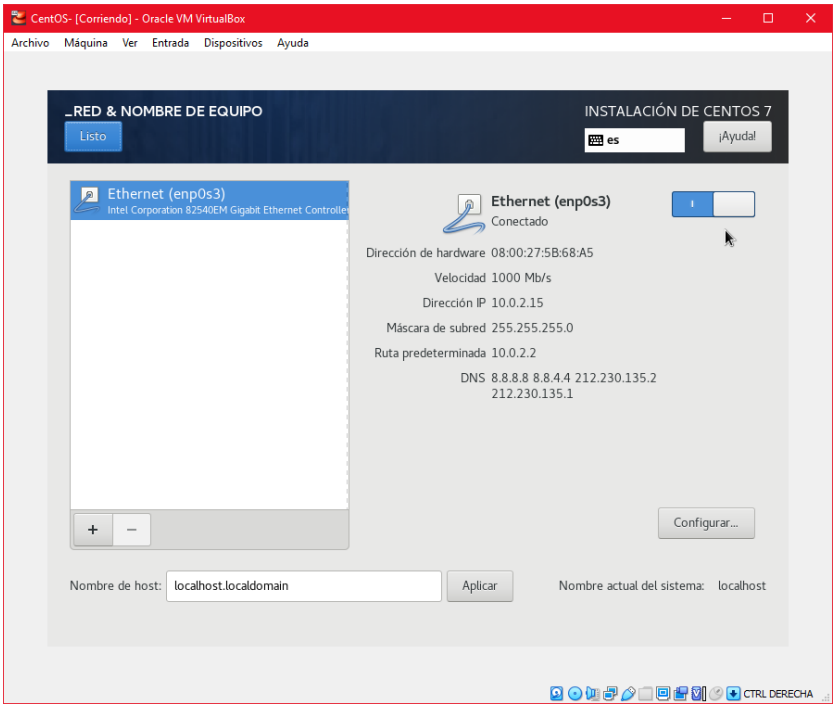


Ilustración 132. Ventana Red & Nombre de equipo

Una vez realizado los pasos mencionados, ya podremos empezar la instalación, por tanto, pulsaremos el botón "Empezar Instalación". Mientras se instala CentOS 7, éste nos preguntará por el usuario y la contraseña de ROOT. Es muy importante anotar estas claves, ya que las necesitaremos para entrar a la máquina y en caso de pérdida tendremos que crear una nueva máquina y perderemos todos los datos.

Cuando haya finalizado la instalación, accederemos a la máquina con el usuario "root" y contraseña la que hemos escrito anteriormente.

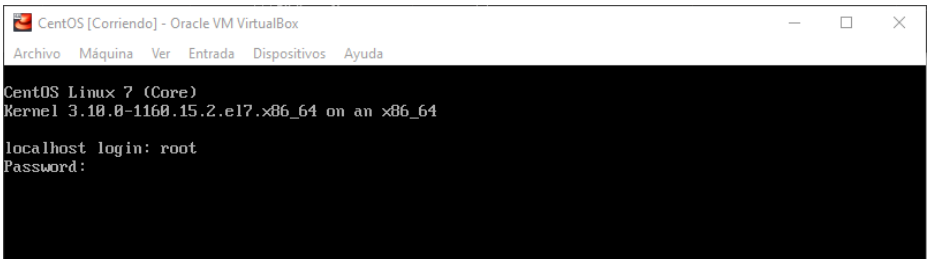


Ilustración 133. Página inicio CentOS 7

Una vez hayamos introducido el inicio de sesión correcto, actualizaremos la máquina mediante el siguiente comando:

```
# yum update -y
```

Llegados a este punto, ya tenemos creada la máquina virtual, así como el Sistema Operativo instalado. Ahora procederemos a instalar las dependencias necesarias.

Instalar LAMP

LAMP significa Linux, Apache, MySQL y PHP. Una pila común para crear e implementar aplicaciones web dinámicas. Para poder instalar y usar correctamente WordPress, necesitamos instalar estos 4 paquetes.

Instalar Apache

En primer lugar, instalaremos apache mediante el siguiente comando:

```
# sudo yum install httpd
```

Una vez haya terminado la instalación, debemos iniciar apache:

```
# sudo systemctl start httpd.service
```

Y habilitamos el arranque automático de apache con el inicio del Sistema Operativo:

```
# sudo systemctl enable httpd.service
```

Para que apache pueda funcionar correctamente, deshabilitamos el firewall de CentOS:

```
# systemctl stop firewalld
# setenforce 0
```

El siguiente paso será comprobar el acceso a la página por defecto de apache, para ello, primero tenemos que ver la IP de nuestra máquina con el comando:

```
# ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

En mi caso, la ejecución de este comando me muestra los siguientes resultados:

```
[root@localhost ~]# ip addr show enp0s3 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
192.168.1.125
fe80::959b:2b02:1858:305d
```

Ilustración 134. Salida dirección IP

La dirección que estamos buscando es la primera, en mi caso: 192.168.1.125.

Ahora accedemos a esa dirección en un navegador de la máquina principal:

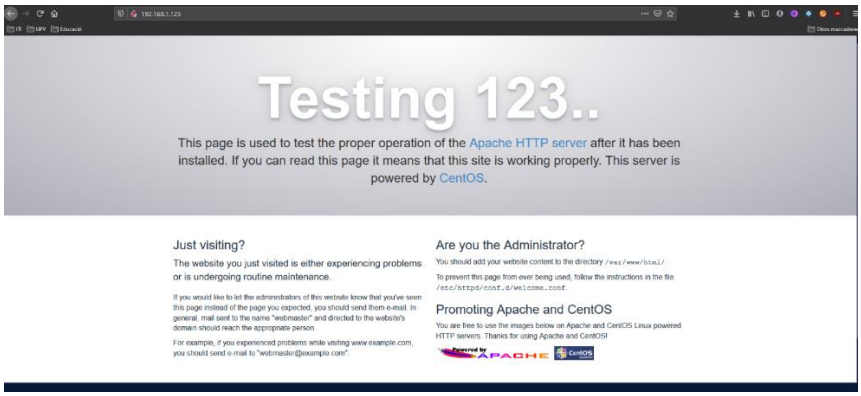


Ilustración 135. Página inicial Apache

Si observamos esta página web, significa que la instalación de apache se ha realizado correctamente.

Instalar MySQL (MariaDB)

El siguiente paso, será instalar el servidor de base de datos MySQL, y más concretamente, MariaDB:

```
# sudo yum install mariadb-server mariadb
```

Una vez haya terminado la instalación, iniciaremos la base de datos y la habilitaremos en el arranque del Sistema Operativo:

```
# sudo systemctl start mariadb

# sudo systemctl enable mariadb.service
```

Y, por último, habilitamos seguridad en la base de datos:

```
# sudo mysql_secure_installation
```

Instalar PHP

Seguidamente, instalaremos PHP con el siguiente comando:

```
# sudo yum install php php-mysql
```

Para que Apache reconozca la instalación de PHP realizada, tenemos que reiniciar el servicio:

```
# sudo systemctl restart httpd.service
```

Finalmente, instalaremos los módulos adicionales necesarios:

```
# sudo yum install php-fpm
```

Testear PHP

Una vez llegados a este punto, ya tenemos la instalación LAMP finalizada, ahora vamos a testear que todo esté correcto. Para ello, vamos a crear una página de información con el siguiente comando:

```
# sudo vi /var/www/html/info.php
```

Esta instrucción nos abrirá un fichero alojado en "/var/www/html/" cuyo contenido será vacío, dentro del fichero escribimos lo siguiente:

```
<?php phpinfo(); ?>
```

NOTA: para poder escribir en el fichero, tenemos que pulsar la tecla "i" y aparecerá "—INSERT—"en la parte inferior de la pantalla:

Ilustración 136. Inserción en vi

Una vez hayamos escrito en el fichero, pulsaremos la tecla "Esc", con lo que desaparecerá la línea de *insert* y escribiremos ":wq" y pulsaremos intro.

Ilustración 137. Contenido del fichero info.php

Seguidamente, habilitamos el firewall para permitir el tráfico http y https, y así poder acceder a la página que hemos creado en el punto anterior:

```
# systemctl start firewalld

# sudo firewall-cmd --permanent --zone=public --add-service=http

# sudo firewall-cmd --permanent --zone=public --add-service=https

# sudo firewall-cmd --reload
```

Cuando hayamos introducido estos 4 comandos, los 3 últimos nos devolverá como respuesta "success", podemos visitar visitaremos la siguiente web http://nuestra_IP/info.php Siendo "nuestra_IP" la IP obtenida anteriormente, en el caso del ejemplo será: http://192.168.1.125/info.php Cuyo resultado es el siguiente:

Ilustración 138. Resultado página info.php

Una vez visualizada dicha página correctamente ha finalizado correctamente la prueba de funcionamiento de PHP en nuestra máquina.

Eliminar fichero información

Es importante eliminar el fichero “info.php” que hemos creado, ya que proporciona información sobre nuestro servidor que no se debe mostrar al exterior para evitar posibles ataques. Es por ello, que ejecutaremos el siguiente comando para eliminar dicha página:

```
# sudo rm /var/www/html/info.php
```

Instalación de WordPress

Una vez hemos realizado la instalación LAMP definida anteriormente, vamos a proceder con la instalación del servidor WordPress.

Base de datos

En primer lugar, vamos a configurar la base de datos para WordPress. Para ello, ejecutamos MySQL como *root* mediante el siguiente comando:

```
# mysql -u root -p
```

Una vez hayamos entrado dentro de MySQL

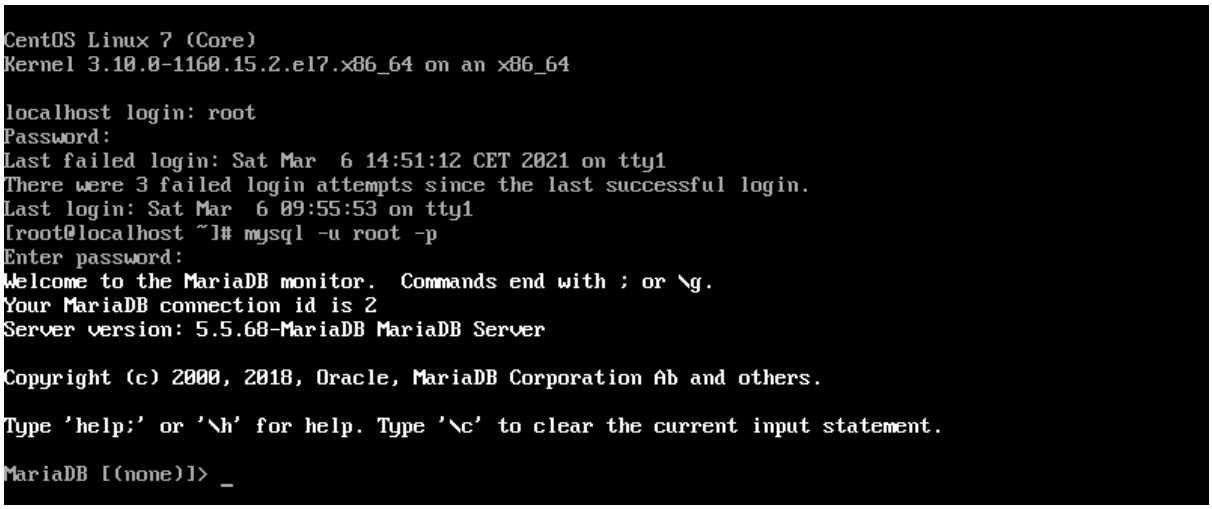


Ilustración 139. Monitor MariaDB

Ejecutaremos las siguientes las siguientes instrucciones:

```
CREATE DATABASE wordpress;

CREATE USER wordpressuser@localhost IDENTIFIED BY ‘password’;
```

NOTA: wordpressuser y password, lo tenemos que sustituir por los valores que deseemos.

```
GRANT ALL PRIVILEGES ON wordpress.* TO wordpressuser@localhost IDENTIFIED BY
'password';

FLUSH PRIVILEGES;

exit
```

En este punto ya tenemos la base de datos preparada para instalar WordPress.

Instalación WordPress

Antes de instalar WordPress, necesitamos instalar un paquete de PHP, para ello, ejecutaremos la siguiente instrucción:

```
# sudo yum install php-gd

# sudo service httpd restart
```

Seguidamente, descargaremos el instalable de WordPress (Versión 5.4.4):

```
# cd ~

# wget http://wordpress.org/wordpress-5.4.4.tar.gz
```

NOTA: en caso de no tener instalada la utilidad wget, podemos instalarla con:

```
# sudo yum install wget
```

Una vez descargado el fichero "wordpress-5.4.4.tar.gz" procedemos a descomprimirlo:

```
# tar xzvf wordpress-5.4.4.tar.gz
```

Cuando se haya descomprimido el fichero, observaremos que tenemos un nuevo directorio llamado "wordpress". Necesitaremos mover dicho directorio al directorio de apache:

```
# yum install rsync

# sudo rsync -avP ~/wordpress/ /var/www/html/
```

Crearemos un directorio para almacenar ficheros:

```
# mkdir /var/www/html/wp-content/uploads
```

Y. por último, reasignamos correctamente los permisos:

```
# sudo chown -R apache:apache /var/www/html/*
```

Configuración WordPress

Una vez realizados los pasos para instalar WordPress, vamos a configurarlo, para ello, nos situaremos en la carpeta donde se aloja nuestro WordPress y usaremos como base el fichero de configuración de ejemplo:

```
# cd /var/www/html

# cp wp-config-sample.php wp-config.php
```

A continuación, abriremos ese fichero en un editor de textos y modificaremos el siguiente contenido:

```
# vi wp-config.php
```

```
// ** MySQL settings - You can get this info from your web host ** //

/** The name of the database for WordPress */

define('DB_NAME', 'wordpress');


/** MySQL database username */

define('DB_USER', 'wordpressuser');


/** MySQL database password */

define('DB_PASSWORD', 'password');
```

Finalizando instalación desde el navegador web

Para finalizar con la instalación de WordPress, vamos a acceder desde la máquina principal a la página web WordPress (IP obtenida anteriormente) y vamos a seguir los pasos del asistente

NOTA:

En caso de aparecer un error como este, o similar:

"Your server is running PHP version 5.4.16 but WordPress 5.6.2 requires at least 5.6.20."

Tendremos que actualizar la versión de Apache, siguiendo los siguientes pasos:

```
# yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

# yum install -y https://rpms.remirepo.net/enterprise/remi-release-7.rpm

# yum install -y --enablerepo=remi-php74 php php-cli

# systemctl restart httpd
```

Comprobamos con: "php -v" que tengamos instala la versión 7.4 de PHP

Ahora, ya podemos seguir los típicos pasos de una instalación de WordPress. Primeramente, escogeremos el idioma:

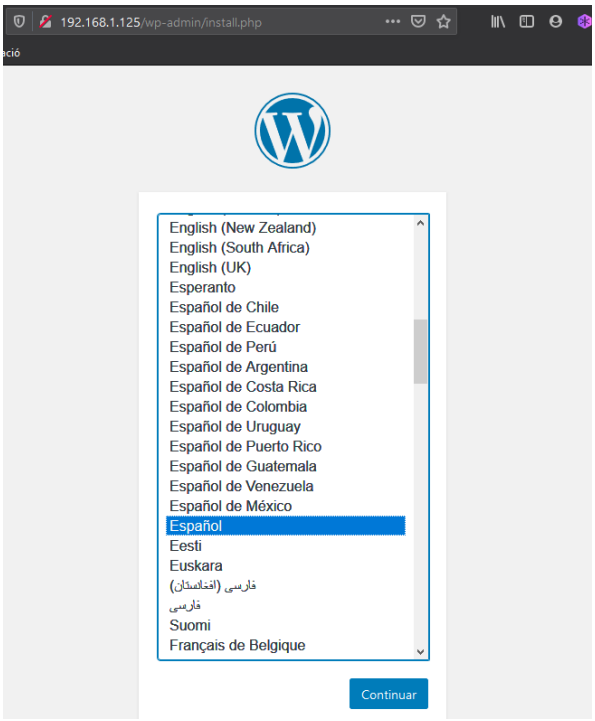


Ilustración 140. Selección idioma WordPress

Seguidamente, definiremos información sobre de nuestro blog, tales como el título y el usuario de acceso:

Ilustración 141. Información básica WordPress

Cunado pinchemos el botón “Instalar WordPress” ya tendremos listo nuestro sitio.

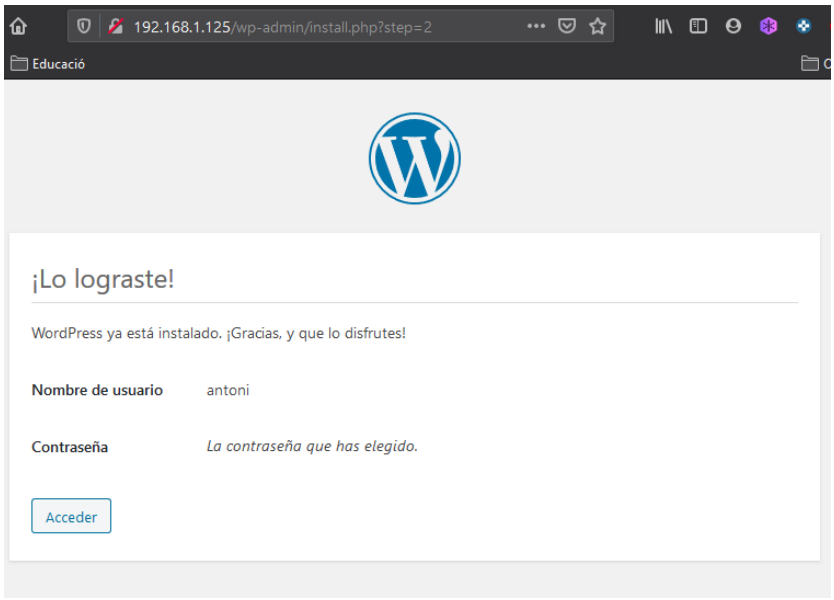


Ilustración 142. Finalización instalación WordPress

Accederemos al panel de administración con las credenciales definidas anteriormente.

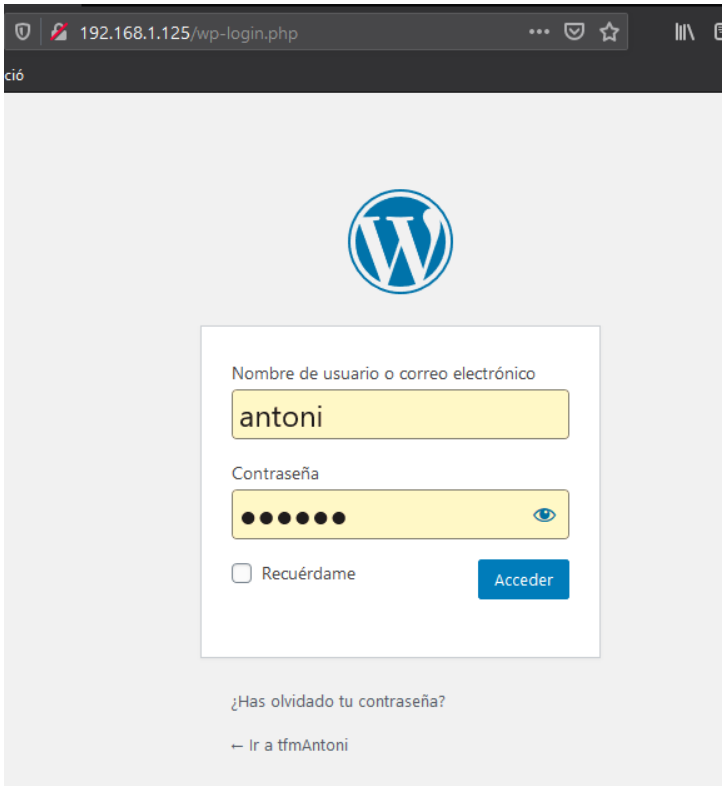


Ilustración 143. Login WordPress

Una vez validadas dichas claves, ya tendremos acceso al panel de WordPress:

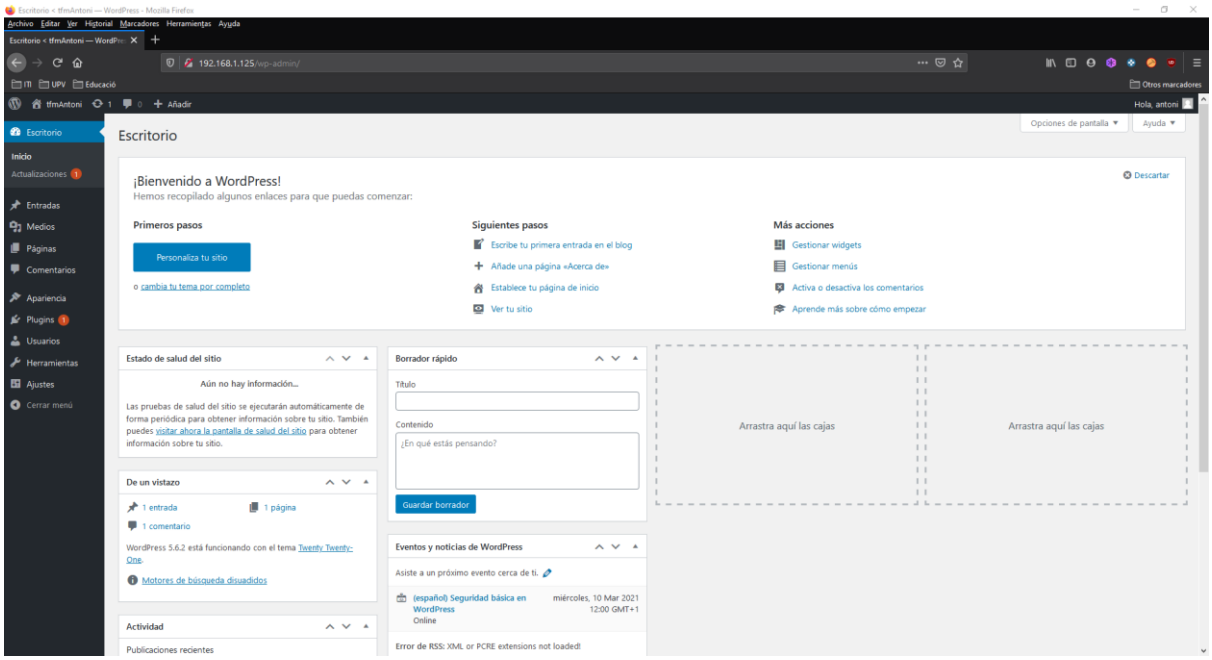


Ilustración 144. Panel de WordPress

Instalación plugin WP-MQTT

Una vez tengamos funcionando correctamente WordPress, vamos a hacer uso del plugin WP-MQTT. Dicho plugin nos permite publicar en un bróker MQTT los siguientes eventos ocurridos en nuestro blog:

- Página visitada
- Inicio sesión usuario
- Fallo al iniciar sesión usuario
- Nuevo post publicado
- Nueva página publicada
- Nuevo comentario publicado

En primer lugar, descargaremos una versión del plugin WP-MQTT que hemos adaptado para las nuevas versiones de WordPress. Para ello descargaremos el siguiente fichero zip en la máquina principal:

https://github.com/antonigimenezrodriguez/wp-mqtt/blob/main/wp-mqtt.zip

Desactivar el firewall en la máquina virtual:

```
# systemctl stop firewalld

# systemctl disable firewalld

# setenforce 0
```

Necesitamos cambiar el *enforcing* del SELinux para el correcto funcionamiento del plugin.

Para ello modificaremos el siguiente fichero:

```
Vi /etc/selinux/config
```

Y lo dejaremos de la siguiente forma:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Ilustración 145. Contenido fichero selinux config

Seguidamente, iremos al menú de *plugins* y añadiremos un plugin nuevo:

Ilustración 146. Panel plugins WordPress

Y pulsaremos el botón “Subir plugin”

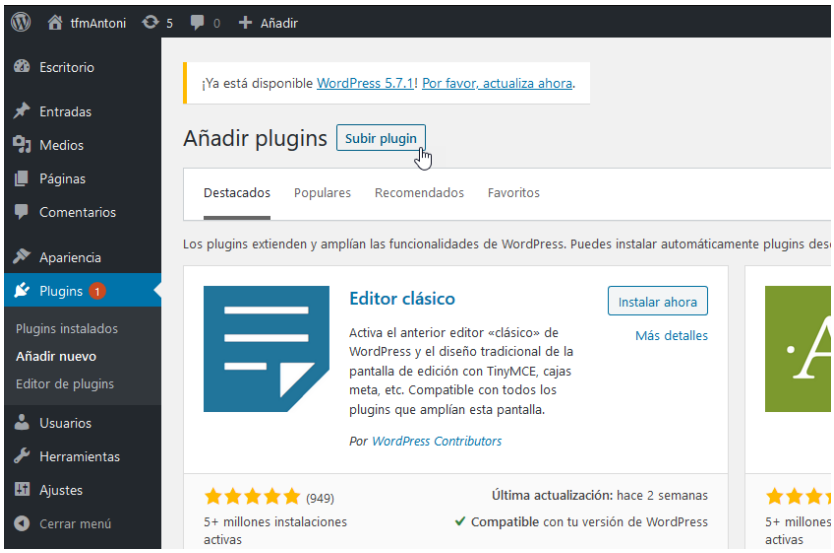


Ilustración 147. Panel añadir plugin de WordPress

En el siguiente menú, seleccionaremos el archivo wp-mqtt.zip, pulsando en el botón "Examinar", que hemos descargado del repositorio de GitHub:

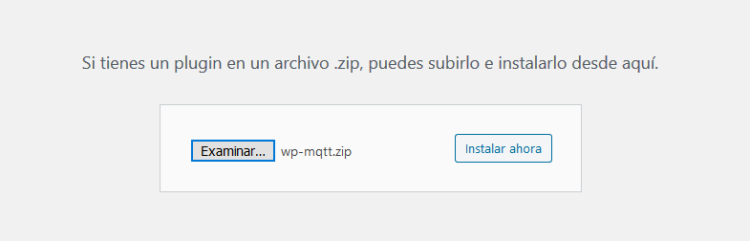


Ilustración 148. Panel subir plugin de WordPress

Una vez instalado, tenemos que activar el plugin. Una vez activado, vamos a configurarlo. Para ello, iremos al panel, ajustes, WP-MQTT

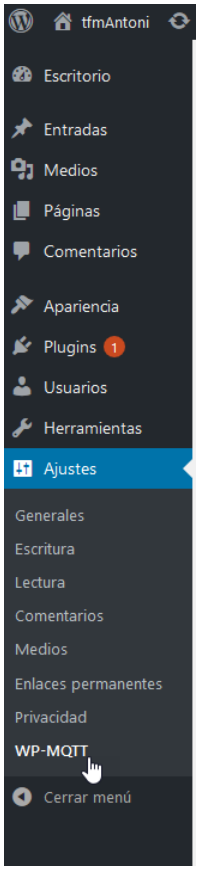


Ilustración 149. Panel ajustes

Una vez dentro, vamos a configurar el plugin de la siguiente forma:

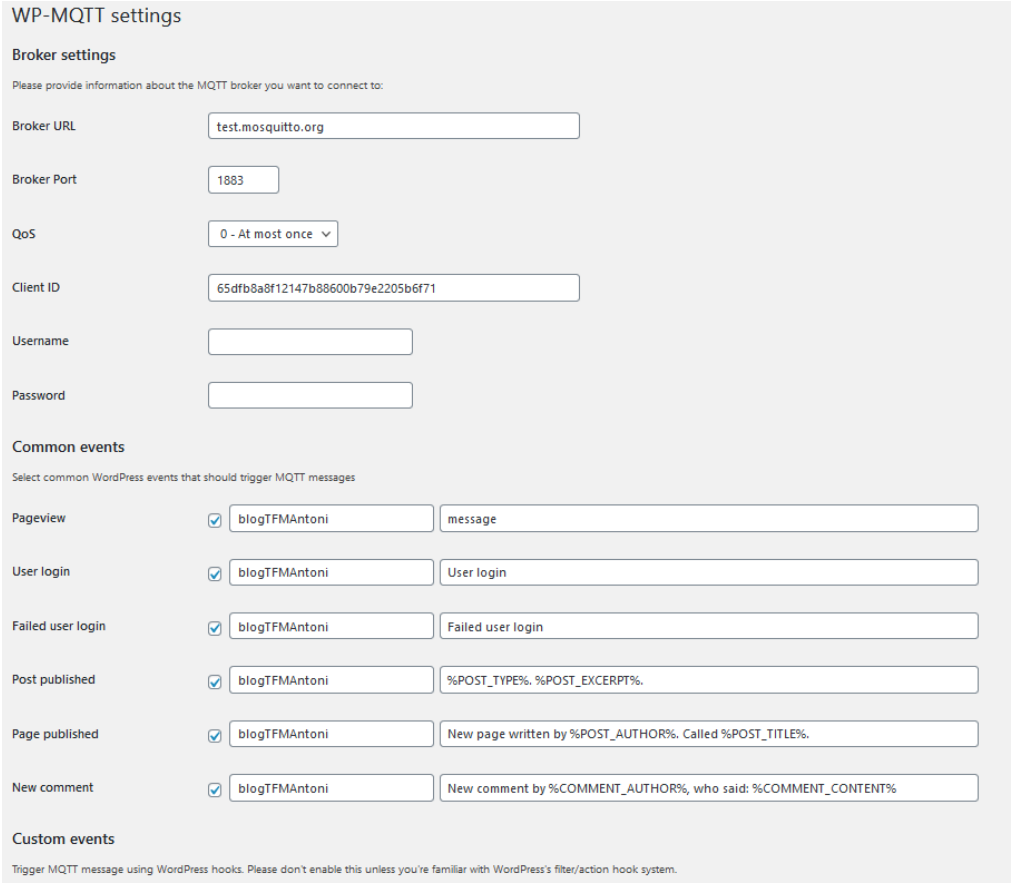


Ilustración 150. Panel de configuración de WP-MQTT

NOTA: la propiedad “ClientID” tiene que ser único, es decir, cada usuario debe utilizar una cadena distinta

Una vez hemos llegado has aquí, tendremos configurado el WordPress para que publique en un bróker MQTT. Podemos comprobar que todo esté correcto con un programa tipo “MQTT.fx” para comprobar que lleguen dichos mensajes.

Suscribirse MQTT Arduino

El código final de la implementación se puede encontrar en la siguiente ruta:

https://github.com/antonigimenezrodriguez/Arduino-IoT-33-MQTT.git

Para realizar la parte de Arduino de suscribirse y escribir en pantalla el mensaje recibido, vamos a hacer uso del IDE de escritorio de Arduino, cuyos pasos de instalación, configuración y preparación de entorno se encuentran en el Anexo I. IDE Arduino

El siguiente paso, será la instalación de las librerías necesarias para ejecutar nuestro código:

- **LiquidCrystal_I2C:**

  Esta librería vamos a descargarla del siguiente enlace:
  https://github.com/antonigimenezrodriguez/LiquidCrystal_I2C/blob/main/LiquidCrystal_I2C.zip
  Ya que se ha tenido que adaptar la librería nativa para hacerla compatible con nuestra placa.

  Una vez descargado el fichero ZIP, lo descomprimiremos en la ruta de instalación del IDE de Arduino, normalmente en: "C/Archivos de programa (x86)/Arduino/libraries"

- **Arduino MQTT Client**:

  La siguiente librería se instalará desde el panel de librerías: Menú Programa / Incluir Librería / Administrar Bibliotecas...

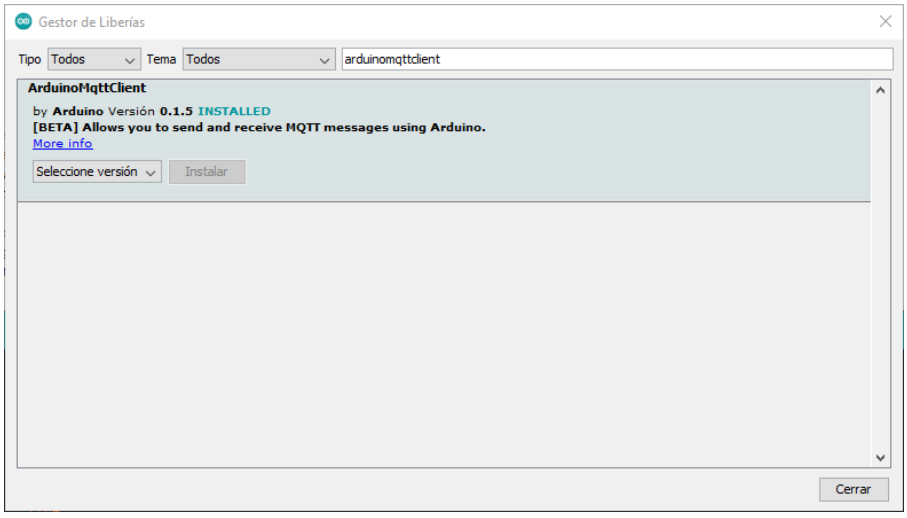


Ilustración 151. Gestor de librerías de Arduino. ArduinoMqttClient

Una vez realizado el Anexo I e instaladas las 2 librerías mencionadas anteriormente. Ya tenemos configurado el entorno del IDE de Arduino para conectarnos al bróker MQTT definido en el Blog de WordPress, suscribirnos a los tópicos configurados en el plugin WP-MQTT y escribir en el LCD el mensaje recibido.

En primer lugar, tenemos que conocer cuál es la dirección que tiene asignada nuestra pantalla, la dirección más habitual es 0x27. Para conocer la dirección de nuestra pantalla LCD, ejecutaremos el siguiente código:

```
#include <Wire.h>
void setup() {
  Serial.begin (115200);
  while (!Serial)  {    }
  Serial.println ("Scanner I2C. Scanning ...");
  byte count = 0;
  Wire.begin();
  for (byte i = 8; i < 120; i++) {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0) {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      }
  }
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");
}
void loop() {}
```

Una vez ejecutado este código, mostraremos la consola serial desde el menú Herramientas / Monitor serie. Obtendremos un resultado como el siguiente:

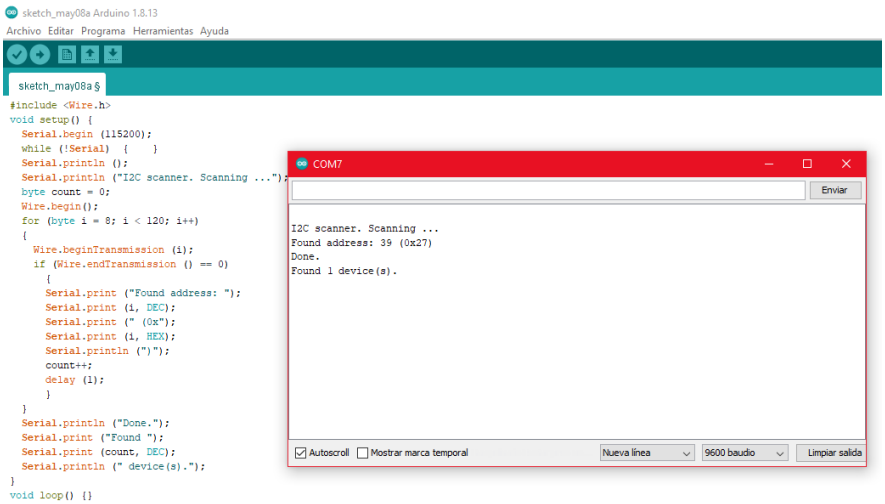


Ilustración 152. Código y resultado I2C Address Scanner

Como podemos observar, nuestra pantalla se encuentra en la dirección 0x27. Una vez sabemos cuál es la dirección de nuestra pantalla LCD, vamos a escribir el código de Arduino encargado de suscribirse a los eventos generados por WP-MQTT y escribirlos en dicha pantalla:

En la parte inicial del fichero "*.ino" vamos a definir las siguiente librerías, constantes y definición de objetos:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ArduinoMqttClient.h>
#include <WiFiNINA.h>
char ssid[] = "YourNetworkSSID";
char pass[] = "YourPassword";
Unsigned long addressLCD = 0x27; //Address LCD
LiquidCrystal_I2C lcd(addressLCD,20,4);
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);
const char broker[] = "test.mosquitto.org";
int        port     = 1883;
const char topic[]  = "blogTFMAntoni";
```

En esta primera parte del código definimos las librerías que vamos a usar mediante las directrices "#include", así como las constantes a usar en el código y la creación de los objetos del LCD y MQTT Client.

NOTA: Tendremos que sustituir el valor de las variables "ssid[]" y "pass[]" por nuestro nombre de la red WiFi y nuestra contraseña, así como la variable "topic[]" tendremos que poner exactamente el mismo nombre que hemos definido en el plugin WP-MQTT de WordPress

Continuamos con la parte del código dentro del método "setup"

```
void setup()
{
  lcd.init();
  lcd.init();
  lcd.backlight();
while (!Serial) {  }
  Serial.print("Attempting to connect to WPA SSID: ");
  Serial.println(ssid);
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    Serial.print(".");
    delay(5000);
  }
  Serial.println("You're connected to the network");
  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);
  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());
    while (1);
  }
  Serial.println("You're connected to the MQTT broker!");
  Serial.println();
  Serial.print("Subscribing to topic: ");
  Serial.println(topic);
  mqttClient.subscribe(topic);
  Serial.print("Waiting for messages on topic: ");
  Serial.println(topic);
  Serial.println();
}
```

En este código inicializamos el LCD (Código marcado en rojo), nos conectamos a la red WiFi (marcado en naranja), nos conectamos al bróker MQTT y nos suscribirnos al tópico (marcado en azul)

Finalmente, en el método “loop” vamos a escribir el siguiente código:

```
void loop()
{
  lcd.backlight();

int messageSize = mqttClient.parseMessage();
  if (messageSize) {
    Serial.print("Received a message with topic '");
    Serial.print(mqttClient.messageTopic());
    Serial.print("', length ");
    Serial.print(messageSize);
    Serial.println(" bytes:");
    int fila = 0;
    int columna = 0;
    lcd.clear();
    while (mqttClient.available()) {
      if(columna == 20){
        columna = 0;
        fila = fila +1;
      }
      if(fila == 4){
        delay(500);
        lcd.clear();
        fila = 0;
      }
      lcd.setCursor(columna, fila);
      char caracter = (char)mqttClient.read();
      lcd.print(caracter);
      Serial.print(caracter);
      columna = columna +1;
      delay(100);
    }
    Serial.println();
  }
}
```

En dicho código, esperamos a que se publique un mensaje en el tópico al que nos hemos suscrito, cuando esto pase, cogemos dicho mensaje y lo escribimos tanto en la consola serie como en el LCD.

El LCD tiene una restricción de 80 caracteres distribuidos en 4 filas de 20 columnas. En el caso que recibimos un mensaje más largo, se borra toda la pantalla y se empieza a escribir de nuevo.

Anexo IV. *Dashboard*

En este anexo explicaremos como realizar la actividad 3. Realización de un *dashboard* en las plataformas Cloud IoT "UbiDots" y "ThingSpeak".

Para poder explorar el elenco de sensores a utilizar, hemos escogido los distintos sensores existentes en Arduino según el tipo de datos de salida:

Tabla 7. Sensores actividad 3

Sensor	Tipo Sensor	Tipo Salida
DHT-11	Digital	Valores decimales, Valores positivos y negativos
HC-SR04	Digital	Funcionamiento por pulsos
Sensor de agua	Analógico	Analógico con calibración manual
ADXL-345	I2C	Lectura de datos a través del canal I2C

En primer lugar, conectaremos el Arduino, la *protoboard* y los sensores siguiendo el siguiente esquema:

Esquema hardware

Dado que el Arduino nano IoT 33 solo tiene una salida de VCC 3.3, conectaremos la patilla VCC 3.3 a la línea positiva superior de la *protoboard*. La salida VCC 5 a la línea positiva inferior y las 2 salidas GND del Arduino, una a la parte negativa superior y la otra a la parte negativa inferior. Tal y como sigue:

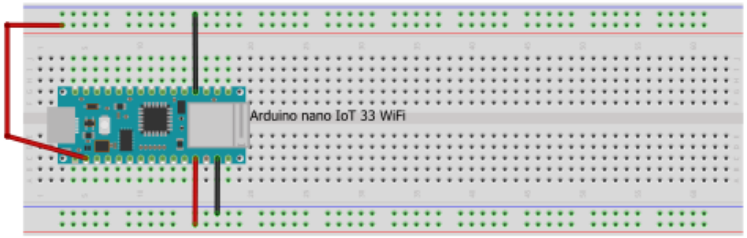


Ilustración 153. Conexión Arduino a la protoboard

Seguiremos conectando el sensor HC-SR04:

Tabla 8. Patillaje HC-SR04

VCC 5V	Línea positiva inferior
GND	Línea negativa inferior
Trigger	D8
Echo	D9

La conexión quedará como sigue en la siguiente imagen:

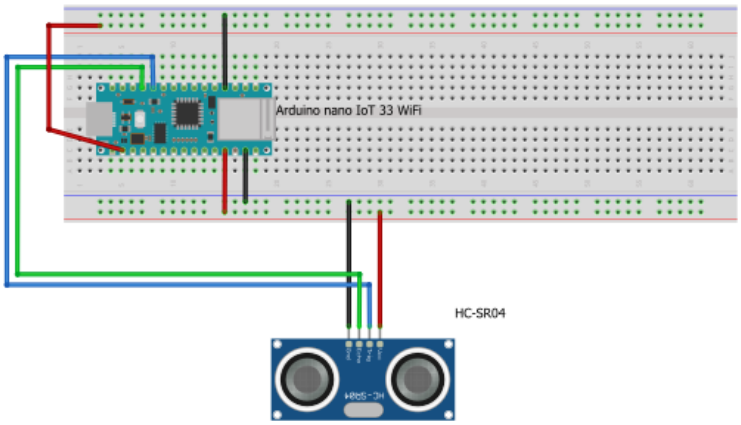


Ilustración 154. Conexión HC-SR04

El siguiente sensor para conectar será el giroscopio ADXL-345:

Tabla 9. Patillaje ADXL-345

VCC 3.3V	Línea positiva superior
GND	Línea negativa superior
SDA	A4
SCL	A5

La conexión quedará como sigue en la siguiente imagen:

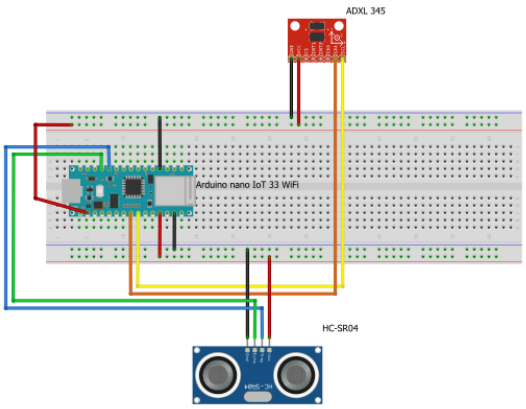


Ilustración 155. Conexión ADXL-345

Continuaremos conectando el sensor DHT-11:

Tabla 10. Patillaje DHT-11

VCC 3.3V	Línea positiva superior
GND	Línea negativa superior
DATA	D2

Cuyas conexiones quedarán:

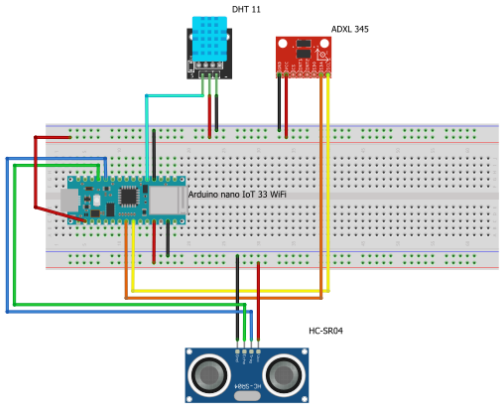


Ilustración 156. Conexión DHT-11

Y, para finalizar con el montaje de los sensores, lo haremos con el Sensor de agua:

Tabla 11. Patillaje Sensor de agua

VCC 3.3V	Línea positiva superior
GND	Línea negativa superior
DATA	A7

El esquema, una vez conectado el Sensor de agua tendremos el esquema final de la actividad:

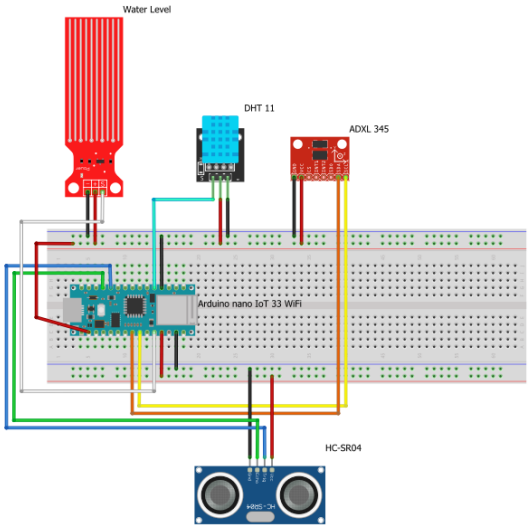


Ilustración 157. Montaje final actividad 3

Ubidots

A continuación, vamos a preparar el portal de Ubidots. Primeramente, accederemos al portal: https://ubidots.com/ y crearemos una cuenta STEM para uso educacional o personal. Una vez hayamos creado la cuenta accederemos a nuestro token, para acceder a él iremos a nuestro perfil y buscaremos la pestaña "API Credentials"

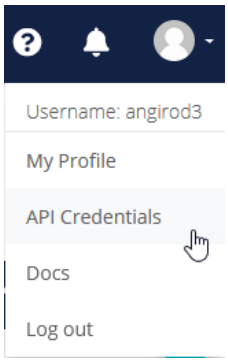


Ilustración 158. API Credentials

Al pinchar en "API Credentials" nos mostrará un nuevo cuadrado dentro de la página con nuestra API Key y nuestro token, en este caso nos interesa el token marcado como "Default token", revelamos dicho token y lo anotamos para más tarde.

ThingSpeak

En primer lugar, accederemos al portal: https://thingspeak.com/ y crearemos una cuenta. A continuación, crearemos un canal nuevo y definiremos los 8 *Fields* de la siguiente forma:

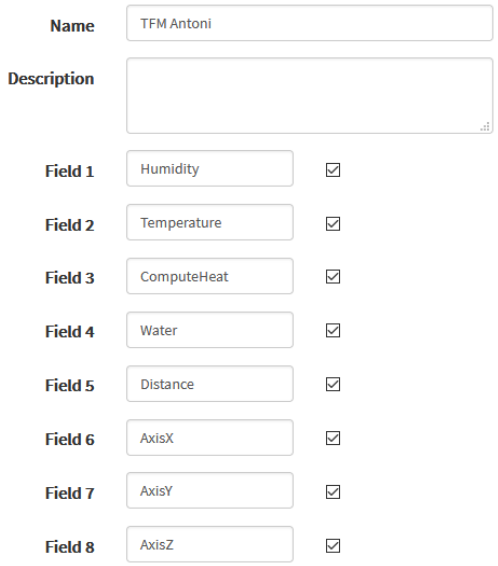


Ilustración 159. Fields Canal ThingSpeak

Para finalizar, accederemos a pestaña "API Keys" y nos anotaremos tanto el "Channel ID" como el "Write API Key" para más tarde.

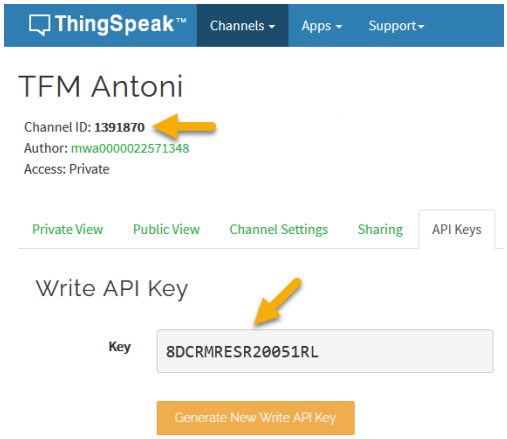


Ilustración 160. Channel ID y token ThingSpeak

Arduino

El Código final está accesible en la siguiente dirección GitHub: https://github.com/antonigimenezrodriguez/Arduino-IoT-33-Dashboard.git

Una vez tengamos conectados los sensores y tenemos las plataformas *cloud* configuradas, procederemos a realizar el código en el IDE de Arduino. Es requisito la realización previamente del Anexo I. IDE Arduino.

Seguidamente, instalaremos las librerías necesarias para esta actividad:

- **ThingSpeak**:

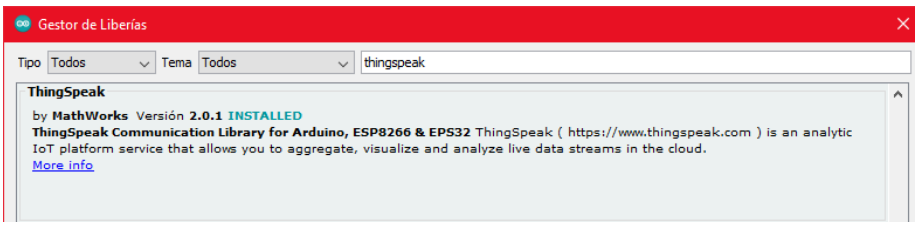


Ilustración 161. Librería ThingSpeak

- **DHT Sensor library:**

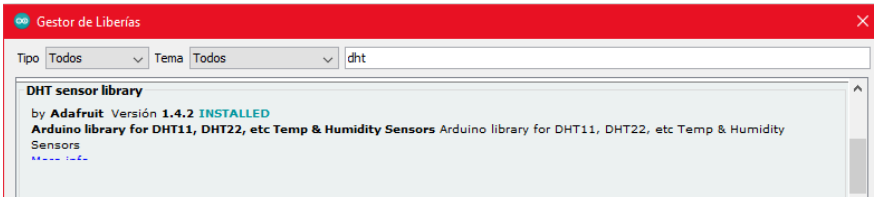


Ilustración 162. Librería DHT

- **NewPing:**

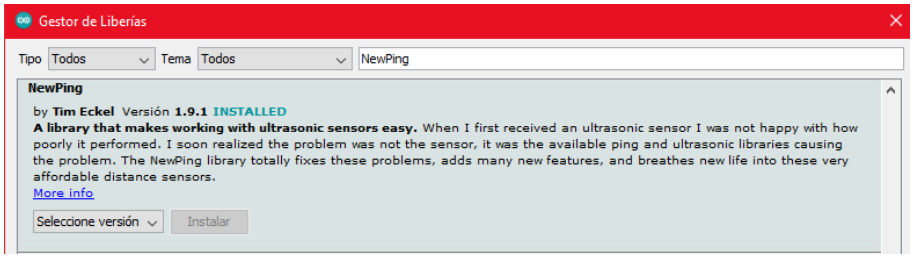


Ilustración 163. Librería NewPing

- **SparkFun ADXL345 Arduino Library:**

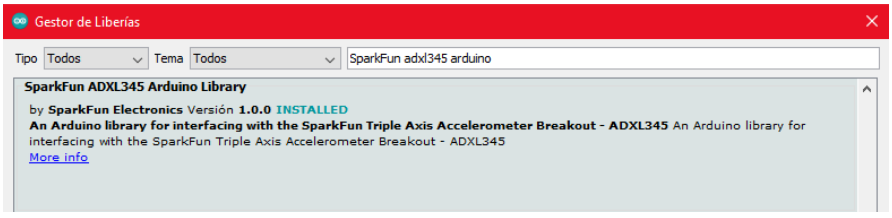


Ilustración 164. Librería SparkFun ADXL345

A continuación, crearemos 3 documentos nuevos que llamaremos "sendValuesThingSpeak.h", "sendValuesUbiDots.h" y "sensors.h".

Empezaremos con el fichero "sensors.h", dicho fichero contendrá todo lo relativo a los sensores, definición de pines, inicialización de sensores y lectura de valores.

Librerías, definiciones e inicializaciones:

```
#include <SPI.h>
#include <Wire.h>
#include <SparkFun_ADXL345.h>
#include "DHT.h"
#include <NewPing.h>

#define DHTPIN 2
#define WATERPIN A7
#define DISTTRIGPIN 8
#define DISTECHOPIN 9
#define GYROSCLPIN A5
#define GYROSDAPIN A4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
NewPing sonar(DISTTRIGPIN, DISTECHOPIN, 200);
ADXL345 adxl = ADXL345();
```

Seguiremos con la activación de los sensores:

```
void startSensors() {
  Serial.begin(9600);
  Serial.println("Iniciando Sensores");
  Serial.println();
  dht.begin();
  adxl.powerOn();
  adxl.setRangeSetting(16);       //Definir el rango, valores 2, 4, 8 o 16
  Serial.println("Sensores iniciados correctamente");
}
```

Para finalizar con este fichero, definiremos las funciones para realizar las lecturas de los sensores:

```
void getDHTValues(float &hum, float &tem, float &comHeat ) {
  hum = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.println(hum);
  // Read temperature as Celsius (the default)
  tem = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.println(tem);
  // Compute heat index in Celsius (isFahreheit = false)
  comHeat = dht.computeHeatIndex(tem, hum, false);
  Serial.print("Compute Heat: ");
  Serial.println(comHeat);
}

float getWater() {
  int water = analogRead(WATERPIN);
  float waterCalc = map(water, 0, 550, 0, 100);
  Serial.print("Water: ");
  Serial.println(waterCalc);
  return waterCalc;
}

unsigned int getDistance() {
  unsigned int cm = sonar.ping_cm();
  Serial.print("Distancia: ");
```

```
  Serial.print(cm);
  Serial.println("cm");

  if (cm == 0 && digitalRead(DISTECHOPIN) == LOW) {
    pinMode(DISTECHOPIN, OUTPUT);
    digitalWrite(DISTECHOPIN, LOW);
    delay(100);
    pinMode(DISTECHOPIN, INPUT);
  }
  return cm;
}

void getGyroscopeValues(int &x, int &y, int &z) {
  adxl.readAccel(&x, &y, &z);
  Serial.print("Xa= ");
  Serial.print(x);
  Serial.print("   Ya= ");
  Serial.print(y);
  Serial.print("   Za= ");
  Serial.println(z);
}
```

El siguiente fichero será “sendValuesUbiDots.h”:

Librerías, definiciones e inicializaciones:

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <avr/dtostrf.h>

#define DEVICE_LABEL "arduino-nano-33"
#define TOKEN "TOKEN_UBI_DOTS"

char const * VARIABLE_LABEL_1 = "sensor";
char const *SERVER = "industrial.api.ubidots.com";
const int HTTPPORT = 443;
char const *AGENT = "Arduino Nano 33 IoT";
char const *HTTP_VERSION = " HTTP/1.1\r\n";
char const *VERSION = "1.0";
char const *PATH = "/api/v1.6/devices/";

char const * SSID_NAME = "SSID_WIFI"; // Put here your SSID name
char const * SSID_PASS = "PASS_WIFI"; // Put here your password

int status = WL_IDLE_STATUS;
WiFiSSLClient sslClient;
```

NOTA: Tenemos que sustituir el contenido de las variables “TOKEN” por el token que hemos anotado en el principio de este anexo, “SSID_NAME” por el nombre de nuestra red WiFi y “SSID_PASS” por la contraseña de nuestra WiFi.

Continuaremos con las siguientes funciones auxiliares pero necesarias:

```
void printWiFiStatus() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
void getResponseServer() {
  Serial.println(F("\nUbidots' Server response:\n"));
  while (sslClient.available()) {
    char ssl = sslClient.read();
    Serial.print(ssl);
  }
}

void waitServer() {
  int timeout = 0;
  while (!sslClient.available() && timeout < 5000) {
    timeout++;
    if (timeout >= 5000) {
      Serial.println(F("Error, max timeout reached"));
      break;
    }
  }
}

void sendData(char* payload) {
  int contentLength = strlen(payload);
  if (sslClient.connect(SERVER, HTTPPORT)) {
    Serial.println("connected to server");
    sslClient.print(F("POST "));
    sslClient.print(PATH);
    sslClient.print(DEVICE_LABEL);
    sslClient.print(F("/"));
    sslClient.print(HTTP_VERSION);
    sslClient.print(F("Host: "));
    sslClient.print(SERVER);
    sslClient.print(F("\r\n"));
    sslClient.print(F("User-Agent: "));
    sslClient.print(AGENT);
    sslClient.print(F("\r\n"));
    sslClient.print(F("X-Auth-Token: "));
    sslClient.print(TOKEN);
    sslClient.print(F("\r\n"));
    sslClient.print(F("Connection: close\r\n"));
    sslClient.print(F("Content-Type: application/json\r\n"));
    sslClient.print(F("Content-Length: "));
    sslClient.print(contentLength);
    sslClient.print(F("\r\n\r\n"));
    sslClient.print(payload);
    sslClient.print(F("\r\n"));
    Serial.print(F("POST "));
    Serial.print(PATH);
    Serial.print(DEVICE_LABEL);
    Serial.print(F("/"));
```

```
    Serial.print(HTTP_VERSION);
    Serial.print(F("Host: "));
    Serial.print(SERVER);
    Serial.print(F("\r\n"));
    Serial.print(F("User-Agent: "));
    Serial.print(AGENT);
    Serial.print(F("\r\n"));
    Serial.print(F("X-Auth-Token: "));
    Serial.print(TOKEN);
    Serial.print(F("\r\n"));
    Serial.print(F("Connection: close\r\n"));
    Serial.print(F("Content-Type: application/json\r\n"));
    Serial.print(F("Content-Length: "));
    Serial.print(contentLength);
    Serial.print(F("\r\n\r\n"));
    Serial.print(payload);
    Serial.print(F("\r\n"));
    waitServer();
    getResponseServer();
  }
  sslClient.stop();
}
```

Finalizaremos este fichero con las funciones principales:

```
void startUbiDots() {
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    while (true);
  }

  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SSID_NAME);
    status = WiFi.begin(SSID_NAME, SSID_PASS);
    delay(1000);
  }
  Serial.println("Connected to wifi");
  printWiFiStatus();
}

void sendValuesToUbiDots(float humidity, float temperature, float computeHeat, float water, int
distance, float X_out, float Y_out, float Z_out) {

  char payload[200];
  char str_val_humidity[30];
  char str_val_temperature[30];
  char str_val_computeHeat[30];
  char str_val_water[30];
  char str_val_distance[30];
  char str_val_xOut[30];
  char str_val_yOut[30];
  char str_val_zOut[30];

  /*4 is the total lenght of number,maximun number accepted is 99.99*/
  dtostrf(humidity, 4, 2, str_val_humidity);
  dtostrf(temperature, 4, 2, str_val_temperature);
  dtostrf(computeHeat, 4, 2, str_val_computeHeat);
  dtostrf(water, 4, 2, str_val_water);
  dtostrf(distance, 4, 2, str_val_distance);
  dtostrf(X_out, 5, 2, str_val_xOut);
  dtostrf(Y_out, 5, 2, str_val_yOut);
  dtostrf(Z_out, 5, 2, str_val_zOut);
  sprintf(payload, "%s", "");
```

```
  sprintf(payload, "{\"");
  sprintf(payload, "%s%s\":%s", payload, "humidity", str_val_humidity);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "temperature", str_val_temperature);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "computeHeat", str_val_computeHeat);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "water", str_val_water);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "distance", str_val_distance);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "gyroscopeX", str_val_xOut);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "gyroscopeY", str_val_yOut);
  sprintf(payload, "%s,\"", payload);
  sprintf(payload, "%s%s\":%s", payload, "gyroscopeZ", str_val_zOut);
  sprintf(payload, "%s}", payload);
  Serial.println("-------------------------------------------------");
  Serial.print("Payload :");
  Serial.println(payload);
  Serial.println("-------------------------------------------------");
  sendData(payload);
}
```

El siguiente fichero será: “sendValuesThingSpeak.h”:

Librerías, definiciones e inicializaciones:

```
#include <WiFiNINA.h>
#include "ThingSpeak.h"

#define HUMIDITY_CHART 1
#define TEMPERATURE_CHART 2
#define COMPUTE_HEAT_CHART 3
#define WATER_CHART 4
#define DISTANCE_CHART 5
#define GYROSCOPE_AXIS_X_CHART 6
#define GYROSCOPE_AXIS_Y_CHART 7
#define GYROSCOPE_AXIS_Z_CHART 8

char ssid[] = "SSID_NAME";   // your network SSID (name)
char pass[] = "SSID_PASS";   // your network password
WiFiClient  client;

unsigned long myChannelNumber = CHANNEL_NUMBER;
const char * myWriteAPIKey = "API_KEY";
```

NOTA: Tenemos que sustituir el contenido de las variables “API_KEY” por el “Write API Key”, “CANNEL_NUMBER” por el número del canal que hemos anotado en el principio de este anexo, “SSID_NAME” por el nombre de nuestra red WiFi y “SSID_PASS” por la contraseña de nuestra WiFi.

Y continuaremos con las funciones principales:

```
void startThingSepak() {
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  ThingSpeak.begin(client);  //Initialize ThingSpeak
}

void sentValueToThingSpeak(float humidity, float temperature, float computeHeat,
float water, int distance, float X_out, float Y_out, float Z_out) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);
      Serial.print(".");
      delay(1000);
    }
    Serial.println("\nConnected.");
  }

  // set the fields with the values
  ThingSpeak.setField(HUMIDITY_CHART, humidity);
  ThingSpeak.setField(TEMPERATURE_CHART, temperature);
  ThingSpeak.setField(COMPUTE_HEAT_CHART, computeHeat);
  ThingSpeak.setField(WATER_CHART, water);
  ThingSpeak.setField(DISTANCE_CHART, distance);
  ThingSpeak.setField(GYROSCOPE_AXIS_X_CHART, X_out);
  ThingSpeak.setField(GYROSCOPE_AXIS_Y_CHART, Y_out);
  ThingSpeak.setField(GYROSCOPE_AXIS_Z_CHART, Z_out);


  // set the status
  ThingSpeak.setStatus("status");
  // write to the ThingSpeak channel
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if (x == 200) {
    Serial.println("Channel update successful.");
  }
  else {
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }
}
```

Finalizaremos con el fichero principal de nuestro proyecto:

Librerías, definiciones e inicializaciones:

```
#include "sensors.h"
#include "sendValuesThingSpeak.h"
#include "sendValuesUbiDots.h"
```

En este caso tan solo referenciaremos los 3 ficheros que hemos creado anteriormente.

El método *setup* tendrá el siguiente contenido:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!Serial) {}
  startSensors();
  startThingSepak();
  startUbiDots();
  delay(10);
}
```

Donde llamaremos a las funciones definidas en los ficheros para sus respectivas inicializaciones.

Y el método *loop*:

```
void loop() {
  // put your main code here, to run repeatedly:

  float h = 0.00;
  float t = 0.00;
  float hic = 0.00;
  getDHTValues(h,t,hic);

  float water = getWater();

  unsigned int cm = getDistance();

  int X_out = 0;
  int Y_out = 0;
  int Z_out = 0;
  getGyroscopeValues(X_out, Y_out,Z_out);

  Serial.println();
  sentValueToThingSpeak(h, t, hic, water, cm, X_out, Y_out, Z_out);
  sendValuesToUbiDots(h, t, hic, water, cm, X_out, Y_out, Z_out);
  delay(2000);
}
```

En este método, llamaremos a las funciones de obtención de valores de los distintos sensores, y una vez obtenidos esos valores, llamaremos a las funciones "sentValueToThingSpeak" y "sendValuesToUbiDots", cuyos parámetros son los valores recogidos anteriormente y la función de éstos es la de mandar la información a las 2 *Clouds*.

NOTA: La *Cloud* de ThingSpeak, en su versión gratuita, tan solo nos permite insertar valores en intervalos de 15 segundos, con lo que sería muy recomendable, que en la última espera ("delay(2000)") fuese de 15 segundos, es decir, "delay(15000)". Con esto conseguiremos que los dos *dashboards* que vamos a construir tengan exactamente los mismos datos.

Dashboards

El último paso de este anexo será la confección de un *dashboard* en las plataformas *Cloud* que hemos visto. Como hemos podido observar con anterioridad, las dos plataformas tienen ciertas limitaciones en versiones gratuitas. Por una parte, ThingSpeak nos limita a un máximo de 8 entradas de valores y la recogida de datos cada 15 segundos. En cambio, en Ubidots no tenemos límite de variables a crear, nos permite insertar datos cada segundo, pero la gran limitación reside en que tan solo podemos insertar 4.000 datos por día.

Si contamos con 8 variables, como es nuestro sistema, podemos hacer 500 inserciones de datos por día, aproximadamente 20 inserciones por hora o una inserción de datos cada 3 minutos.

ThingSpeak

Para crear el dashboard en esta *Cloud*, tendremos que acceder al canal que hemos creado anteriormente y navegar hasta la pestaña "Private View":

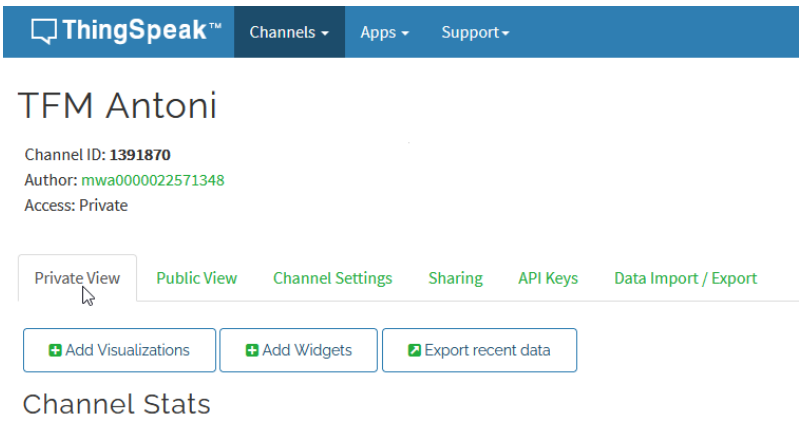


Ilustración 165. Private View ThingSpeak

Una vez situados en dicha pestaña, pulsaremos sobre el botón "Add Visualizations":

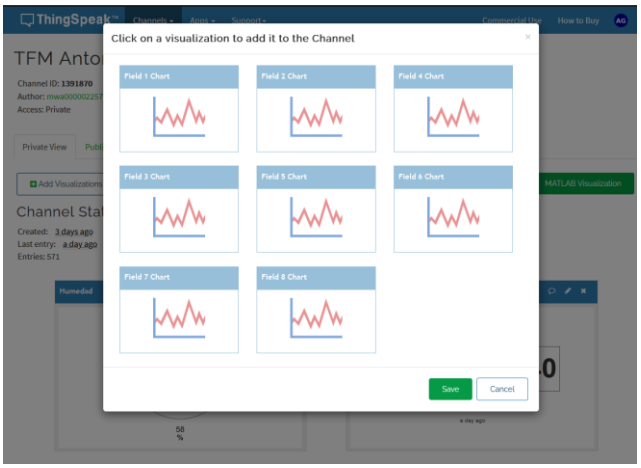


Ilustración 166. Añadir gráficas ThingSpeak

En la siguiente ventana seleccionaremos las variables que queramos mostrar y pulsaremos el botón “Save”.

Ilustración 167. Gráficas ThingSpeak

A continuación, podemos añadir nuevas visualizaciones pulsando el botón “Add Widget” y tendremos la posibilidad de añadir 3 diferentes visualizaciones:

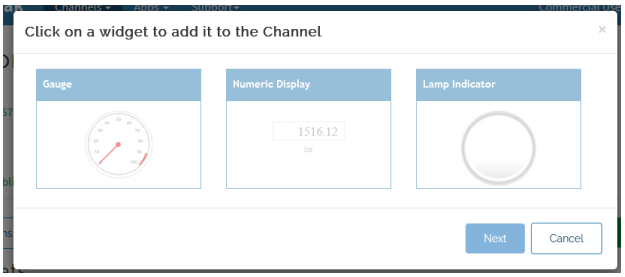


Ilustración 168. Widgets ThingSpeak

Gauge: Vamos a configurar un *widget* de tipo “Gauge” para medir la humedad, lo seleccionaremos y pulsaremos en el botón “Next”, nos mostrará una ventana para configurarlo:

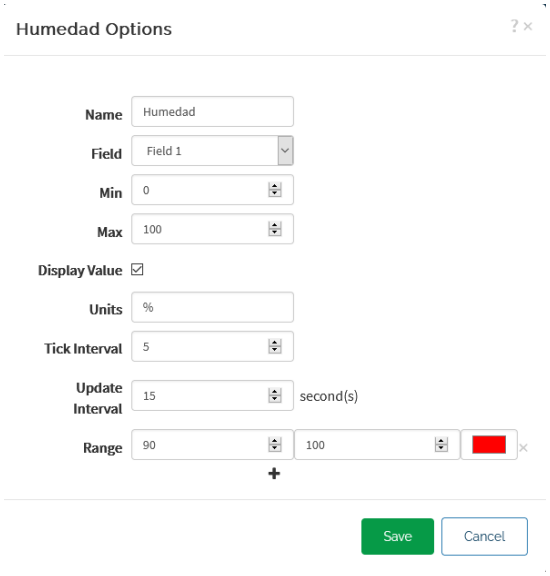


Ilustración 169. Configuración Gauge ThingSpeak

En la ilustración podemos observar que hemos configurado el *Widget* para que muestre los valores de “Field 1” (valores de humedad). En las propiedades “Min.” Y “Max.” definimos cuáles serán nuestros valores máximo y mínimo, en el caso del sensor de la humedad nos devuelve un valor entre 0 y 100.

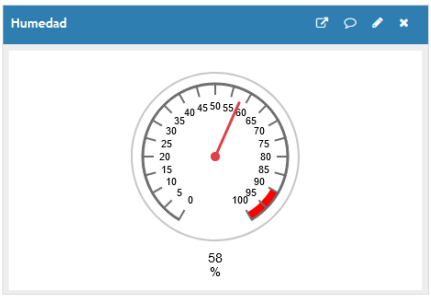


Ilustración 170. Resultado Gauge ThingSpeak

Numeric display: En este *widget* vamos a mostrar la temperatura, para ello, crearemos un *widget* de tipo numérico con la siguiente configuración:

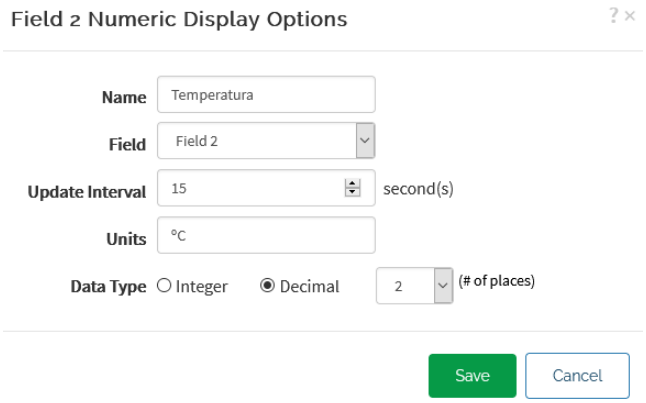


Ilustración 171. Configuración Numeric Display ThingSpeak

Como podemos observar, sincronizamos el valor del "Field 2" (temperatura) con dicho *widget*, marcamos como unidades ºC y configuramos para que nos muestre 2 decimales

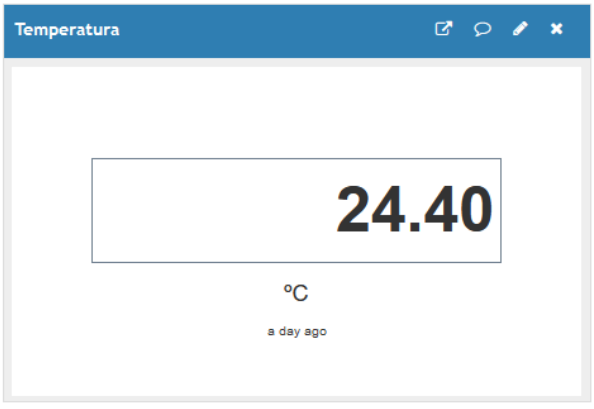


Ilustración 172. Resultado Numeric Display ThingSpeak

Lamp Indicator: Este *widget* no es más que una luz cuyos valores son encendida o apagada según una condición, que hemos definido para que se encienda una luz roja cuando el nivel de agua es superior al 50%:

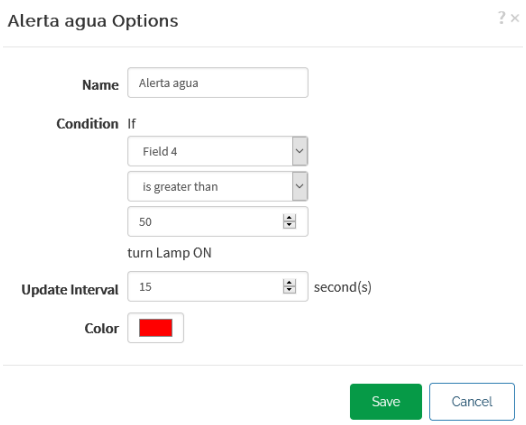


Ilustración 173. Configuración Lamp Indicator ThingSpeak

La configuración que podemos observar es la sincronización con el "Field 4" (Agua) y se activará cuando es mayor que (is greather tan) y el siguiente valor, en nuestro caso, 50.

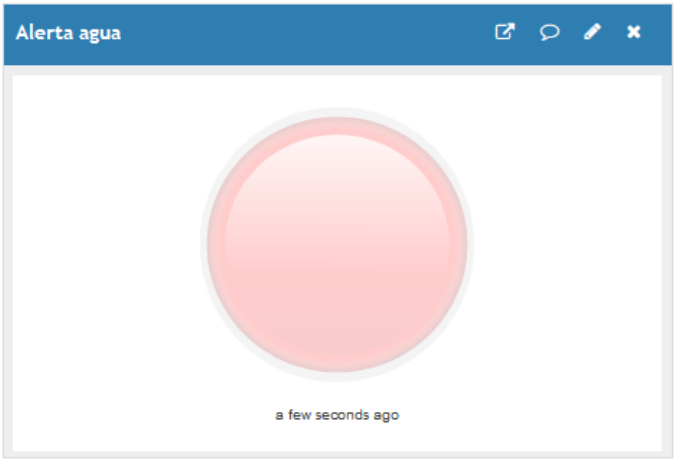


Ilustración 174. Resultado Lamp Indicator ThingSpeak

Ubidots

En Ubidots, a diferencia de ThingSpeak, no tenemos crear las variables previamente, tanto el dispositivo como las variables, se crearán con el primer envío de datos al servidor. Por tanto, la primera vez que accedamos al portal, no tendremos ningún dispositivo conectado ni ninguna variable:

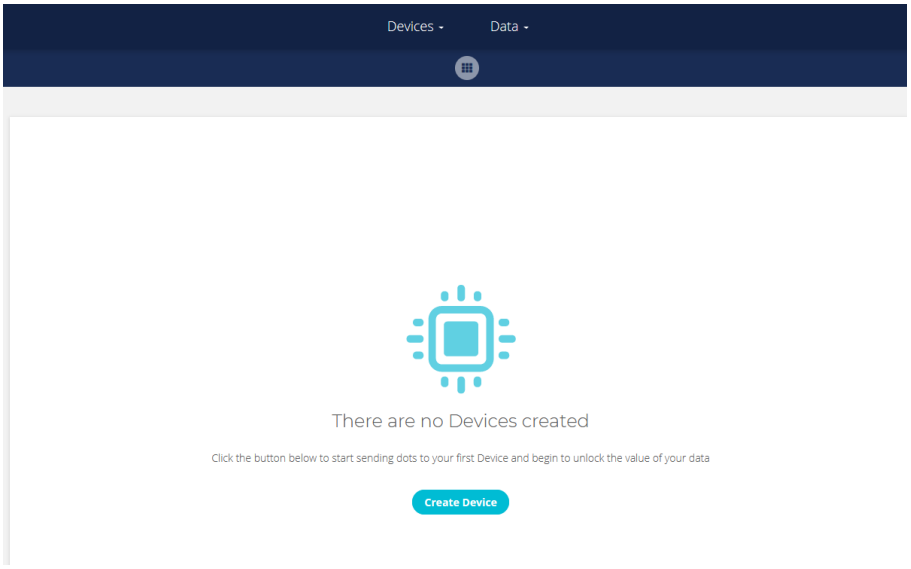


Ilustración 175. Panel Ubidots vacío

Una vez hayamos realizado el primer envío de datos, usando el código mencionado anteriormente, podremos observar que nuestro dispositivo se ha dado de alta, y con él, las 8 variables que hemos definido para recoger los datos de los sensores:

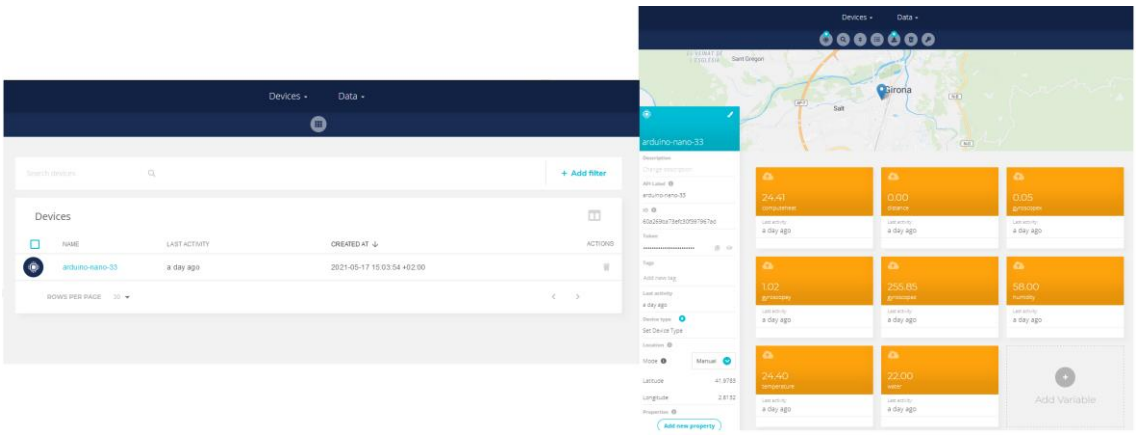


Ilustración 176. Portal Ubidots con el dispositivo y las variables

Una vez tengamos creados correctamente las variables en el portal, iremos a la pestaña "Data / Dashboards" para empezar a crear nuestro *Dashboard*.

Las distintas opciones de *widgets* que nos ofrece Ubidots son las siguientes:

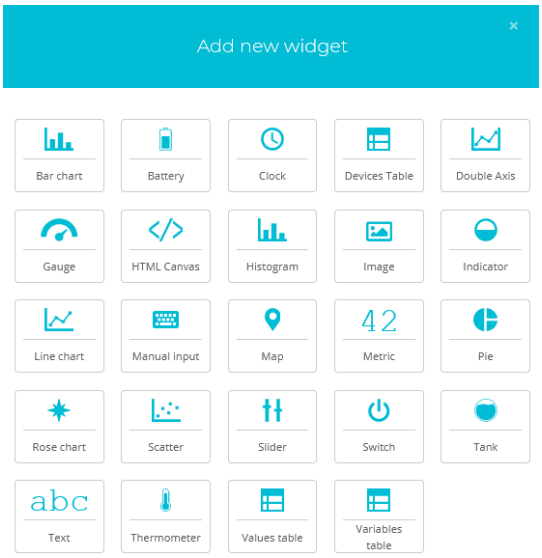


Ilustración 177. Widgets Ubidots

En primer lugar, vamos a crear uno de tipo "Line Chart". Seleccionaremos el botón y nos cambiará a la ventana de configuración del *widget*, que lo realizaremos de la siguiente forma:

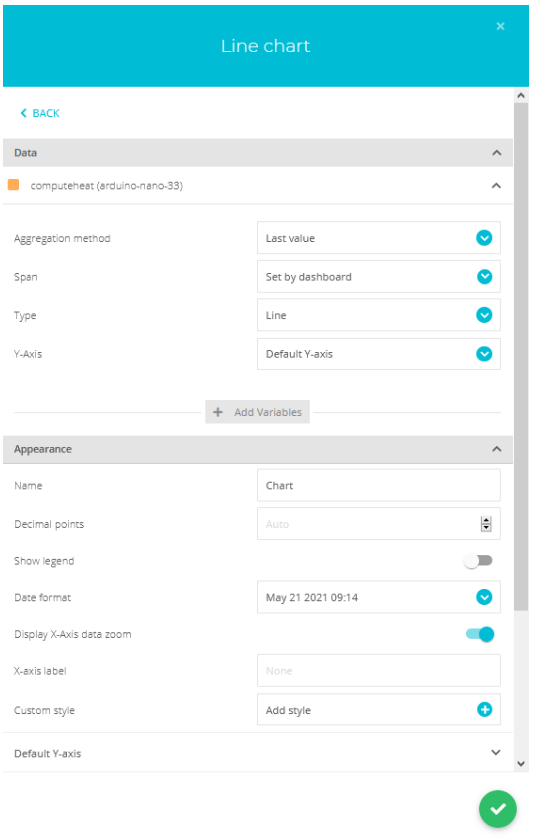


Ilustración 178. Configuración Line Chart

Sincronizaremos dicho *widget* con la variable "computeheat" del Arduino, las opciones que siguen las dejaremos por defecto. Obtendremos el siguiente resultado:

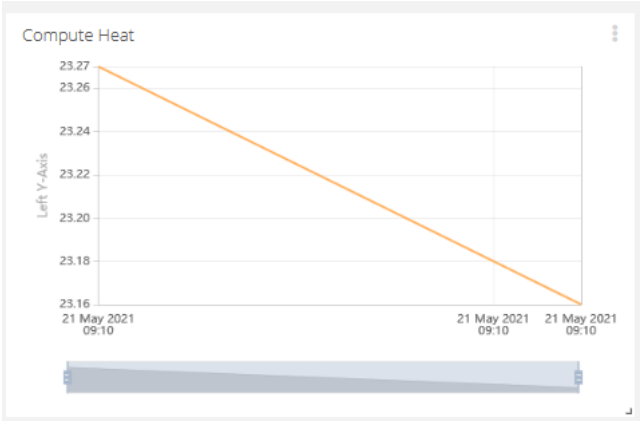


Ilustración 179. Resultado Line Chart Ubidots

Continuaremos con la creación de un gráfico de tipo *slider* para los 3 valores del giroscopio. Lo configuraremos tal y como sigue:

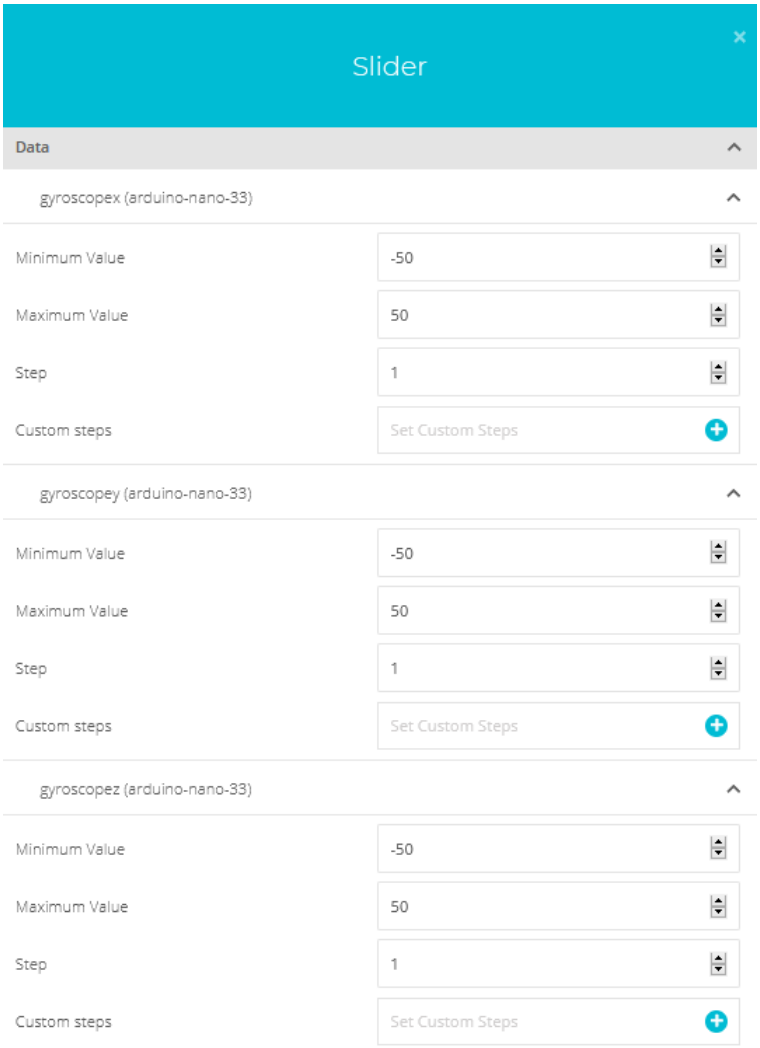


Ilustración 180. Configuración Slider Ubidots

Vincularemos las 3 variables del giroscopio (“gyroscopex”, “gyroscopey” y “gireoscopez”) al gráfico, configuraremos el valor mínimo y el valor máximo.

La configuración de los valores mínimo y máximo dependerán del rango que hemos elegido en la inicialización del sensor, las recomendaciones son:

Tabla 12. Valores máximo y mínimo ADXL-345

Rango (fuerza g ejercida)	Valor mínimo	Valor máximo
2	-255	255
4	-150	150
8	-100	100
16	-50	50

Una vez configurado, obtendremos el siguiente resultado:

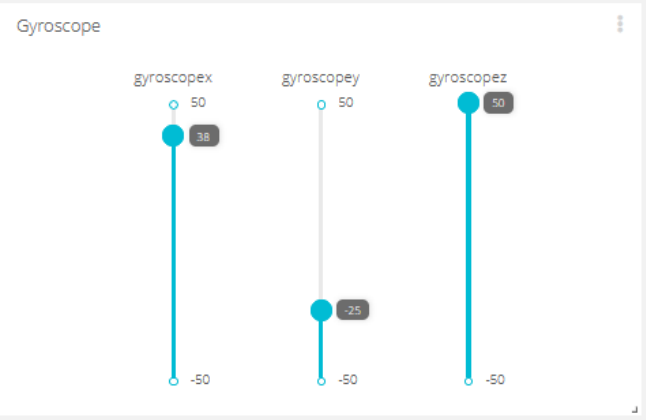


Ilustración 181. Resultado Slider Ubidots

Seguidamente, configuraremos el siguiente *widget* de tipo *Gauge* vinculado a la variable de humedad:

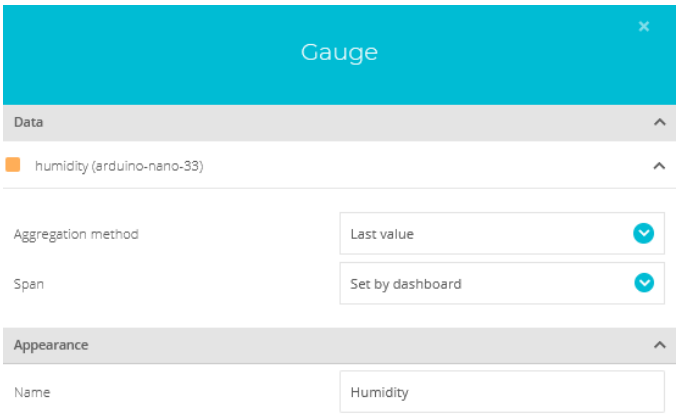


Ilustración 182. Configuración Gauge Ubidots

Cuyo resultado final será:

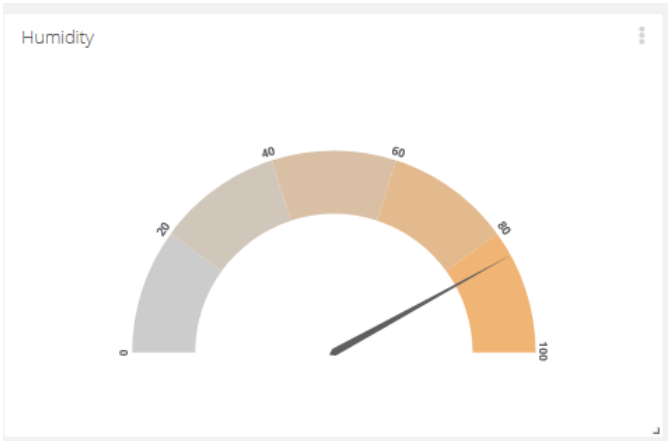


Ilustración 183. Resultado final Gauge Ubidots

El siguiente widget que insertaremos será de tipo *Tank*, lo vincularemos a la variable "water" y lo configuraremos de forma que el color azul aumente de intensidad cuando mayor sea el porcentaje de agua:

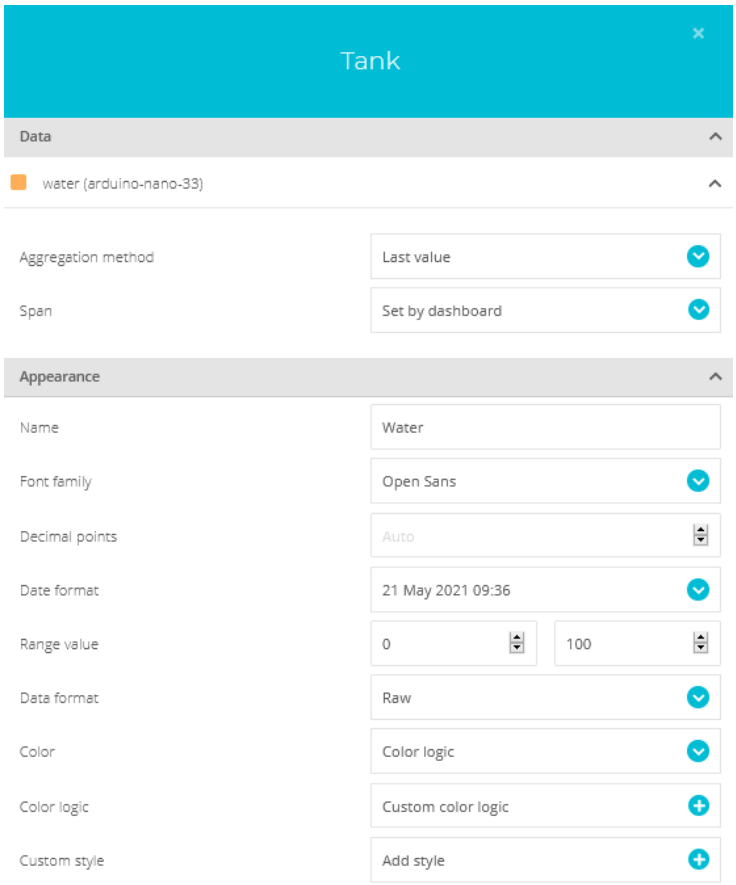


Ilustración 184. Configuración básica Tank Ubidots

Para realizar la configuración cromática, seleccionaremos "Color logic" en el desplegable de la opción "color". Al seleccionar dicha opción, aparecerá un nuevo

campo llamado “Color logic”, aquí pulsaremos el botón “+” y lo configuraremos de la siguiente forma:

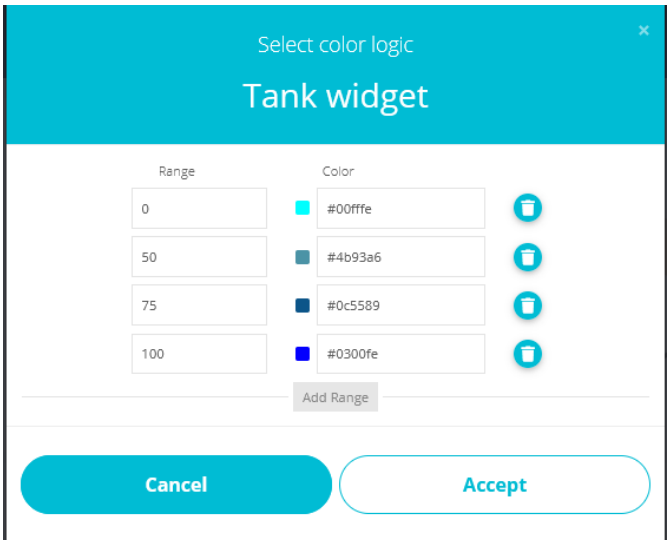


Ilustración 185. Configuración cromática Tank Ubidots

Podemos añadir tantos puntos como queramos, hemos decidido dividirlo en los rangos de 0 a 50, de 51 a 75 y de 76 a 100. Después de realizar estas configuraciones, el resultado final será:

Ilustración 186. Resultado cromático Tank Ubidots

Continuaremos configurando el *widget* de tipo *Thermometer* asociado a la variable temperatura:

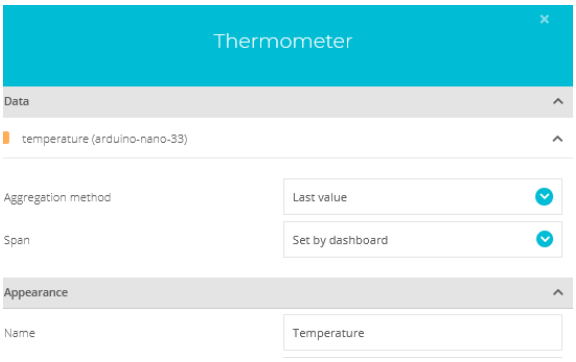


Ilustración 187. Configuración Thermometer Ubidots

Al igual que el *widget* de tipo *tank*, nos permite realizar una configuración cromática de colores, un ejemplo podría ser los típicos mapas de temperatura que podemos ver en los telediarios:

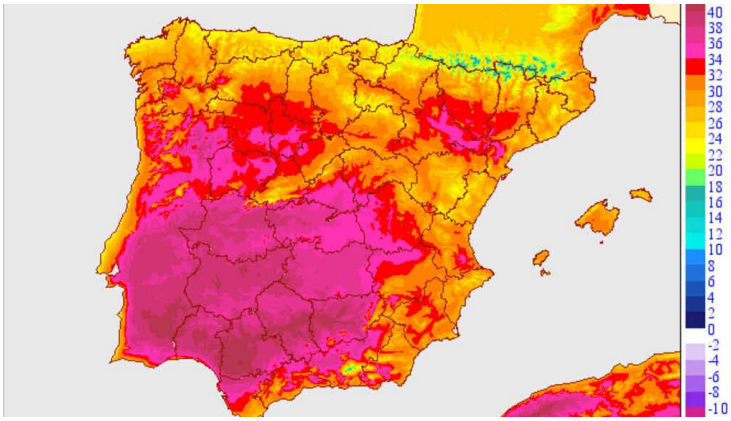


Ilustración 188. Posible ejemplo cromático temperatura

El resultado final de nuestro *widget* será:

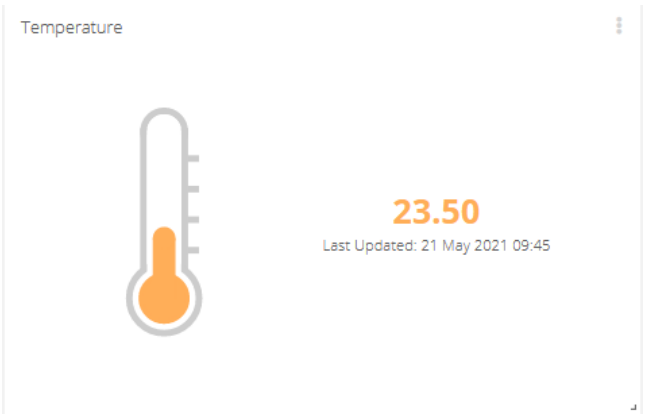


Ilustración 189. Resultado final Thermometer Ubidots

Finalizaremos con el *widget* de tipo *metric* que lo asociaremos al valor de la distancia y cuya configuración será la siguiente:

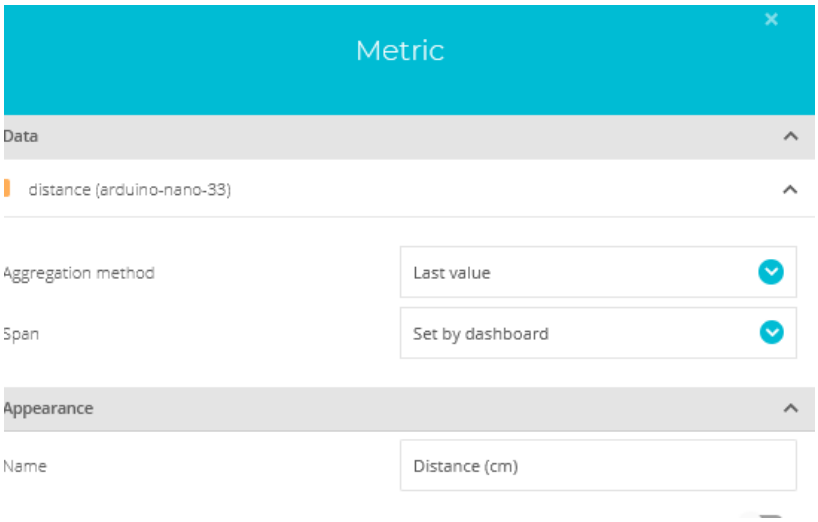


Ilustración 190. Configuración metric Ubidots

El resultado se corresponde con el que observamos a continuación:

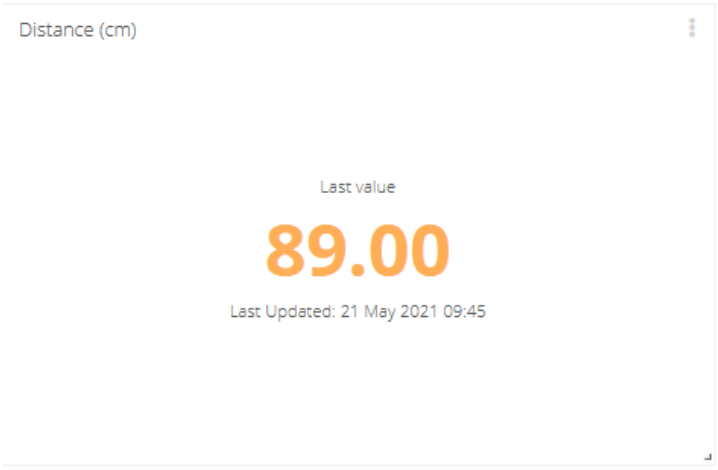


Ilustración 191. Resultado Metric Ubidots

Una vez configuradas todas las variables con sus *Wigdets*, podremos observar el resultado final de nuestro *dashboard*:

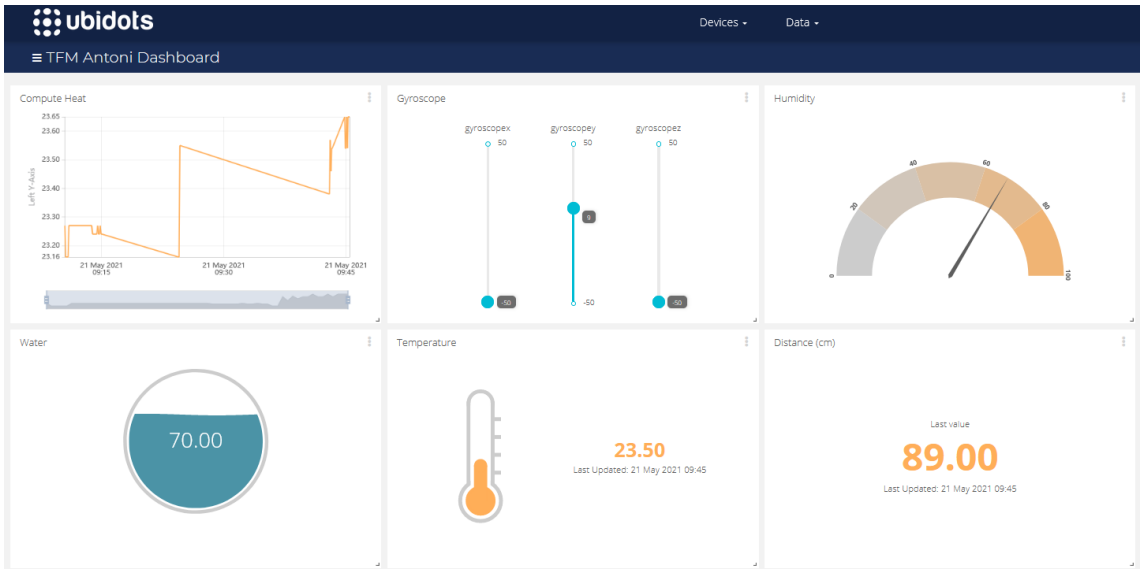


Ilustración 192. Resultado final Dashboard Ubidots

Anexo V. Privacidad de nuestra aplicación

Política de privacidad de nuestra APP

En School Gardens for Future Citizens (“eSGarden”) respetamos su privacidad. Esta Política de privacidad ("Política") explica las prácticas de privacidad para la aplicación móvil ("Aplicación").

DATOS PERSONALES QUE TRATAMOS

- No recopilamos ninguna información personal que pueda identificarlo directamente. Cuando utiliza la aplicación, no le pedimos que proporcione ninguna información personal.
- Utilizamos herramientas de terceros para gráficos, pero no para recopilar ningún tipo de información personal. Usamos Google Analytics para recopilar información analítica sobre su uso de la Aplicación. Esta herramienta de análisis recopila su dirección IP, tipo de dispositivo, uso del idioma, duración de la sesión, contenido al que accedió en la Aplicación, clics en la interfaz de usuario, frecuencia y alcance de uso de la Aplicación.

CÓMO PROCESAMOS Y UTILIZAMOS LOS DATOS PERSONALES

- Procesamos la Información analítica agregada para comprender cómo los usuarios interactúan con la Aplicación para que podamos desarrollarla y mejorarla.

CUANDO SUS DATOS PERSONALES SE COMPARTEN CON OTROS

- No vendemos su información personal a terceros.
- No compartiremos su información con terceros, excepto en los eventos que se enumeran a continuación o cuando nos brinde su consentimiento explícito e informado.
- Compartiremos la información de análisis con el proveedor de servicios de Google, que nos ayuda con el funcionamiento interno de la aplicación. Esta empresa está autorizada a utilizar la información recopilada solo cuando sea necesario para proporcionarnos estos servicios y no para sus propios fines.

PRIVACIDAD DE LOS NIÑOS

- La Aplicación está destinada a niños menores de 8 años. No recopilamos directamente ninguna información personal de niños menores de 8 años.
- La información de análisis que recopilamos a través de herramientas de terceros solo se utiliza para respaldar el funcionamiento interno de la Aplicación, es decir, para mantener y desarrollar la aplicación, y para ofrecer publicidad contextual adecuada para niños.
- La información de análisis que procesamos es información agregada que nosotros mismos no podemos atribuir a un usuario específico.

RETENCIÓN DE DATOS

- No conservamos ningún dato personal que nosotros mismos podamos atribuir a un usuario específico. Solo conservamos información analítica agregada que no se puede atribuir a un usuario específico.

COOKIES

- No utilizamos cookies en la aplicación. Las cookies de terceros para análisis nos ayudan a comprender cómo los usuarios interactúan con la Aplicación, mediante la recopilación de datos que no lo identifican directamente.

INFORMACIÓN ADICIONAL PARA USUARIOS DE LA UE

- La base legal bajo la ley de la UE para procesar la Información de análisis para desarrollar y mejorar la Aplicación son nuestros intereses legítimos en comprender cómo los usuarios usan la Aplicación y tomar decisiones comerciales sobre el desarrollo posterior de la Aplicación.
- La información analítica que recopilamos se procesará en la UE. Si transferimos su información analítica de la UE a otras jurisdicciones, lo haremos utilizando las salvaguardas adecuadas determinadas por la Comisión de la UE.
- El GDPR otorga al interesado ciertos derechos para acceder, actualizar o eliminar su información, obtener una copia de su información, retirar el consentimiento y objetar o restringir el procesamiento de datos. Sin embargo, dado que solo procesamos información analítica agregada que no se puede atribuir a un usuario específico, estos derechos no se aplican a usted en su uso de la Aplicación.
- Sujeto a la ley aplicable, tiene derecho a presentar una queja ante la autoridad local de protección de datos. Si se encuentra en la UE, de acuerdo con el artículo 77 del RGPD, puede presentar una queja ante la autoridad de control, en particular en el Estado miembro de su residencia, lugar de trabajo o de una presunta infracción del RGPD. Para obtener una lista de las autoridades de supervisión de la UE, acceda a la siguiente dirección: http://ec.europa.eu/newsroom/article29/document.cfm?action=display&doc_id=50061

CAMBIOS A ESTA POLÍTICA DE PRIVACIDAD

- De vez en cuando, podemos cambiar esta Política, en cuyo caso le informaremos de la Política actualizada a través de la Aplicación. La versión actualizada de la Política siempre estará disponible en la Aplicación.

CONTÁCTENOS

- Si tiene alguna consulta, queja o sugerencia, puede contactarnos en esgardenerasmus@gmail.com. Haremos todo lo posible para resolver su problema de manera oportuna.

Última actualización: 11 de diciembre de 2020