MOOC IOT

Módulo 3. INTERFACES DE USUARIO Y PLATAFORMAS DE DESARROLLO

3.1 INTERFACES DE USUARIO

Por Alberto Guillén Perales

Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada.

---

1. Introducción

De una manera informal, es posible definir las interfaces de usuario (UI, User Interfaces) como aquellas que nos permiten interactuar con los dispositivos que componen la infraestructura IoT para supervisarlos, monitorizar el estado y modificar su comportamiento. En otras palabras, son los cacharros, programas o aplicaciones que nos comunican con las máquinas sin tener que escribir programas (código).

Las UIs juegan un rol crucial en lo que se denomina: experiencia de usuario (notado UX, User eXperience). La definición de este concepto no es única y en [1] hay disponible una lista. No obstante, en términos generales, se puede afirmar que esta UX es algo subjetivo y concreto de un individuo que no concierne únicamente a la interfaz de usuario y también incluye otros elementos como las expectativas e imagen corporativa entre muchos otros.

Aunque hay bibliografía e incluso hay un estándar al respecto: ISO 9241 [5], esta área en el campo de IoT todavía está en una fase inicial que debe madurar. Hay que proceder con cautela ya que hay más dimensiones (dispositivos especializados, sistemas hardware y software heterogéneos, tipos de usuarios, etc.) que en una aplicación web clásica o web-app para smartphone. Si no se tienen en cuenta estas dimensiones como un todo, es posible llegar a hacer interfaces de usuario correctas de manera independiente (y es posible que ni eso) pero generar una experiencia de usuario incorrecta desde un punto de vista global [3]. Este hecho se denomina con el tecnicismo de *interusabilidad*. Valga como ejemplo de complejidad que hay que abordar, el hecho de que las acciones que se ejecuten en el mundo real no podrán deshacerse fácilmente (o quizá nunca) como sí ocurre por lo general con un sistema software.

2. Principios de diseño

El diseño de una interfaz de usuario es una tarea bastante compleja debido a la interdisciplinariedad que representa: psicología, artes, gestión de negocio y tecnología como mínimo. Aunque no es el objetivo de este curso, es positivo conocer qué principios suelen usarse como pilares para construir dichas interfaces y poder así evaluarlas de manera más razonada.

Dentro de los principios de diseño están [7]:

A. Principio del mínimo atosigamiento: con el fin de no desviar nuestro foco de concentración consciente.
B. Principio de formación de hábitos: repitiendo una tarea se convertirá más sencilla de hacer, por tanto, una interfaz debe permitir asimilar la repetición de manera fácil e intuitiva.

Estos principios quedan englobados dentro de una metodología (*framework*) popular para diseñar interfaces de usuario desarrollado conocido como *Honeycomb* (panal) [8]. En este entorno se presentan unas facetas que definen la experiencia de usuario: Útil, Usable, Deseable, Localizable, Accesible, Creíble, Valiosa.

Sin embargo, estos principios y facetas estarán restringidos en el contexto de IoT por los siguientes elementos [3]:

A. Contexto del mundo real: la situación concreta de aplicación del sistema impondrá una serie de restricciones inexistentes en el contexto homogéneo de una aplicación web. Por ejemplo, en el caso de aplicarlo al mundo de la conducción, es deseable no molestar/distraer lo mínimo posible al conductor (según el Principio del mínimo atosigamiento). Cuando haya que controlar un horno, habrá que priorizar elementos como evitar que haya un incendio. En el contexto familiar, tener varios perfiles de usuario y uno (o, peor, varios administradores del sistema) pueden llevar al agotamiento y frustración de los usuarios y de los administradores (como suele ocurrir en otros entornos), con el agravante de que en el IoT alguien se puede quedar sin acceso a la casa en mitad de la noche o sin poder prepararse la cena.

B. Diseño para sistemas de dispositivos y escalabilidad: los dispositivos tendrán que conectarse con otros dispositivos (de su mismo tipo o de otro), apareciendo el concepto de interusabilidad. Como suele decirse, una cadena se rompe por el eslabón más débil, de modo que debemos asegurarnos que la calidad de la interfaz es buena en todos sus niveles. Otro reto que hay que afrontar es la escalabilidad. Conforme aumente el número de dispositivos y usuarios, aumentará (en algunos casos exponencialmente) el tiempo de respuesta, monitorización y todo ello acabará proporcionando una experiencia desagradable para el usuario. El hecho de tener que interactuar con flotas de dispositivos suponen un reto cognitivo para el usuario con cierta similitud a programar un programa en paralelo en contraposición con hacerlo en secuencial.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

C. Diseño para sistemas conectados en red: en un entorno estándar, es normal asumir problemas menores como que haya desconexiones, retrasos en la carga de una web o en el envío de un correo-e. Sin embargo estos elementos en el contexto de IoT pueden no ser recibidos con tan buen agrado por parte de los usuarios, p. ej. si el usuario quiere encender una bombilla probablemente no acepte un retraso y lo mismo es aplicable a esperar tener encendida la calefacción y que el sistema no avise de que no ha podido conectarse con el termostato… otras situaciones más comprometidas son posibles cuando pulsemos varias veces sin respuesta un botón de abrir/cerrar la puerta de la casa sin saber en qué estado se ha quedado finalmente.

Para satisfacer dichas restricciones específicas, se propone el uso de un nuevo entorno de trabajo (*framework*). En [3] muestran un esquema que agrupa los distintos elementos en un sistema IoT y los ubica en un eje indicando su grado de visibilidad, de menor a mayor, tal y como ilustra la Figura 1.

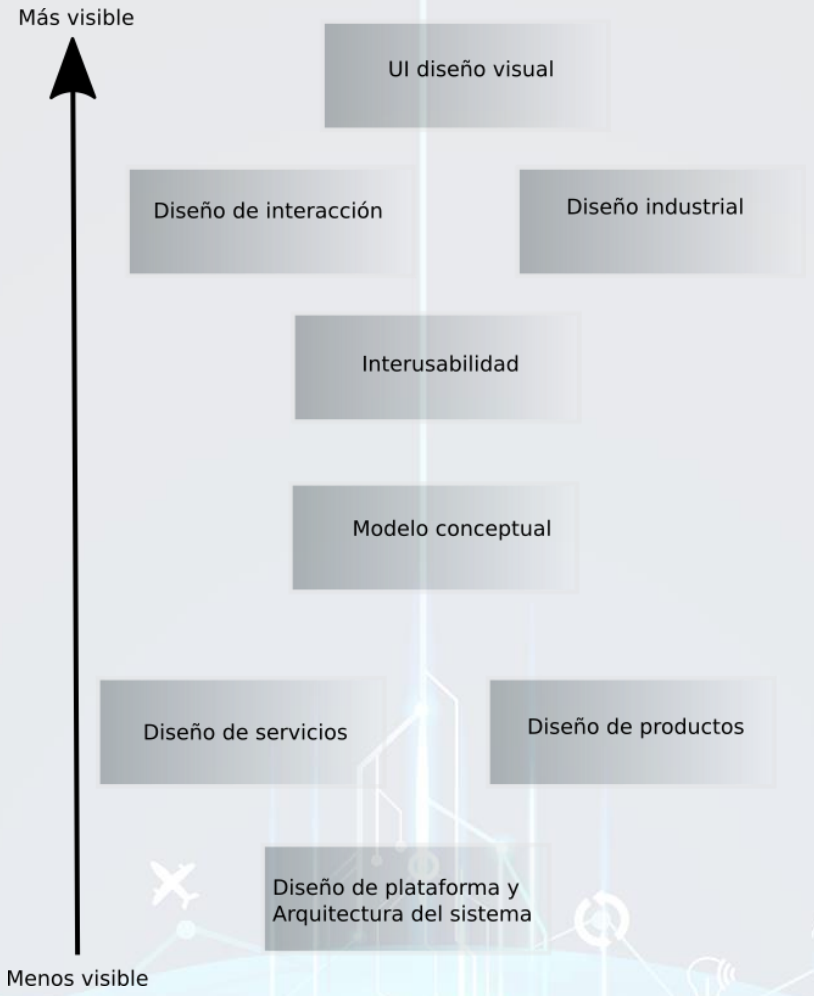


Figura 1. *Esquema que agrupa los distintos elementos en un sistema IoT según su grado de visibilidad*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

En esta figura aparece como pilar fundamental de la experiencia de usuario el concepto de modelo que debe responder a la pregunta: ¿Cómo piensan los usuarios en el sistema? A partir de esa respuesta se podrá desarrollar la *interusabilidad*, dotando de mayor o menor autonomía al sistema para, posteriormente hacer los diseños de interacción e industriales que finalmente nutrirán de sustento lógico a la capa de visualización e interacción con el usuario.

Esta última capa, en el caso de que sea una pantalla, debe especificar cómo se distribuye las imágenes y la información , la elección de controles (botones, desplegables, campos de texto, joysticks,...). El diseño de la interacción y la UI van estrechamente relacionados y deben retroalimentarse.

3. Tipos de Interfaces para sistemas IoT

Dentro de la gran variedad de posibles interfaces, vamos a presentar en esta sección una lista de posibles tipos, a modo de taxonomía, completando la lista propuesta en [4]. Este listado no es exhaustivo, pero permite obtener una panorámica general de sus posibilidades.

A. Interfaces por línea de órdenes

Son las interfaces de los primeros ordenadores personales donde había que introducir el código del programa. Posteriormente, con la aparición de los Sistemas Operativos (SOs), era posible interactuar con el computador introduciendo órdenes (*commands*) o llamando a programas.

Aunque estas interfaces aparentemente estén obsoletas, tienen una gran ventaja, la posibilidad de automatizar tareas. Es por ello que siguen siendo muy populares entre programadores. Otro de los elementos positivos es su poco coste computacional y baja latencia en la comunicación (solo hay que transmitir texto, no imágenes en HD como puede ocurrir con otras interfaces).

Como desventaja principal aparece la elevada curva de aprendizaje para hacer uso del sistema y, además, la necesidad de tener que usar un teclado para introducir los órdenes.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

```
alberto@fedora:~$ cd /
alberto@fedora:/$ ls -la
total 80
dr-xr-xr-x.  20 root    root  4096 nov 23 07:56 .
dr-xr-xr-x.  20 root    root  4096 nov 23 07:56 ..
lrwxrwxrwx.   1 root    root     7 ene 26  2021 bin -> usr/bin
dr-xr-xr-x.   7 root    root  4096 nov 21 19:41 boot
drwxr-xr-x.   3 alberto root  4096 nov 28 19:59 BORRAME
drwxr-xr-x.   4 alberto root  4096 sep 18 08:32 CUIDADO
drwxr-xr-x.  22 root    root  4300 dic  6 16:45 dev
drwxr-xr-x. 172 root    root 12288 dic  5 19:00 etc
drwxr-xr-x.   7 root    root  4096 nov  1 19:50 home
lrwxrwxrwx.   1 root    root     7 ene 26  2021 lib -> usr/lib
lrwxrwxrwx.   1 root    root     9 ene 26  2021 lib64 -> usr/lib64
drwx------.   2 root    root 16384 abr 23  2021 lost+found
drwxr-xr-x.   2 root    root  4096 ene 26  2021 media
drwxr-xr-x.   2 root    root  4096 ene 26  2021 mnt
drwxr-xr-x.   6 root    root  4096 oct  3 20:29 opt
dr-xr-xr-x. 420 root    root     0 dic  5 19:00 proc
dr-xr-x---.   7 root    root  4096 nov  9 01:40 root
drwxr-xr-x.  52 root    root  1420 dic  9 09:58 run
lrwxrwxrwx.   1 root    root     8 ene 26  2021 sbin -> usr/sbin
drwxr-xr-x.   2 root    root  4096 ene 26  2021 srv
dr-xr-xr-x.  13 root    root     0 dic  5 19:00 sys
drwxrwxrwt.  27 root    root   660 dic 10 01:15 tmp
drwxr-xr-x.  13 root    root  4096 jul  8 19:57 usr
drwxr-xr-x.  21 root    root  4096 jul 15 16:03 var
alberto@fedora:/$
```

Figura 2*: Imagen de pantalla que muestra órdenes para el computador introducidas manualmente*.

B. Interfaces de audio y voz

Estas interfaces han aumentado en popularidad recientemente debido a la gran mejora en el procesamiento automático del habla. Tienen la gran ventaja de permitir la comunicación sin usar las manos ni tener que estar mirando al dispositivo. Si el sistema es lo suficientemente avanzado, llegará a recoger matices intrínsecos del lenguaje oral como el tono de voz, especialmente útiles en situaciones de estrés y nervios. De cara a enviar información al operador o usuario, este aspecto será de gran utilidad para las alertas que requieran más atención.

Como desventaja principal aparece la “linealidad”, es decir, el sistema solo presenta una opción a la vez. Es lo que ocurre cuando tenemos que esperar a escuchar el mensaje completo para tomar una decisión cuando hablamos con un sistema de estas características.

Otro inconveniente es que el ruido de fondo o la forma particular de hablar del operador haga que el sistema no procese correctamente el mensaje y no tengamos alternativa para corregirlo.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 3. *Interfaz de audio que contiene como entradas/salidas un micrófono y un altavoz, permitiendo la comunicación con los dispositivos inteligentes a través de la voz. (Fuente: Pixabay).*

C. Controles físicos

Con este término hacemos referencia a una gran variedad de dispositivos físicos que nos permiten controlar valores. Por citar algunos elementos: volantes, ruedas, interruptores, botones, palancas, pedales, potenciómetros… todos ellos nos sirven para proporcionar entradas al sistema y responder a eventos.

Aunque pueda parecer que estamos en la era de las pantallas táctiles, la realidad es que, para ciertas operaciones, el control físico es más amigable y mejora la UX. Desde el punto de vista de la accesibilidad, este tipo de controladores marcan la diferencia entre poder ser usados o no por personas con visión reducida.

Otra ventaja añadida es el aspecto económico. El coste de reparación y reemplazo, sustituir una palanca, interruptor, pulsador es mucho más barato que reparar una pantalla (táctil o no) y el consumo energético de este tipo de dispositivos es cero, siendo este un aspecto fundamental en elementos empotrados.

Por último, la fiabilidad ante fallo software y la robustez ante las condiciones de operación también hacen de este tipo de controles la mejor opción ya que su rango de operación en situaciones ambientales adversas (calor, sol, lluvia, humedad) es mucho mayor que otros dispositivos de interfaces clásicos.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

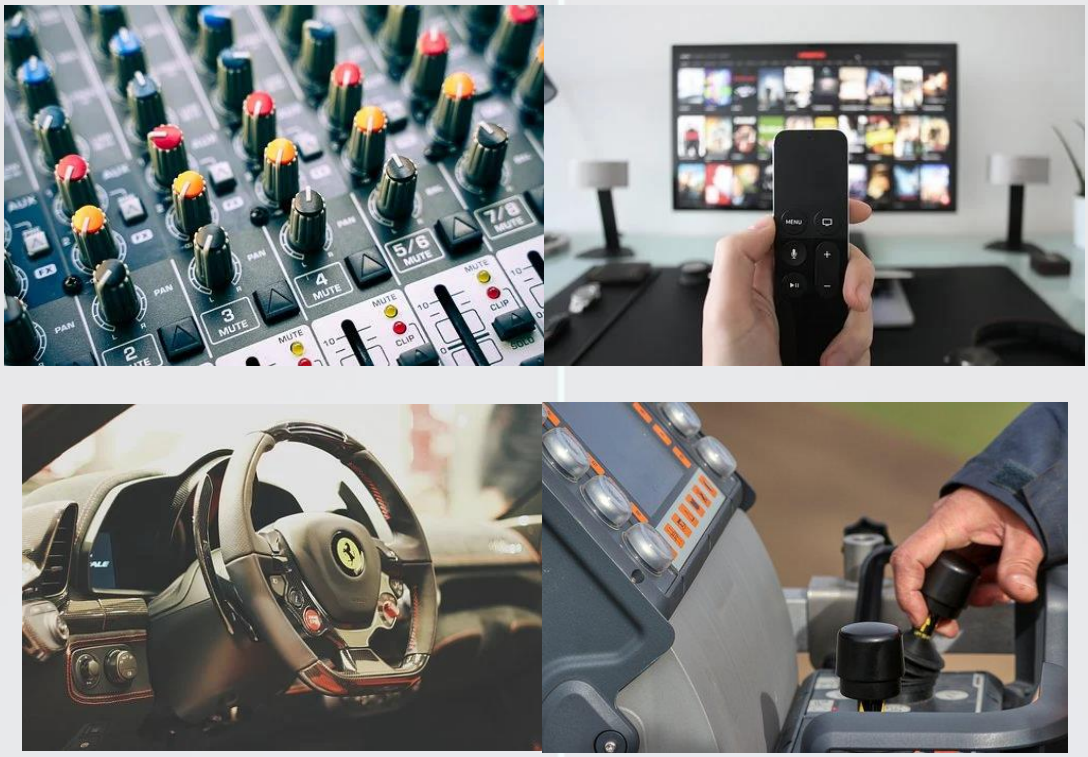


Figura 4: *Ejemplos de dispositivos físicos que nos permiten controlar valores (Fuente: Pixabay)*

D. Pantallas táctiles

Estas interfaces normalmente aúnan a varios de los tipos que hemos comentado anteriormente (y algunos de los que quedan por comentar).

Consisten en una pantalla que es capaz de detectar la posición donde es tocada, como tiene el 100% de los teléfonos inteligentes actuales.

Permiten una interfaz muy visual e incluso permiten disponer de un teclado en pantalla para escribir. No se alcanza la velocidad de un teclado normal pero sí que cumple lo necesario para introducir información. Además, realizar pulsaciones largas o cortas o utilizar varios dedos (o manos) si la pantalla permite *multitouch*.

Como desventajas, en el contexto de IoT, presentan: el coste de la pantalla, dificultad para visualizarse en entornos con mucha luz, poca sensibilidad en las interacciones, posibilidad de fallos software, exigir al dispositivo que procese la interfaz gráfica con el aumento en prestaciones necesario, el aumento de consumo y el coste del dispositivo.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

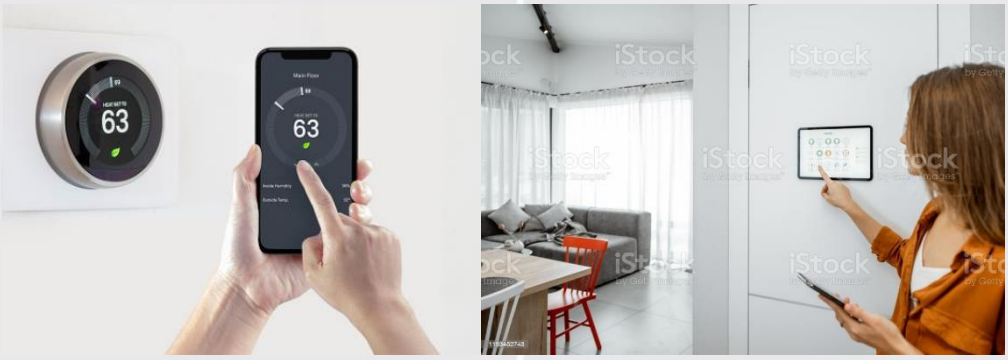


Figura 5: *Mujer controlando dispositivos inteligentes del hogar con una tableta digital. Fuente [5].*

E. Interfaces visuales, gráficas y aplicaciones Web

En general, todos los dispositivos suelen tener un LED o pequeña pantalla (*display*) para mostrar el estado (encendido/apagado) y alguna información adicional, p.ej. un LED para confirmar que recibe información de un mando por infrarrojos. Otro ejemplo de IoT puede ser los LEDs de colores que posee Google Nest para indicarnos que está escuchando.

Figura 6: *Pantalla de un termostato para un radiador controlable por aplicación móvil. Fuente: [6].*

Cuando el dispositivo posee una pantalla de mayor calidad permite visualizar ventanas, menús y gráficos mediante el uso de bibliotecas gráficas. En el entorno de dispositivos Linux como la Raspberry Pi nos encontramos las bibliotecas GTK+, Qt. Además, existe la posibilidad de integrar esas visualizaciones dentro de un navegador Web usando HTML y CSS [8]. Con la primera, utilizando el lenguaje de programación C, los desarrolladores crean ventanas, botones y demás elementos clásicos que estamos acostumbrados a ver en las aplicaciones de escritorio de un PC. Qt, nos permite incluir animaciones, aceleración de vídeo, además, pero, como contrapartida, hay que pagar por su

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

uso en dispositivos empotrados (como podría considerarse la Raspberry Pi). Respecto a las aplicaciones o webapps dentro de un smartphone hay dos alternativas: programadas en el sistema nativo (p.ej. aplicación Android o iOS) o ser una aplicación web incrustada. Normalmente la comunicación se hace a través de APIs (*Application Program Interface*) que comentaremos con más detalle a continuación.

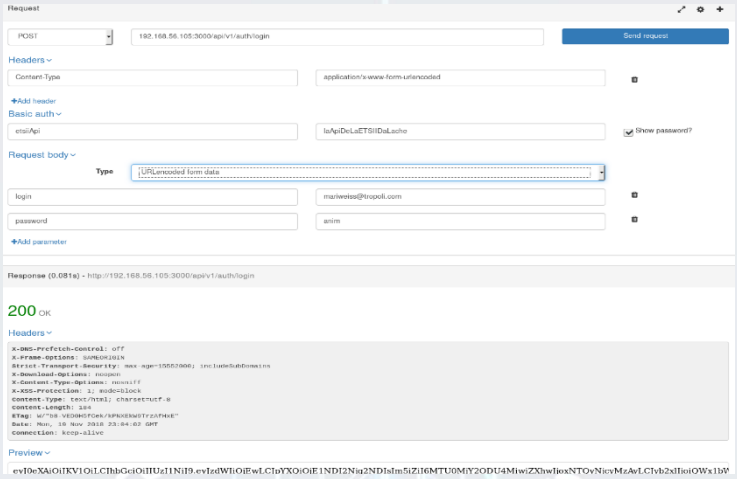
F. API REST

La interacción con algunos dispositivos se lleva a cabo mediante llamadas a funciones (pequeños programas) que usan RPC (*Remote Procedure Calls)*. Aunque con la aparición de la computación en la nube y otros paradigmas, se ha implantado como un estándar las API de tipo REST.

REST no es más que un acrónimo para *REpresentational State Transfer* acuñado por Roy Fielding en su tesis por la Univerisdad de California (Irvine) [11]. Este tipo de API implica que no hay un estado previo entre la comunicación del cliente y el servidor y que ésta debe contener toda la información necesaria para entender la petición. Esto las hace especialmente propicias para aplicaciones web en móviles y para IoT debido a la alta expansibilidad y escalabilidad [12].

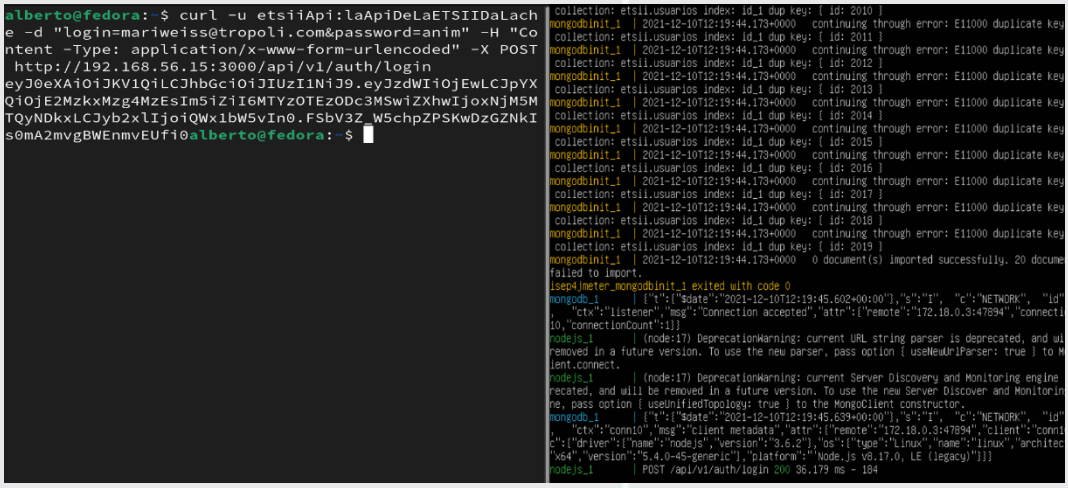
Cuando un cliente hace una petición al servidor, utiliza el protocolo HTTP (HyperText Transfer Protocol) y codifica toda la información dentro del cuerpo de la petición o, generalmente, dentro de un objeto en formato JSON o XML.

Las llamadas a la API se hacen a través de URIs (*Uniform Resource Identifiers*), a través de la línea de órdenes, utilizando plugins (Rested o Postman, p. ej.), o de manera transparente al usuario dentro del código de la interfaz.



(a)

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND



(b)

Figura 7: *Llamada de tipo POST a una API REST mediante:*
(a) el plugin RESTED de Firefox.
(b) usando la orden curl desde la línea de órdenes

Al usar este tipo de comunicación, se integra el servidor web (HTTP) dentro del dispositivo IoT y establecer comunicación directa con él, o bien, llevar la información a un sistema centralizado (en un servidor dedicado o en la nube) e interactuar de manera transparente al que hace la petición.

Una gran ventaja es que, al ser elementos estándar, no es necesario tener bibliotecas especializadas ni hechas a medida para establecer la comunicación. La desventaja es que se depende de la conexión a la red para establecer la comunicación y habrá que estar muy atentos a la configuración para evitar que la API sea explotada mediante alguna vulnerabilidad en el protocolo o en el procesamiento de los parámetros enviados.

G. Interfaces próximas: interfaces gestuales, hápticas y Realidad Aumentada

Además de las ya listadas, hay una serie de interfaces que han sido probadas en algunos dispositivos (especialmente en videojuegos) y que estarán presentes en el campo de IoT.

Las interfaces gestuales reconocen gestos y movimientos que utilizan como entrada para el sistema. Una de las interfaces más populares de este tipo es Kinect, integrada por Microsoft en su consola XBox. En el campo de IoT hay algunas pruebas al respecto hechas por Ninja Sphere y Onecue para controlar algunos elementos del hogar. Aunque en el campo de los videojuegos sea una interacción extendida, para IoT todavía tiene que mejorar la precisión para evitar falsos positivos (recordemos la dificultad o imposibilidad para deshacer cosas) y lidiar con el cansancio y fatiga muscular que podría llegar a ocasionar su uso continuado.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 8. Vídeo de Ninja Spere. *Accesible con el enlace de la referencia [13]*

Las interfaces hápticas se basan en el sentido del tacto, que puede detectar cambios en la rugosidad de la superficie, vibraciones o pulsaciones. De un modo muy sencillo, ya usamos la vibración del teléfono inteligente al pulsar las teclas como realimentación de que, efectivamente, ha realizado una lectura. Hay varias investigaciones al respecto y seguro que veremos su aplicación a IoT pronto.

Como híbrido entre gestuales y hápticas, aparecen las interfaces tangibles que utilizan un objeto con el que se interactúa mediante manipulación física. Principalmente se usan en museos y algunos instrumentos musicales experimentales.

Estas alternativas de interfaz se verán enormemente beneficiadas de la combinación con la Realidad Aumentada (AR). Gracias al enriquecimiento de la información visual sobre los elementos que estemos accionando o los gestos que hagamos tendremos una infinidad de posibilidades que repercutirán en el uso eficiente y condiciones óptimas de operación de los dispositivos IoT.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 9: *En un futuro cercano usaremos interfaces gestuales combinadas con elementos táctiles y realidad aumentada. Fuente: [14].*

4. Aspectos relacionados con seguridad: Autenticación y Autorización

A lo largo del texto hemos mencionado operador y usuario, algunas veces con un mismo significado. Sin embargo, esta ambigüedad de denominación puede resolverse fácilmente estableciendo roles que personalicen la interfaz según el tipo de usuario que la esté manipulando.

A. Roles

Los requisitos de interfaces en una solución IoT dependerán del tipo de servicio y dispositivos utilizados, aunque sí que se pueden identificar como estándar los siguientes perfiles de usuario [2]:

- **Administrador:** tendrá acceso al provisionado de dispositivos, configuración de usuarios y de la infraestructura y otros recursos.
- **Consumidor:** tendrá acceso a los dispositivos y servicios que tenga contratados y esté usando. Puede haber soluciones donde este rol se solape con el de operador.
- **Operador:** controlará el panel de monitorización y configuración, alertas y gestión de dispositivos.
- **Analista:** podrá usar los datos proporcionados por el panel de monitorización y telemetría para llevar a cabo tareas de análisis de datos.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

B. Autenticación y Autorización

Una vez definidos los tipos de roles más comunes es preciso clarificar dos conceptos clave para las interfaces: Autenticación y autorización.

La autenticación hace referencia al proceso de verificar la identidad de un usuario, de modo que garantiza que es quien dice ser. La autorización es verificar si el usuario, una vez autenticado, puede hacer ciertas acciones o mirar ciertos valores. Los roles nos permitirán definir qué debe hacer cada tipo de usuario, pero necesitaremos un sistema que implemente esas políticas de acceso y control.

Para autenticar, tenemos varias posibilidades:

1. Utilizando usuario y contraseña
2. Mediante token y/o llaves públicas privadas
3. Biometría

5. Consideraciones sobre privacidad

En el contexto del IoT los aspectos sobre privacidad y seguridad deben ser tratados desde el principio al fin del diseño e implantación, ya que habrá muchas etapas en las que ambos elementos puedan verse comprometidos. Una de estas etapas es el uso de las UI que, en ciertos contextos, puede ser el enlace más débil.

La huella que dejamos durante nuestra interacción con el dispositivo puede suponer un riesgo para nuestra privacidad. La forma única que tenemos de teclear, tocar y responder a las interfaces puede constituir un patrón inconsciente que, procesado con algoritmos de aprendizaje automático, pueda identificarnos unívocamente [15].

Aunque en ciertos contextos de seguridad esto pueda resultar tranquilizador, debemos ser conscientes de la pérdida de privacidad e intimidad que el hecho de interactuar con una interfaz supone.

Por otra parte, el proceso de autenticación a través de una UI es un potencial riesgo para la seguridad ya que alguien puede obtener nuestras credenciales fácilmente. Piense en cómo su banco le recomienda tapar su mano mientras teclea el PIN para acceder al cajero automático.

6. Conclusiones

En esta unidad se ha definido el concepto de interfaz de usuario, contextualizándolo dentro del concepto de experiencia de usuario. Tras indicar unas directrices y limitaciones en su diseño, se han analizado los distintos tipos de interfaces que nos podemos encontrar, estudiando sus ventajas y desventajas para el contexto de IoT. Finalmente se han descrito algunos conceptos importantes de cara a la seguridad en el uso de las interfaces.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

7. Referencias bibliográficas

[1] https://www.allaboutux.org/ux-definitions

[2] https://docs.microsoft.com/en-us/azure/architecture/guide/iot-edge-vision/user-interface

[3] User Experience Design for the Internet of Things. Claire Rowland. Oreilly. 2015

[4] "Commercial and Industrial Internet of Things Applications with the Raspberry Pi", Ioana Culic, Alexandru Radovici and Cristian Rusu. 2020. ISBN-13: 978-1484252956

[5] https://www.istockphoto.com/es/foto/mujer-controlando-dispositivos-inteligentes-con-una-tableta-digital-en-casa-gm1193452743-339469454?utm_source=pixabay&utm_medium=affiliate&utm_campaign=SRP_image_noresults&referrer_url=http%3A%2F%2Fpixabay.com%2Fimages%2Fsearch%2Fphysical%2520controls%2520iot%2F&utm_term=physical+controls+iot

[6] https://cdn.lampade.it/image/1600x/homematic-ip-termostato-per-radiatore-basic-3067005-33.jp

[7] ISO9241 https://www.iso.org/standard/63500.html

[8] Commercial and Industrial Internet of Things Applications with the Raspberry Pi: Prototyping IoT Solutions, By Ioana Culic, Alexandru Radovici, Cristian Rusu

[9] Humane Interface, The: New Directions for Designing Interactive Systems By Jef RaskinPUBLISHED BY:Addison-Wesley Professional PUBLICATION DATE:March 2000

[10] https://semanticstudios.com/user_experience_design/ **Peter Morville, 2004**

[11] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[12] https://www.redhat.com/en/topics/api/what-is-a-rest-api

[13] Vídeo de Ninja Spere. *https://www.youtube.com/watch?v=UzSYuQQoBeU*

[14] https://pixabay.com/es/images/search/augmented%20reality/

[15] User identification and authentication using multi-modal behavioral biometrics, Kyle O.Bailey, James S.Okolica, Gilbert L.Peterson Computers & Security Volume 43, Pages 77-89 June 2014.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Módulo 3. INTERFACES DE USUARIO Y PLATAFORMAS DE DESARROLLO

3.2 PLATAFORMAS DE DESARROLLO

Por Alberto Guillén Perales

Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada.

1. Introducción

El término plataforma o marco de desarrollo suele ser referido frecuentemente por su término en inglés *framework*: en el contexto de IoT, es un *middleware* entre una o más aplicaciones IoT que presenta una capa de red entre la que el resto de frameworks se comunican e interactúan [1].

De una manera informal, un *framework* es un conjunto de herramientas que permiten desarrollar aplicaciones de una manera estructurada sin tener que implementar de cero muchos aspectos. En el caso de IoT, es especialmente útil de cara a las comunicaciones e interusabilidad. Sin embargo, pueden suponer algunas limitaciones para personalizar ciertas cuestiones específicas, que, desde la perspectiva de IoT pasan por: baja latencia en comunicaciones, entornos síncronos y asíncronos, entornos con interferencias, bajo consumo, entre otros.

2. Elementos de un Framework

En esta sección describiremos de manera general qué elementos suelen ser los más comunes dentro de una plataforma de desarrollo. Hay que considerar que cada framework tendrá algunas abstracciones, otras no, y su propia nomenclatura.

A. Modelo de Datos y abstracción del sistema

Gracias a este elemento, podemos tener una forma homogénea de interactuar con los distintos dispositivos y sistemas. Por ejemplo, si pensamos en un sensor de temperatura, habrá que leer de su patilla correspondiente un valor y luego interpretarlo y codificarlo. Todas esas operaciones quedan abstraídas gracias al modelo de datos que representa ese dispositivo como:

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

```
{
   "sensor1" = "/sensortemp1
  {
        "tempActual"="25"
        "magnitud"="centígrados"
  }
}
```

Estas abstracciones posibilitan sustituir los elementos de bajo nivel de manera transparente a los elementos de capas superiores. Durante ese proceso de abstracción, hay ciertos elementos de información que quizá no sean contemplados, como en este caso concreto, el tiempo de refresco de la medición o los valores máximos y mínimos que el sensor puede medir.

Hay una serie de lenguajes que suelen ser usados para escribir estos modelos. Entre los más populares encontramos: JSON (Javascript Object Notation), XML (eXtensible Markup Language) y YAML (Yet Another Markup Language). Este último está adquiriendo mucha popularidad debido a la claridad de lectura por los humanos y minimización de errores de sintaxis. Prueba de ello es que está siendo adoptado por numerosas tecnologías (Ubuntu, Ansible, SnapperML, etc.).

B. Nodo IoT y Operaciones

Es frecuente encontrar la abstracción Nodo dentro del modelo de datos. Estos nodos encapsulan distintos sistemas que pueden estar compuestos por múltiples dispositivos.

La forma en la que estos nodos interactúan entre sí es reflejada mediante la abstracción de operación. Estas operaciones entre nodos engloban el cómo se descubren entre ellos, intercambio de mensajes, registro de eventos, etc.

Estas operaciones también pueden incluir elementos de conectividad más específicos, así como cuestiones relacionadas con la seguridad. Respecto a esta última, es normal que los framework proporcionen roles, credenciales y políticas de acceso para los nodos abstractos (e incluso físicos). También suelen proveer de mecanismos para implementar la seguridad en las comunicaciones con los nodos y entre éstos.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

C. Elementos de gestión y seguridad

El framework puede llegar a dar soporte para la gestión de dispositivos, hasta el punto de poder aplicar actualizaciones en el software de los dispositivos. No obstante, los fabricantes de dispositivos suelen dificultar esta tarea con el fin de que el software y el control del dispositivo sea lo más privativo posible.

Dentro de la gestión también podemos incluir aspectos de seguridad como la gestión del almacenamiento que puede contener datos sensibles.

También es probable encontrar mecanismos que permitan validar el estado del dispositivo y confirmar que el estado lógico (a nivel de software) se corresponde con el estado físico (lo que está haciendo/midiendo el hardware).

3. Arquitectura de un framework

Una vez listados los elementos básicos, vamos a analizar cómo se relacionan y organizan. Para ello definiremos distintas capas: Objeto Dato, Interacción entre nodos y comunicaciones y Aplicaciones y análisis [1]. Estas tres capas se sustentan sobre un sustrato que podríamos denominar plataforma que incluye distintos mecanismos de comunicación (p.ej. gestionar los dispositivos WiFi y los canales seguros con TLS) procurando el mínimo consumo. La plataforma también incluye los drivers para interactuar con el dispositivo, así como el entorno de ejecución necesario.

A. Capa de Objeto Dato

En esta capa se incluyen los nodos, como abstracciones de los dispositivos físicos. Por ejemplo, podemos tener un nodo de temperatura en una habitación que obtiene su valor agrupando distintas medidas de dispositivos físicos distribuidos por ésta. Debido a que cada dispositivo físico tiene su identificador (DeviceID) creado por el fabricante, es lógico definir un identificador único universal que permita hacer referencia a los nodos de manera homogénea y sin ambigüedades. Para ello se usa la nomenclatura establecida por el UUID (Universally Unique IDentifier) que consiste en un número de 16 bytes (es decir, 128 bits[1]) expresado en dígitos hexadecimales para facilitar la lectura por humanos.

Esta capa engloba algunos elementos de seguridad como listas de control de acceso (ACLs) y credenciales de autenticación protegiendo la información que pueden proporcionar los nodos a las aplicaciones que se comuniquen con esta capa.

[1] bit, la unidad mínima de información, codificado en 0 y 1 lógicos Por tanto un UUID son 128 0s o 1s seguidos

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

B. Capa de Interacción entre nodos

Estas comunicaciones e interacciones entre nodos se definen de manera que puedan ser interpretadas por operadores humanos, facilitando su monitorización y depuración. Un framework suele dar soporte a distintos tipos de intercambio de mensajes utilizando distintos protocolos. Se engloban, por tanto, los mecanismos de seguridad que deben aplicarse para evitar accesos no deseados o la manipulación del contenido de los mensajes entre los nodos.

C. Capa de abstracción de la plataforma

Esta capa abstrae el Hardware subyacente, por eso también se denomina HAL (Hardware Abstraction Layer).

Los elementos que suelen ser abstraídos se agrupan en: seguridad, interconexión y sensorial.

La relación de las tres capas se ilustra en la Figura 1.

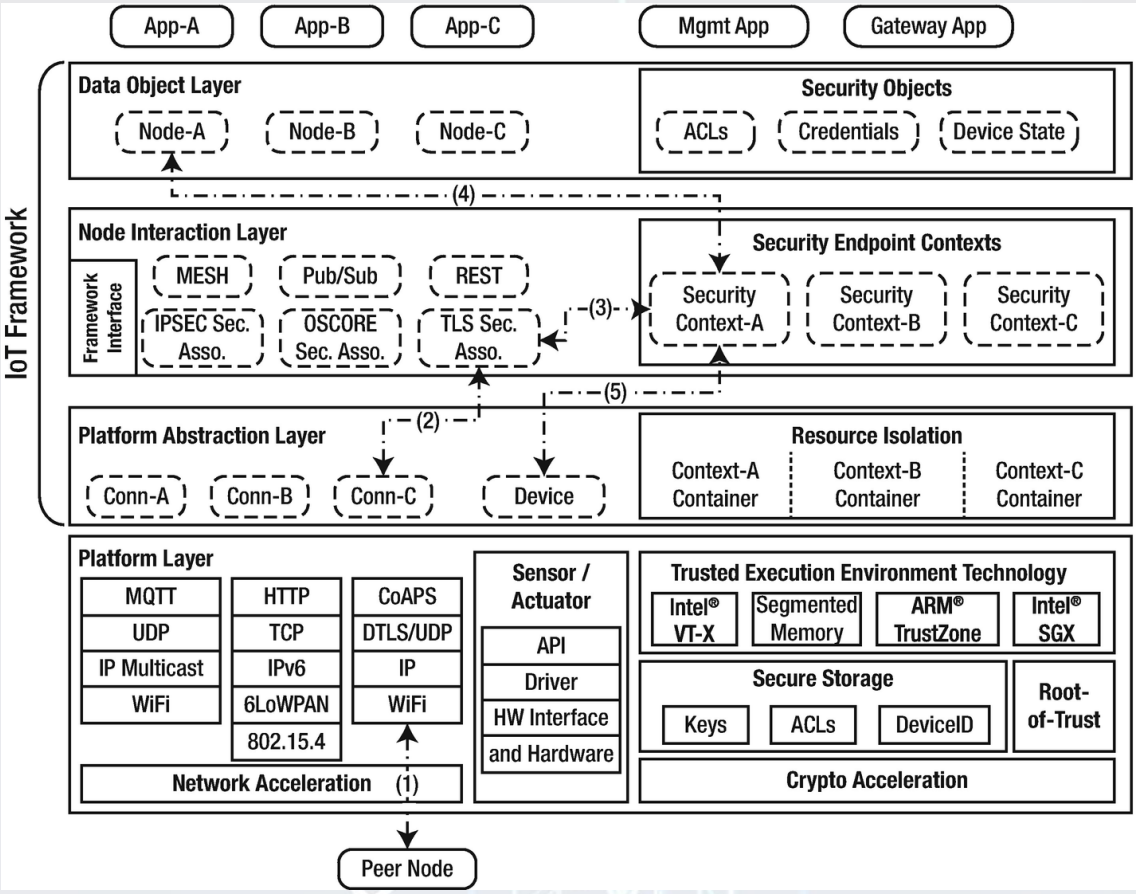


Figura 1: *Arquitectura en capas de un framework para IoT. Cada uno de los elementos de las capas puede mapearse con las abstracciones correspondientes en cada capa. En la figura se representa un camino de interacción en el que hay un nodo que establece una conexión (Peer node) con otro nodo (Nodo A). Para ello, usa el hardware de comunicación de red que tiene su representación abstracta dentro del framework. Tras descifrar el mensaje y comprobar los permisos de acceso, se accede a la abstracción Nodo A que finalmente consulta con el dispositivo. Fuente: [2]*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Cuando pensamos en las capas de abstracción, es posible establecer otra estructura, como la que nos presenta [3] que muestra menos solapamiento entre capas y una diferenciación como:

1. Dispositivo IoT: Desde sensores sencillos hasta dispositivos autónomos más complejos.
2. *Gateway* IoT (portal o entrada): centraliza varios sensores y dispositivos para poder establecer comunicación con ellos.
3. Conectividad: engloba las distintas tecnologías de comunicación usadas en las capas posteriores.
4. Gestión de conexiones: Establece las conexiones a nivel lógico incluyendo las abstracciones de identificación de dispositivos y configuraciones.
5. Habilitación inteligente (análisis de datos): capa en la que se aplican técnicas de IA (ML y DL) para extraer información y conocimiento de los datos generados en las capas inferiores.
6. Habilitación de aplicación (Plataforma IoT): engloba la totalidad de la plataforma incluyendo API, visualización, gestión de dispositivos y configuraciones.
7. Interfaz de usuario: cómo los operadores y los usuarios interactúan con la plataforma
8. Usuario: Define quiénes son los clientes finales y sus distintos tipos.
9. Solución / Proveedor de servicio: capa que define la relación entre el cliente y la empresa que proporciona el servicio.
10. Solución a nivel industrial: capa para cubrir aspectos de B2B que no son tan necesarios en los clientes finales, pero sí en relaciones empresariales (regulaciones específicas, estándares de operación, etc).

Aunque cada capa está más desarrollada en la fuente original, podemos ilustrar con un ejemplo real cada una:

Caso de estudio: Un ayuntamiento quiere disminuir la factura de la luz utilizando un sistema inteligente para el alumbrado público. Dado que es una institución pública, podríamos considerar que pertenece a la capa 9-10 así que debe plantearse la elección/diseño y desarrollo de los 8 restantes: desde Usuario a Dispositivo IoT. Para la capa 7, tendrá que ver los requisitos de los operarios del ayuntamiento. Para la 6, ver dónde se implantará la solución (servidor propio o virtualizado, p.ej.). En las capas 4 y 5, ver dónde se almacenará la información, quién tendrá privilegios para acceder a ella, cómo se va a procesar, entre otros. Los dispositivos integrados en las farolas, cómo se comunican y cómo la plataforma accede a esos datos corresponderían a las capas 1,2 y 3 respectivamente.

4. Estándares de Frameworks

En esta sección vamos a comparar distintos frameworks analizando los elementos comunes y las diferencias. Se hará una diferenciación si éstos están destinados al consumidor particular o al mundo industrial, coincidiendo con las capas definidas en la arquitectura de framework anterior.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

a. A Nivel de Consumidor

A. OCF (Open Connectivity Foundation) https://openconnectivity.org/

Esta fundación surge ante la disparidad de protocolos de comunicación y tecnologías que dificultan y condicionan la adquisición de dispositivos por parte de los usuarios [4]. Partiendo de esta premisa, el objetivo que establecen es desarrollar y evolucionar especificaciones y establecer un programa de certificaciones para tener comunicaciones seguras dispositivo a dispositivo, dispositivo a nube y nube a nube.

El framework que proponen define tres capas, tal y como indica la Figura 2, aunque los detalles técnicos pueden encontrarse en [6]:

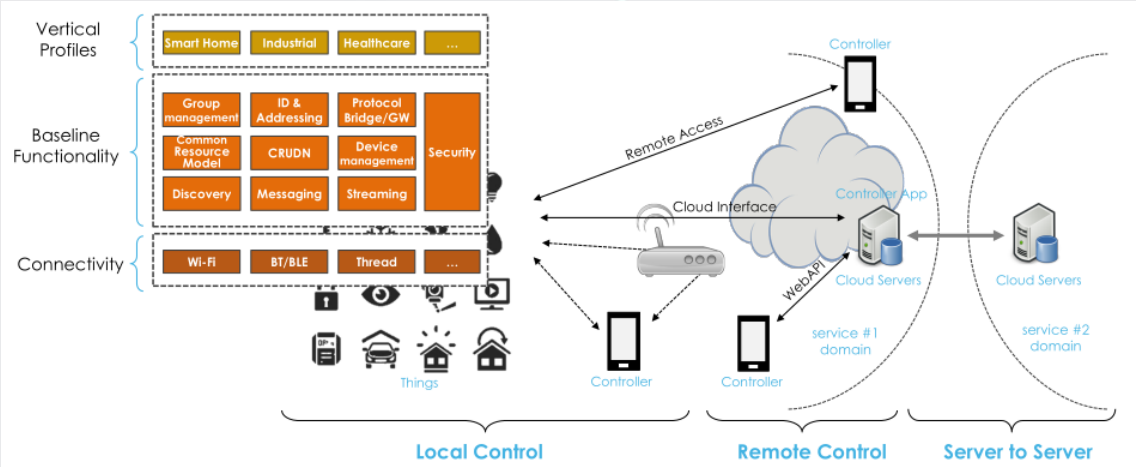


Figura 2: *Capas de abstracción del framework de OCF. Fuente: OCF [5]*

B. AllSeen Alliance/AllJoyn

Aunque el proyecto original surge de Qualcomm y es evolucionado por la Linux Foundation, en 2016 pasó a integrarse dentro de la OCF. La arquitectura de AllJoyn [7] tiene tres tipos de nodo: hoja, enrutador y puente. Los nodos hoja no se comunican con otros dispositivos directamente, sino que lo hacen a través de los nodos enrutadores. Estos enrutadores pueden comunicarse mediante los nodos puente.

C. Universal Plug and Play

Este estándar y framework es bastante más antiguo que los anteriores (primera especificación publicada en 2008 [8]) y tenía como objetivo no necesitar ninguna configuración de conexión por parte del operador. Sin embargo, el permitir a aplicaciones y dispositivos establecer conexiones y abrir puertos de manera automática conlleva riesgos de seguridad que hacen poco recomendable su uso actualmente [9].

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

D. Lightweight Machine 2 Machine (LWM2M)

Este proyecto [10] fue concebido dentro de la *Open Mobile Alliance* (OMA) para tratar con la gestión de dispositivos IoT. Tras la fusión de OMA con la alianza IPSO (*Internet Protocol for Smart Objects)* se ha continuado desarrollando este framework para que tome un enfoque de propósito general.

La arquitectura de este framework puede verse de manera general en la Figura 3.

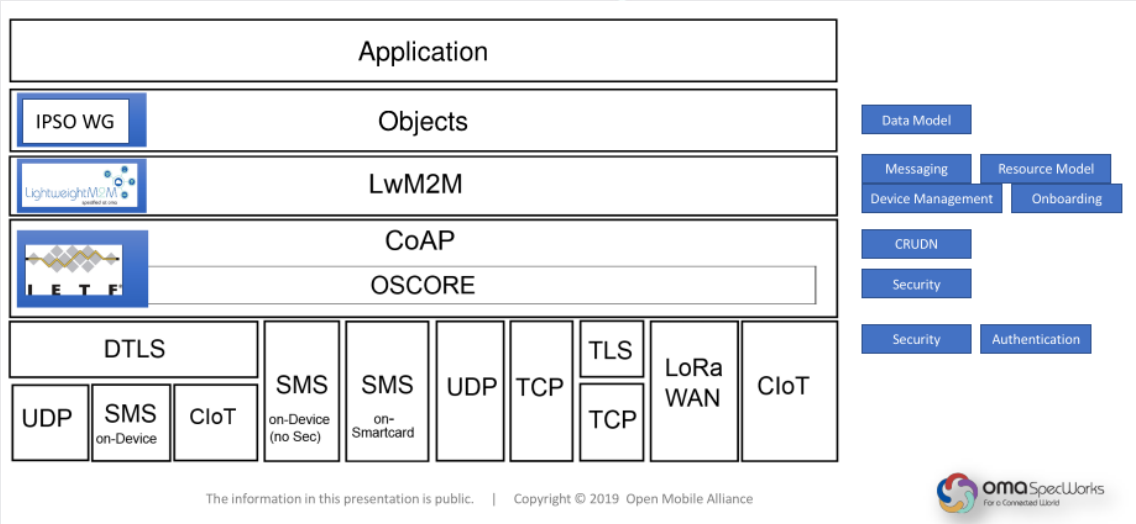


Figura 3: *Pila del protocolo LwM2M. Fuente: [15]*

E. One Machine to Machine (OneM2M)

OneM2M está compuesto por un conglomerado de organizaciones de estandarización internacionales. Tras cuatro años de trabajo, en 2016, publicaron lo que puede considerarse un framework IoT completo [11].

Al igual que la descripción general de la sección anterior, se definen tres capas para los distintos elementos del framework:

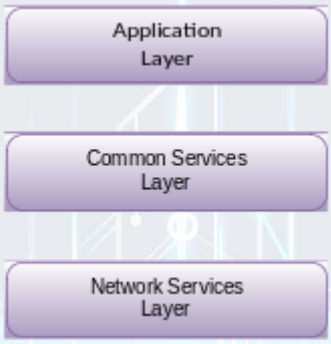


Figura 4: *Capas del framework OneM2M. Fuente: [16]*

UNIVERSIDAD DE GRANADA
abiertaugr
la formación abierta de la UGR
CC BY NC ND

b. A Nivel Industrial

Los frameworks listados a continuación fueron diseñados para satisfacer las necesidades de la industria, lo que se denomina como Industrial IoT (IIoT). Concretamente, a diferencia de un hogar, hay muchos más sistemas de control semiautónomos y su complejidad es bastante mayor.

A. Industrial Internet of Things Consortium (IIC) and OpenFog Consortium

El IIC se fundó en 2016 por grandes empresas que apuntan a las necesidades del sector energético, sanitario, transporte y de la industria de fabricación. Posteriormente, en 2018 se unió al OpenFog fundado en 2015 por grandes gigantes tecnológicos. En vez de proponer un framework para servir de referencia a implementaciones, el consorcio propone en primer lugar una arquitectura de referencia estableciendo como pilares de diseño: seguridad, escalabilidad, resiliencia, confianza, privacidad y autonomía [12], tal y como ilustra la Figura 5.

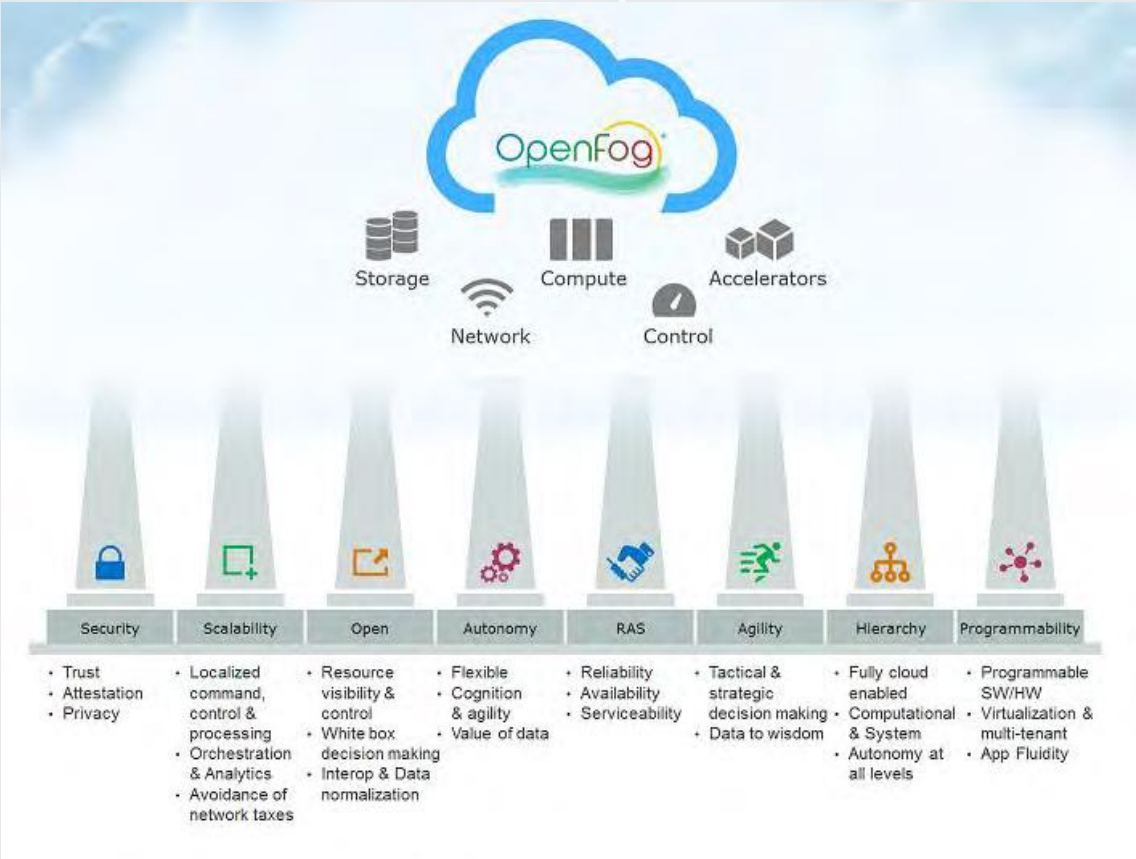


Figura 5: *Pilares de diseño de la arquitectura OpenFog. Fuente: [17]*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

B. Open Platform Communications-Unified Architecture (OPC-UA)

Tiene como punto de partida la extensión de la tecnología OLE (Object Linking and Embedding) para automatizar tareas. OPC se creó en 1996 con la idea de aplicar esta tecnología al mundo de la empresa. Los servidores de OPC se conectan con los sensores, actuadores y PLC (Controladores Lógicos Programables) asegurando la interoperabilidad entre éstos. La arquitectura unificada se basa en la clásica OPC pero añadiendo más funcionalidades como: mejora en las comunicaciones y sistema de descubrimiento de dispositivos, uso de más sistemas operativos, inclusión de más hardware (PCs, servidores cloud, microcontroladores), entre otros [13].

C. Data Distribution Service (DDS)

Este framework fue diseñado para los procesos de control industrial y el grupo que lo respalda es el OMG (*Object Management Group*). La primera versión de DDS se presentó en 2004 pero en 2015 sufrió una actualización [14]. Define cuatro entidades entorno al dato: dominios, tópicos, escritores y lectores y define una serie de políticas para determinar la calidad del servicio (QoS).

5. Entornos de desarrollo para IoT

Existen una variedad de entornos que, guiados por las recomendaciones de los estándares comentados en las secciones previas, permiten llevar a cabo proyectos sencillos como los que se propondrán en los siguientes módulos hasta despliegues en el campo industrial.

A continuación, comentaremos algunos de los más populares.

A. KAA IoT

En https://www.kaaiot.com/ tenemos información detallada sobre el producto y la posibilidad de analizar casos de estudio. Con este framework podemos integrar sensores, portales de comunicación (gateways), PLC industriales, vehículos, vestibles y smartphones.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

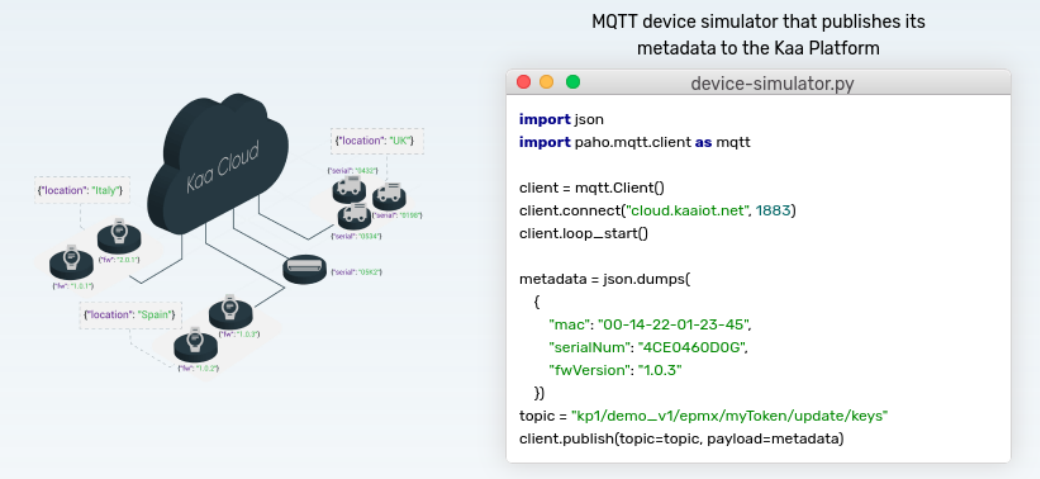


Figura 6: *Esquema de la plataforma KAA. Fuente: [18]*

Los dispositivos se especifican como objetos en notación JSON y se comunican con la nube mediante el servicio MQTT.

Para el procesado de los datos contamos con integración de bases de datos como Cassandra, MongoDB, InfluxDB y otras.

Además, tenemos la posibilidad de comunicarnos utilizando distintas integraciones entre sistemas como APIs Rest, websockets y usar herramientas como SAP y Salesforce.

B. MACCHINA.io (https://macchina.io/)

Este framework se subdivide en dos productos: Edge y Remote. El primero se centra en aplicaciones edge basadas en Linux utilizando C++ y Javascript. Con el segundo, podemos tener gestión de la infraestructura mediante paneles web y aplicaciones móviles. Dentro de las funcionalidades que nos brinda este framework, destaca ser una arquitectura débilmente acoplada proporcionando mayor flexibilidad.

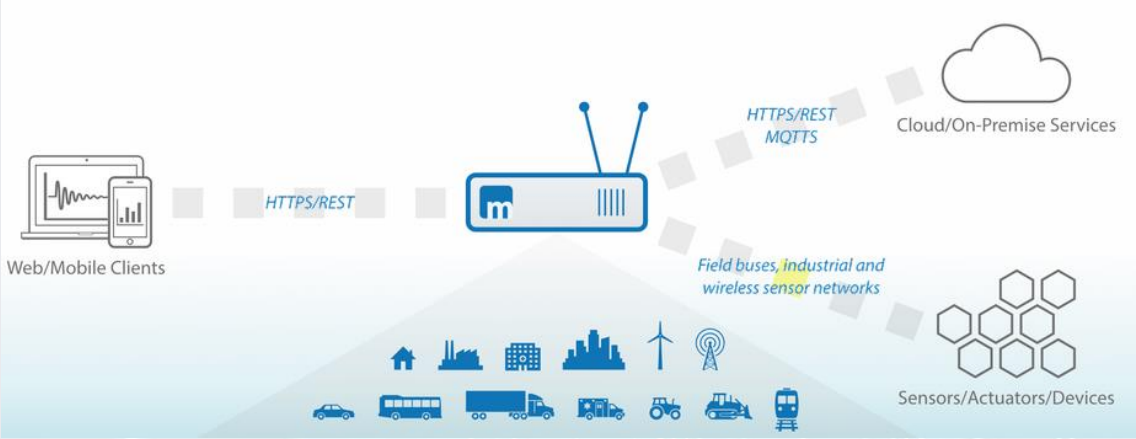


Figura 7: *Diagrama de la arquitectura de Macchina. Fuente: [19]*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

C. ZETTA (https://www.zettajs.org/)

Este framework está programado en Node.js, uno de los lenguajes interpretados con creciente popularidad. Utiliza APIs Rest, websockets y programación reactiva que permite trabajar con flujos de datos (data streams) de una manera más adecuada que con un enfoque clásico.

D. GE PREDIX

Este producto de General Electrics es un framework para IoT Industrial. Posee distintos módulos para gestionar los elementos de la infraestructura (assets) y sus operaciones.

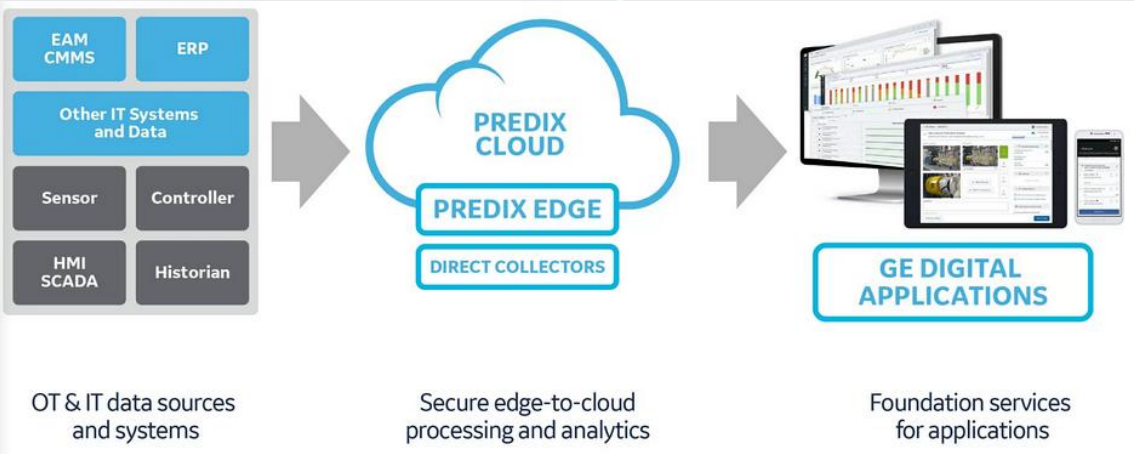


Figura 8: *Diagrama de la arquitectura de Predix. Fuente: [20]*

E. ThingSpeak (https://thingspeak.com/)

Es un framework centrado en la parte analítica (capa 4 de abstracción) en el que es posible llevar a cabo visualizaciones y predicciones utilizando MATLAB. También proporciona la recolección de datos con API Rest y MQTT y permite comunicar acciones a los dispositivos compatibles.

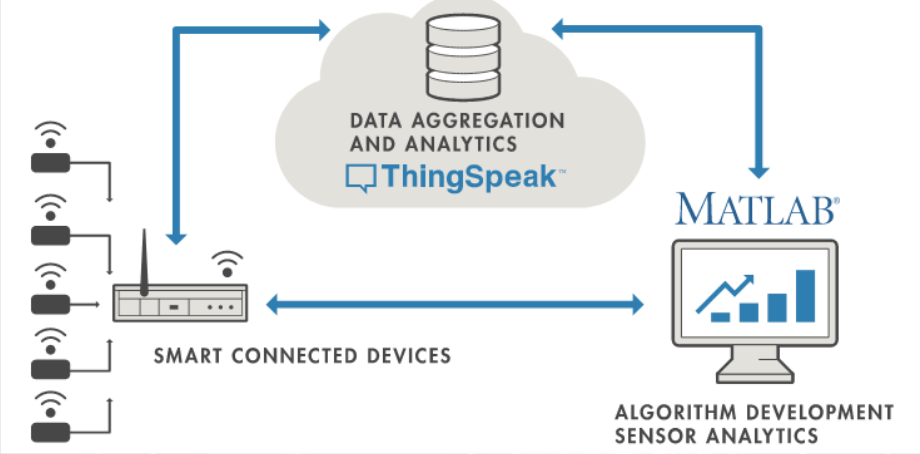


Figura 9: *Diagrama de la arquitectura de ThingSpeak. Fuente:[21]*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

F. Device Hive https://www.devicehive.com

Es una solución de código abierto (algo con bastante importancia desde el punto de vista de cumplir normativa de protección de datos). Utiliza tecnología de contenedores para tener un despliegue flexible en lo que a infraestructura de cómputo se refiere y proporciona escalabilidad. Respecto a las conexiones, al igual que otras soluciones permite usar API Rest, Websockets y MQTT. También se integra con soluciones de big data como ElasticSearch, Apache Spark y Cassandra.

G. Arduino https://docs.arduino.cc/cloud/iot-cloud

Aunque Arduino hace referencia a una marca con muchos productos, también está incluido dentro de la categoría de framework ya que posee un software que engloba la parte de gestión y programación de dispositivos. Se utiliza un lenguaje de programación visual muy indicado para aspectos didácticos y encaminado al nuevo paradigma de low-code. Además, ahora existe el producto de Arduino cloud que nos presenta un panel para monitorizar y controlar los dispositivos.

H. Node-RED

Es otra solución en la línea de low-code orientado a aplicaciones guiadas por eventos. Sin ser tan completa como la solución propuesta por Arduino, sigue siendo interesante desde un punto de vista didáctico y, en la línea de este curso, permite hacer pruebas sencillas sin mucha dificultad.

I. AWS IoT y otras soluciones comerciales

Hay otras soluciones comerciales como Salesforce IoT, Oracle IoT, RTI IoT, Mocana, Azure IoT pero todas se basan en ideas ya comentadas y desarrolladas.

La solución de Amazon puede ser bastante atractiva de cara al amplio ecosistema de servicios que integra y la gran cantidad de recursos a nivel de documentación y formación que proporcionan. Concretamente, en https://aws.amazon.com/es/iot/ aparecen listados algunos de éstos servicios específicos para IoT:

- FreeRTOS: sistema operativo para microcontroladores que integra con los demás servicios
- Greengrass: plataforma para combinar trabajo en nube y en el borde (edge)
- ExpressLink: capa de software para transformar cualquier dispositivo en uno IoT
- IoT Core: plataforma para conectar dispositivos sin desplegar infraestructura específica
- IoT Device Defender: auditoría y configuración de seguridad para dispositivos
- IoT Device Management: gestión de dispositivos
- FleetWise: recopilación y gestión de datos para flotas de vehículos
- IoT SiteWise: analítica de datos a nivel industrial
- IoT Events: detección y respuesta para eventos
- IoT Analytics : analítica de datos
- IoT TwinMaker : para crear gemelos digitales

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

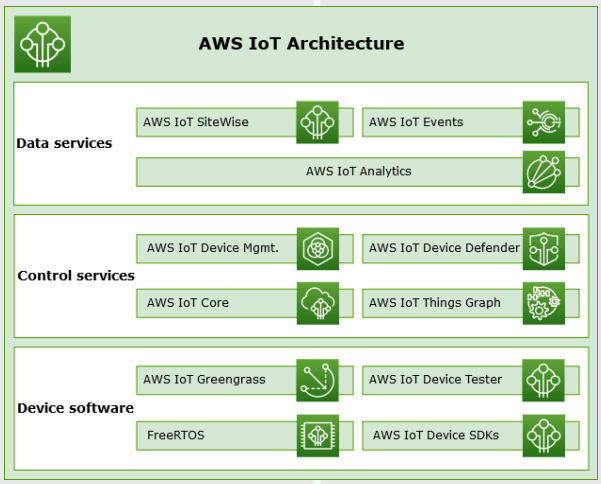


Figura 10: *Arquitectura del framework de AWS Fuente: [22]*

Dentro de las abstracciones que se definen, existe un duplicado sombra de un dispositivo (shadow device) que mantiene la interacción con el dispositivo aunque éste esté desconectado del sistema en ese momento. Gracias a esa caché, cuando el dispositivo vuelva a conectarse, se actualizará el resultado.

También es destacable el sistema de reglas de eventos y dispositivos que especifican, a muy alto nivel, cambios de estado y eventos que queramos hacer sobre los dispositivos.

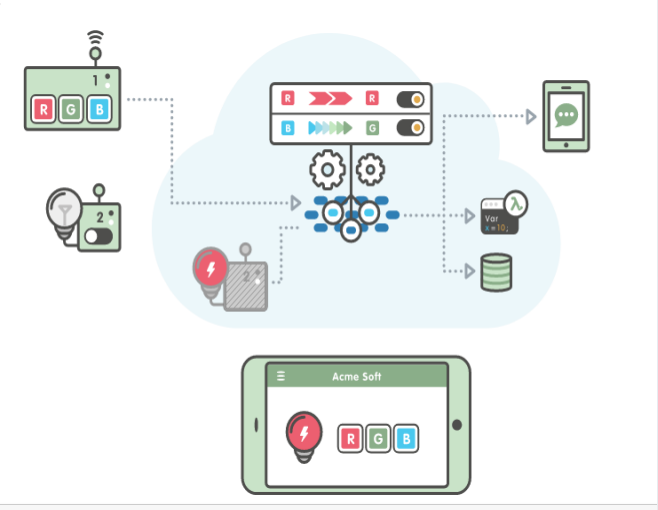


Figura 11: Dispositivo sombra que permite interactuar con las interfaces y los sistemas de reglas, aunque no esté conectado en un momento concreto el dispositivo (en este caso, una bombilla con el identificador 2). Fuente: AWS IoT Core Quick start guide.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

6. Conclusiones

Esta unidad ha definido el concepto de framework para IoT y estudiado qué elementos lo caracterizan. Tras analizar los estándares disponibles, se han listado algunas de las soluciones más populares para diseñar y desplegar nuestra infraestructura IoT. Entre las plataformas comentadas se han revisado soluciones comerciales y de código abierto que permitirán llevar a cabo cualquier proyecto imaginable.

7. Bibliografía

[1] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things, David Hanes, Gonzalo Salgueiro, Rob Barton , Cisco Press 2017
[2] Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment , Sunil Cheruvu, Anil Kumar, Ned Smith, David M. Wheeler, Apress, 2019
[3] https://www.iot.org.au/wp/wp-content/uploads/2016/12/IoT-Reference-Framework-Application-Guide-v1.0.pdf
[4] https://openconnectivity.org/foundation/
[5] https://openconnectivity.org/wp-content/uploads/2021/11/OCF_2.2.4_Specification_Overview.pdf
[6] https://openconnectivity.org/specs/OCF_Core_Specification_v2.2.4.pdf
[7] https://github.com/alljoyn/alljoyn.github.com/wiki
[8] https://www.iso.org/news/2008/12/Ref1185.html
[9] https://nordvpn.com/es/blog/what-is-upnp/
[10] https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/
[11] https://www.onem2m.org/technical/published-specifications/release-1
[12] https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf
[13] https://opcfoundation.org/about/opc-technologies/opc-ua/
[14] https://www.omg.org/spec/DDS/1.4
[15] https://www.openmobilealliance.org/release/LightweightM2M/Lightweight_Machine_to_Machine-v1_1-OMASpecworks.pdf
[16] https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=26462
[17] https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf
[18] https://www.kaaiot.com/
[19] https://macchina.io/
[20] https://www.ge.com/digital/iiot-platform
[21] https://thingspeak.com/
[22] https://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Módulo 3. INTERFACES DE USUARIO Y PLATAFORMAS DE DESARROLLO

3.3 HERRAMIENTAS DE DESARROLLO HARDWARE Y SOFTWARE

Por Beatriz Prieto Campos

Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada.

1. Introducción

El objetivo de esta unidad es describir y utilizar herramientas concretas que permitan ver de cerca algunos ejemplos prácticos que ayuden a comprender mejor cómo funciona IoT.

El hardware puede parecer algo muy lejano, que sólo entienden los ingenieros, pero gracias a Internet cada vez se está haciendo más popular, de forma que cualquiera puede desarrollar en casa pequeños o grandes proyectos de diversa naturaleza. Esta circunstancia ha hecho que surjan numerosas plataformas hardware y software al alcance de todos. La popularización de Internet ha sido decisiva en esta explosión porque ha permitido compartir miles de proyectos. Ejemplos de estas plataformas son ARDUINO, Raspberry Pi, Lilypad, Micro bit, SparkFun, Grove Kit, etc. [1]

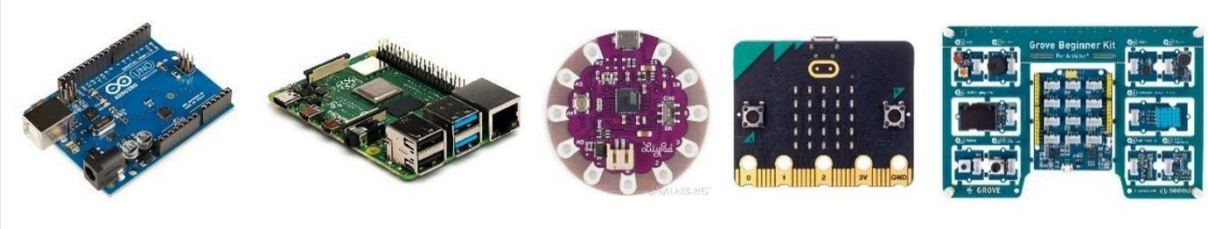


Figura 1. *Ejemplos de placas hardware abiertas.*

¿Te gustaría construir tu propio robot que circule de forma autónoma por tu casa? ¿O saber hacer una matriz de leds de colores que parpadeen como tú quieras? ¿O saber convertir tu cubo de basura en un cubo de basura inteligente? ¿O construir tu propia máquina de hacer cócteles? Estos son algunos de los proyectos hechos con Arduino más vistos en YouTube [2] y que cualquiera puede hacer en casa.

Una de las plataformas que vamos a poner como ejemplo es ARDUINO [3] por ser hardware libres y una de las más populares en todo el mundo.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

2. ¿Qué es ARDUINO?

Es una plataforma electrónica de hardware libre, es decir, accesible a todo el público, basada en una placa de circuito impreso con un microcontrolador y un entorno de desarrollo fácil de utilizar. Contiene los componentes básicos de un ordenador sencillo (memoria, procesador y entradas/salidas), a los que se pueden conectar otros elementos muy variados y en cuya memoria se almacena un programa que le dice a esos elementos lo que queremos que hagan. Esta plataforma se ha diseñado con el objetivo de que cualquiera pueda construir dispositivos digitales que pueden recoger información del entorno (temperatura, pulsación de un botón…) y controlar otros dispositivos (activar un motor, encender una luz…) a partir de esa información [3].

¿Por qué tiene tanto éxito ARDUINO?

El éxito de esta plataforma se debe a diferentes motivos, y los más destacados son:

- **Barato**: el precio tanto de las placas como de los diferentes componentes es muy asequible, lo que es fundamental para que pueda llegar a todo el mundo.
- **Multiplataforma**: se puede utilizar con multitud de sistemas operativos como Windows, Linux, Mac OS, etc.
- **Entorno de programación sencillo**: existen herramientas de programación tanto para niños o principiantes como para programadores avanzados.
- **Código abierto tanto el software como el hardware**: es una plataforma en la que tanto el código fuente como el hardware está al alcance de cualquiera de forma gratuita y con las licencias necesarias para que se pueda utilizar en diferentes contextos.
- **Uso de "Shields"**: Arduino permite utilizar placas de circuito impreso complementarias que proporcionan mayor conectividad, ampliando la funcionalidad de la placa principal. Por ejemplo, hay shields que proporcionan conectividad bluetooth, Wifi, visualizadores LCD, etc.

3. ¿Qué necesito?

Para llevar a cabo un proyecto, ya sea sencillo o complejo, necesitamos una placa ARDUINO, lo que queramos conectarle (sensores, actuadores, etc.), un cable USB y un ordenador con un entorno de programación que nos permita crear y transferir a la placa el programa que le va a decir al hardware lo que tiene que hacer (figura 2).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 2. *Lo que necesito para hacer un proyecto con ARDUINO.*

Además de los elementos mencionados, es muy común utilizar placas de trabajo o de pruebas (protoboard). En muchas ocasiones, hay que conectar diversos componentes y, por lo tanto, hay utilizar un número elevado de cables siendo complejo e incómodo conectarlos directamente a la placa. Las placas de trabajo (protoboard) facilitan este conexionado y permiten organizar mejor el diseño. Una placa de trabajo (figura 3) es sencillamente una tabla con orificios que se encuentran conectados eléctricamente entre sí, siguiendo líneas. En esos orificios se pueden pinchar cables y componentes electrónicos para hacer prototipos de circuitos electrónicos.

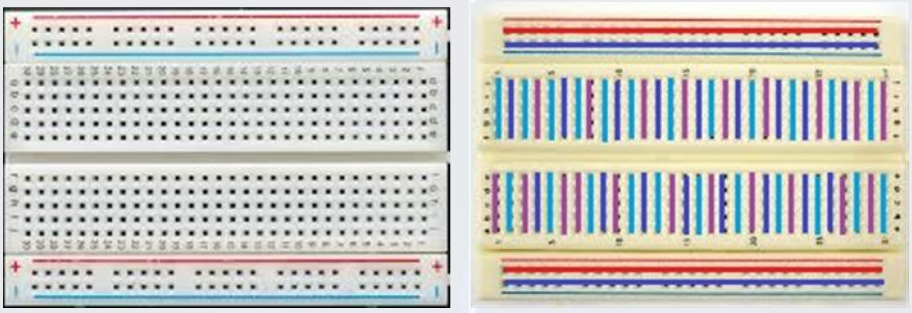


Figura 3. *Placa de trabajo y conexionado interno.*

En la figura 4 puede verse un ejemplo de uso de una placa de trabajo. En este caso, en lugar de conectar directamente los leds a la placa, se hace en la placa de trabajo con lo que facilita el conexionado y todo queda más ordenado.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

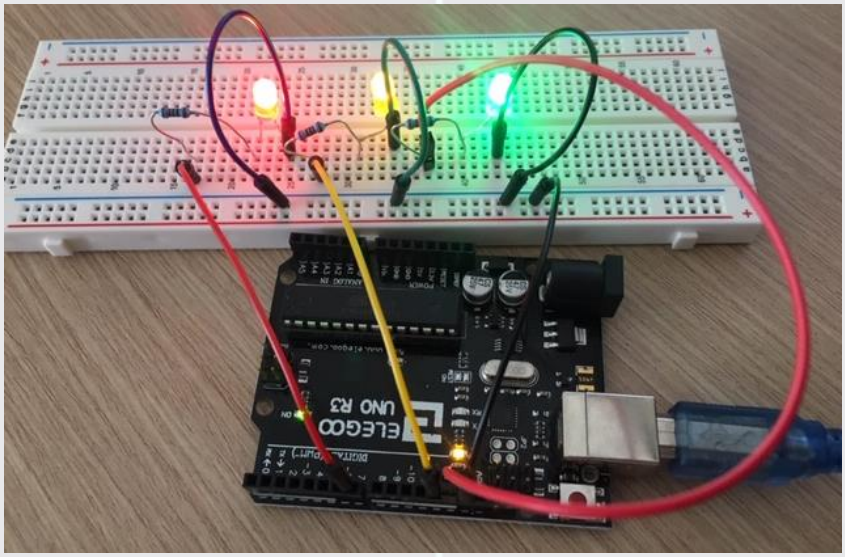


Figura 4. *Ejemplo de uso de una protoboard.*

Resumidamente, los pasos a seguir a la hora de llevar a cabo cualquier proyecto serían los siguientes:

1. Hacer un diseño en el que se indique cómo se deben conectar todos los elementos a la placa.
2. Escribir el programa con las acciones que queremos que haga nuestro hardware y comprobar que no tiene errores.
3. Conectar adecuadamente a la placa los elementos que vayamos a utilizar (sensores, actuadores…), siguiendo el esquema de diseño del punto 1.
4. Transferir el programa del ordenador a la memoria de la placa mediante un cable USB o por Bluetooth (si es posible).
5. Ejecutar el programa y ver los resultados.

4. ¿Qué cosas puedo hacer?

Básicamente, lo que hacemos en cualquier proyecto de ARDUINO es leer entradas (el estado de un sensor, una huella en un botón, un mensaje de Twitter, etc.) y según los valores de esas entradas llevar a cabo acciones (activar un motor, encender un LED, publicar algo online, etc.) (Figura 5). Por ejemplo, ver el valor de un sensor de humedad, y según sea ese valor, abrir una válvula por la que salga agua. Con esta idea podría hacer, por ejemplo, un proyecto que consista en obtener el grado de humedad de la tierra y según sea, regar o no una maceta.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

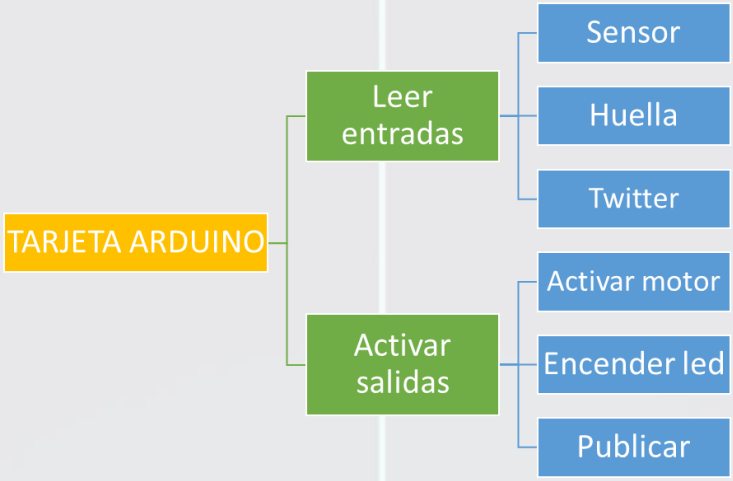


Figura 5. *Ejemplos de "cosas" que puedo hacer con ARDUINO.*

En la figura 6 se ven diferentes elementos que podemos utilizar en nuestros proyectos. Por ejemplo, sensores de corriente, de gases, de temperatura, presión y humedad, de lluvia, etc. y, según esos valores, usar una pequeña pantalla (display) para mostrar mensajes, activar motores para mover cosas, etc. Conectando estos dispositivos a nuestra placa se pueden desarrollar infinidad de proyectos.

Figura 6. *Ejemplos de sensores, actuadores y otros elementos.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

5. Ejemplos de placas

La placa es el elemento fundamental, ya que dispone de los elementos esenciales de un ordenador básico y gracias al programa que contiene, podemos interactuar con otros dispositivos y con el entorno. Existen muchos tipos de placas, según el proyecto que se quiera llevar a cabo. Por ejemplo, Arduino tiene [4]: placas de iniciación (UNO, LEONARDO, MICRO), KITS para educación, placas con funcionalidades avanzadas (NANO, WIFI-REV2, DUE, MEGA) o placas para IoT (NANO, MKR). El objetivo de IoT es conectar "cosas", objetos cotidianos a Internet o entre ellos. Las placas orientadas a IoT, lo que hacen es facilitar esa tarea. En general, este tipo de placas tiene las siguientes características:

- Son pequeñas, de forma que las podemos incorporar en objetos pequeños.
- Consumen poco. En la mayoría de los casos en IoT es necesario que las "cosas" estén conectadas prácticamente todo el tiempo, por eso deben consumir poco.
- Ofrecen diversas tecnologías de comunicación como Bluetooth, WIFI, GSM, etc. para facilitar la conexión de estas placas a Internet y/o entre ellas.

Para entender mejor cómo son y cómo funcionan estas placas vamos a ver dos ejemplos. Para el primer ejemplo utilizaremos la placa básica de Arduino: UNO R3 y para el otro ejemplo usaremos otra placa orientada a IoT, compatible con Arduino, pero de otro fabricante: NODE MCUV3.

Placa ARDUINO UNO R3 [5]

Es una placa orientada a principiantes, para empezar a desarrollar proyectos sencillos. Contiene los siguientes elementos:

- Un microcontrolador ATmega328P de 8 bits que tiene una memoria interna de 32 KB. En esa memoria es donde se almacenará el programa que escribamos.
- 14 entradas/salidas digitales y 6 entradas analógicas a las que conectamos los sensores y actuadores.
- La placa se puede alimentar con un cable USB o con una entrada para conectar un adaptador AC-DC o pilas.
- Un led integrado en la placa cuyo encendido/apagado se puede controlar con la entrada/salida digital 13.

En la figura 7 se puede ver la placa en la que se indican alguno de sus elementos.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

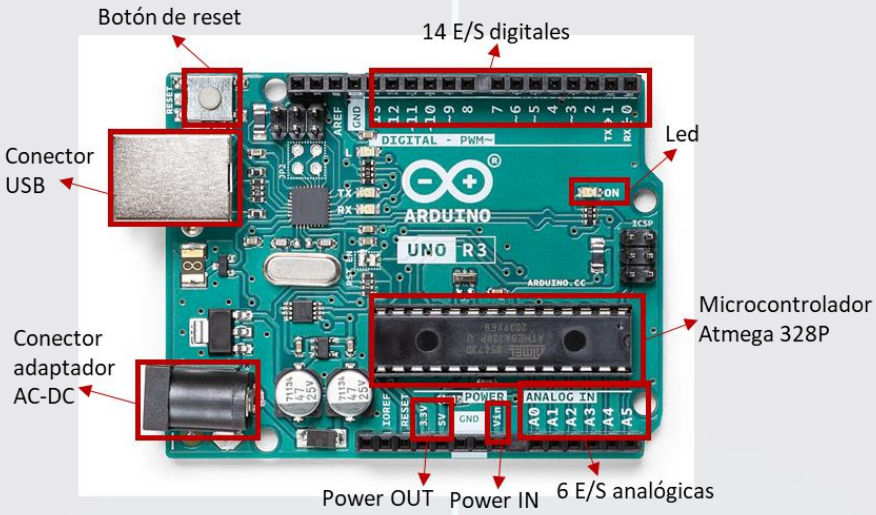


Figura 7. *Algunos elementos de la placa ARDUINO UNO R3.*

Placa NODEMCU V3 [6]

NodeMcu V3 (figura 8) es una placa de desarrollo, abierta, a nivel de hardware y software. Dispone de un chip con conexión WiFi y compatible con TCP/IP. Además, su tamaño es bastante reducido y su procesador de bajo consumo. Por todo ello, es muy adecuada para aplicaciones IoT. Gracias a su módulo WIFI se pueden enviar y recibir datos e incluso controlar las entradas y salidas de forma remota e inalámbrica.

Esta placa tiene un procesador de 32 bits (Tensilica L106) de bajo consumo, un módulo WIFI ESP8266 y una memoria RAM de 50 KB. Dispone de 1 entada analógica y 13 entradas/salidas digitales. Tiene un conector micro USB para conectarlo a un ordenador o para alimentar la placa.

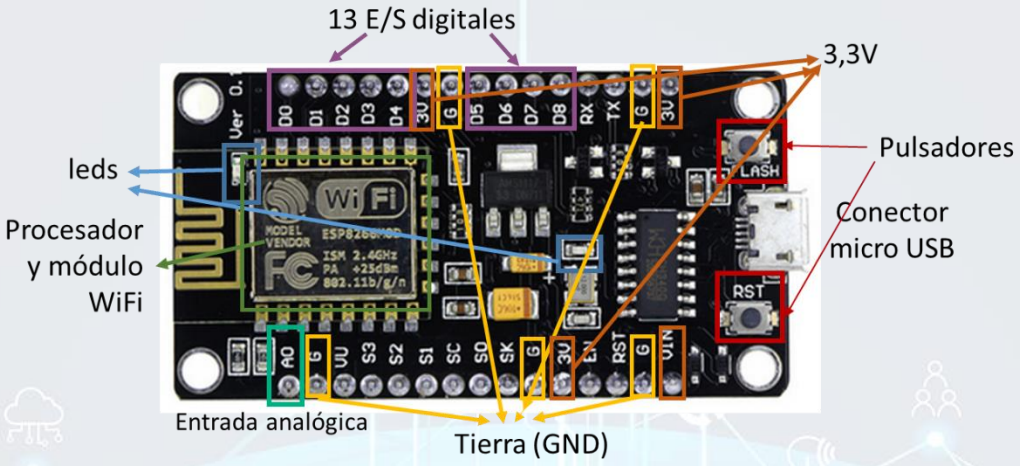


Figura 8. *Algunos elementos de la placa NODE MCU V3.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

6. Entornos de programación

Para poder llevar a cabo cualquier proyecto es necesario un programa que le indique al hardware lo que tiene que hacer. El hardware sin el software no tiene sentido y viceversa. Una de las ventajas de estas plataformas libres es que existen numerosas herramientas que nos facilitan la tarea de desarrollar estos programas, incluso sin saber nada de programación. Vamos a describir dos tipos de herramientas, una que nos va a permitir programar de forma visual o por bloques y que está orientada a niños y principiantes, y otra no visual, para programadores más avanzados.

Además de estas herramientas existen otras online que permiten simular proyectos. Si no se dispone del hardware físicamente (placa, sensores, actuadores, etc.) se puede simular su comportamiento. Algunas de ellas, permiten tanto la programación por bloques como directamente en un lenguaje de alto nivel. También veremos un ejemplo.

Entornos de programación no visuales

ARDUINO dispone de dos herramientas que permiten escribir y cargar programas en la placa para usuarios con conocimientos de programación. Una de ellas se utiliza online a través de un navegador (Chrome, Firefox, Safari o Edge), llamada *Arduino web editor* [7] y otra, que hay que descargarla e instalarla en un ordenador, *Arduino Software (IDE)* [8]. Esta segunda opción es compatible con la mayoría de los sistemas operativos (Windows, Linux, Mac OS).

Arduino IDE tiene un editor de texto para escribir los programas directamente en un lenguaje de alto nivel. Hay una barra de herramientas con menús para acceder a diferentes funciones, una zona de mensajes, y una consola de texto (figura 9).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

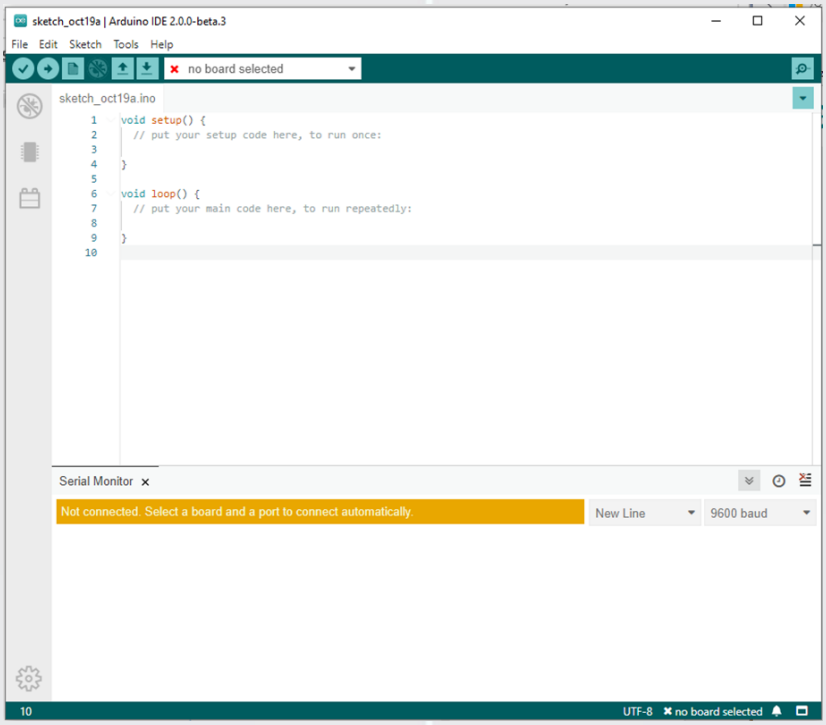


Figura 9. *Editor Arduino IDE.*

La versión online no requiere instalación y se recomienda que se utilice con el navegador Chrome. Para acceder a esta versión, sólo es necesario entrar en la web https://create.arduino.cc/editor, crear una cuenta e instalar un pequeño programa (*Arduino create agent* [9]) para que el editor web reconozca la placa que utilicemos, cargue el programa y acceda al monitor serie. A partir de ahí se accede al editor online, para escribir programas y cargarlos en la placa. En la Figura 10 se ve el aspecto del editor online.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

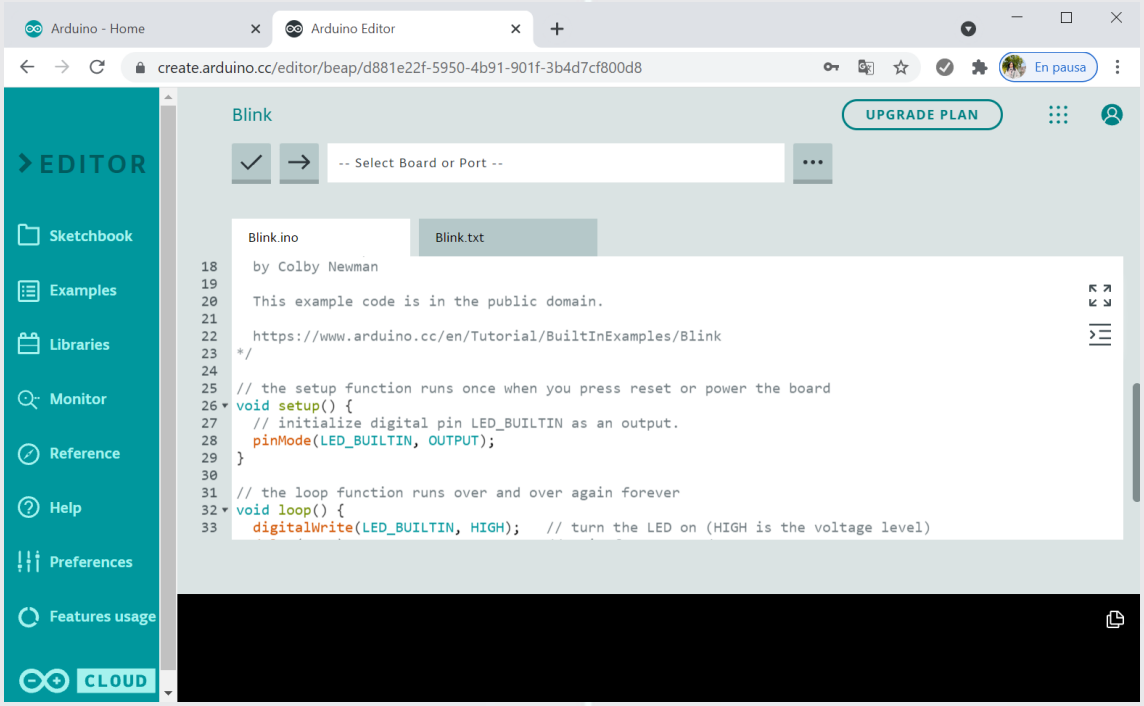


Figura 10. *Aspecto de Arduino web editor.*

Entornos de programación visual por bloques

La programación visual por bloques permite aprender de forma sencilla conceptos básicos de programación. Las herramientas de programación por bloques ayudan a encontrar una solución óptima a un problema concreto escribiendo una secuencia lógica y ordenada de instrucciones representadas mediante bloques gráficos. Existen multitud de herramientas de este tipo como Arduinoblocks, snap4arduino, visualino, Ardublock, BlocklyDuino, MBLOCK, MiniBloq, etc.

Un ejemplo de una herramienta que simula el comportamiento del hardware y que permite la programación por bloques es Tinkercard [10]. Tinkercad es una aplicación web gratuita y fácil de usar que permite simular diseños 3D, circuitos electrónicos y código. Para poder utilizarlo hay que crearse una cuenta. Nuestro objetivo es hacer una simulación de circuitos electrónicos por lo que, al entrar, tenemos que seleccionar “Circuitos” en el menú de la izquierda (figura 11).

En la parte central hay un botón para “Crear nuevo circuito” y debajo un listado de todos los circuitos que vayamos creando. Al situar el cursor sobre cada circuito creado aparece un botón de MODIFICAR para acceder a dicho circuito, y poder editarlo o simularlo.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 11. *Entorno Tinkercad.*

Vamos a ver un ejemplo sencillo con detalle para entender cómo funciona todo que consiste en encender y apagar un led (emisor de luz).

Cuando se lleva a cabo cualquier proyecto lo primero es tener claros los objetivos. En nuestro ejemplo, el objetivo es hacer parpadear un led. A continuación, hay que ver qué componentes necesitamos. En este ejemplo serán una placa, ARDUINO UNO R3, un led, una resistencia y un par de cables. Para encender el led es necesario suministrarle corriente, la corriente que sale de la placa es muy elevada y puede fundir el led. Para evitarlo, vamos a utilizar una resistencia. En una resistencia parte de la corriente que recibe se disipa en forma de calor de forma que la corriente que sale de la resistencia es inferior a la que entra. La corriente que le llegue al led, depende del valor de la resistencia, que se da en Ohmios. Una resistencia de 220 Ω es suficiente para que el led no se funda. El led tiene dos patas, una de ellas hay que conectarla a tierra y la otra pata a la resistencia, que a su vez va a la entrada/salida de la placa que le suministra la corriente. La entrada/salida que vamos a seleccionar es la número 12. Una vez elegidos los componentes hay que conectarlos según se indica en el diseño o esquema de la figura 12.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

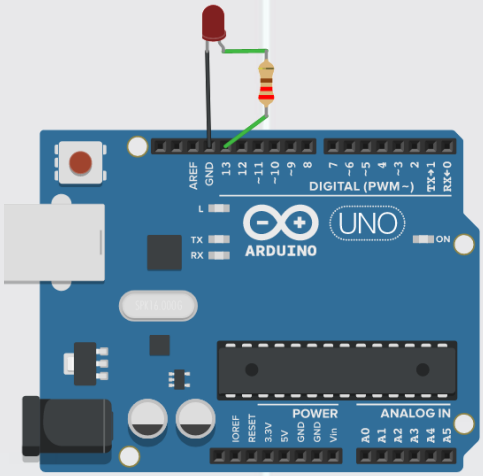


Figura 12. *Esquema de conexión del led.*

Para hacer el montaje en Tinkercad vamos al menú de la derecha, seleccionamos Componentes, Básico y en el buscador escribimos Arduino. Nos saldrá un listado de placas y, seleccionamos ARDUINO UNO R3 y la arrastramos a la parte central. Cada vez que seleccionamos un componente tenemos la opción de ponerle un nombre. De forma similar buscamos la resistencia y el led (Figura 13).

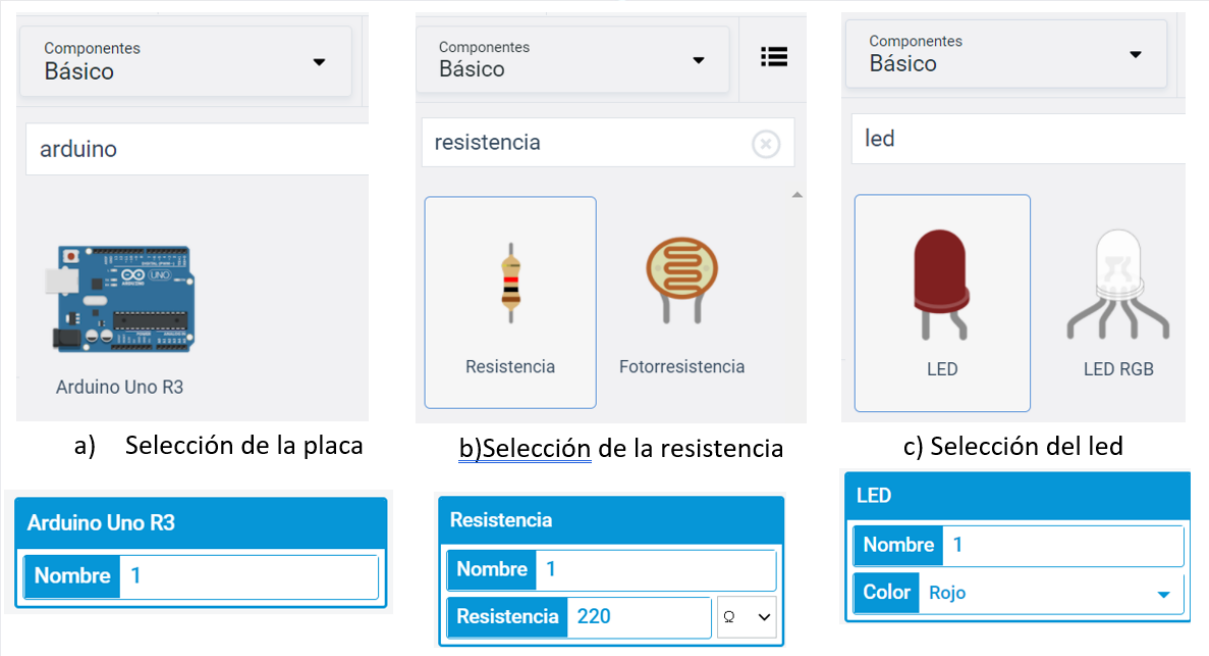


Figura 13. *Selección de componentes*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Para conectar los componentes, sencillamente picamos en el origen y arrastramos hasta el destino. Al final todo ha de estar conectado como se indica en la figura 12. El siguiente paso es escribir el programa. Vamos a utilizar la programación por bloques. Si picamos en el botón de "Código" (figura 18), aparece un editor nuevo. En la parte izquierda se hace la selección del bloque a utilizar y en la parte derecha vamos construyendo el programa (figura 14).

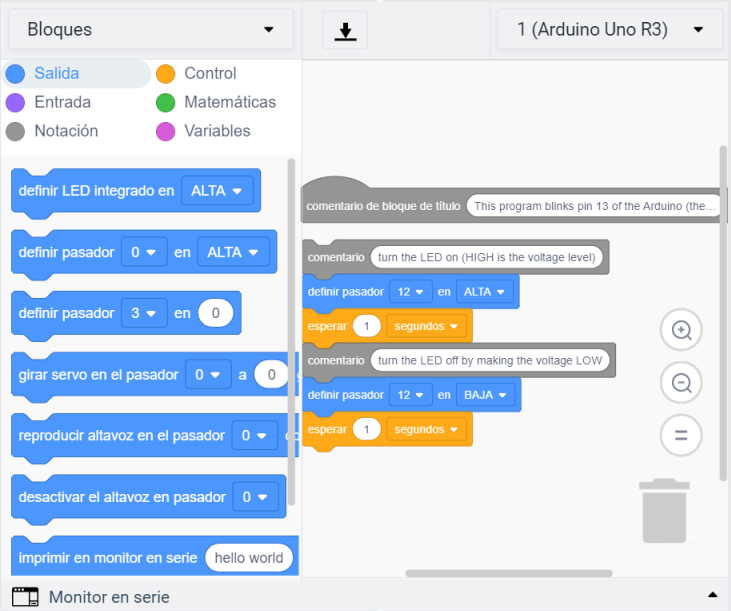


Figura 14. *Programación por bloques.*

Para construir los programas podemos usar bloques relacionados con: salidas, entradas, control, matemáticas, variables y notación (comentarios). A la hora de hacer el programa hay que arrastrar los bloques a la parte de la derecha, según lo que necesitemos hacer.

Vamos a ver paso a paso cómo hacer el programa. Primero queremos encender el led. Para ello hay que indicarle a qué entrada/salida está conectado el led a la placa. Seleccionamos la entrada/salida (pasador) 12 y hay que ponerlo en ALTA (encendido). Se mantiene encendido 1 segundo (esperar 1 segundo).

Figura 15. *Encendido del LED.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

A continuación, lo apagamos y lo mantenemos así otro segundo.

Figura 16. *Apagado del LED.*

El programa completo podemos verlo en la figura 17. Cuando ejecutemos estas instrucciones se van a repetir de forma indefinida hasta que detengamos la ejecución.

Figura 17. *Programa de parpadeo de un LED.*

Si tuviéramos físicamente el hardware, el siguiente paso sería cargar el programa en la placa, a través de un cable USB, y por último observar su funcionamiento. Como estamos utilizando un simulador, inicamos la simulación. Para ello picamos en el botón “Código” para ocultarlo y pulsamos el botón de “Iniciar simulación” (figura 18). Tenemos que observar cómo el led se enciende y apaga cada segundo, de forma indefinida hasta que detengamos la ejecución (figura 19).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

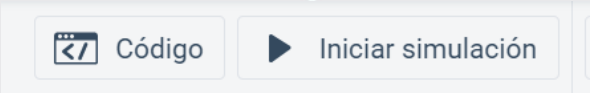


Figura 18. *Botones de Código e Inicio de simulación.*

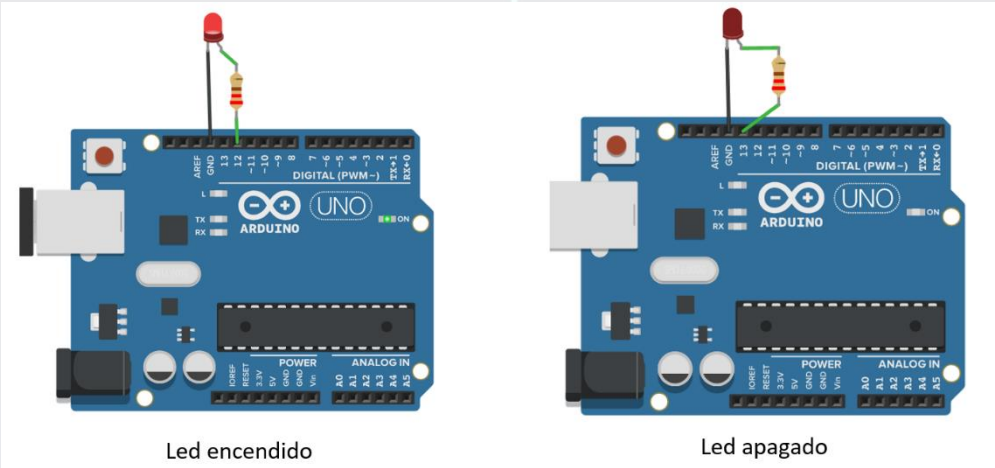


Figura 19. *Resultado de la simulación.*

7. Conclusiones

En esta unidad se han descrito algunas herramientas para desarrollar proyectos sencillos o complejos. Gracias a estas plataformas y a Internet podemos tener más cerca la tecnología, tocarla y experimentar con ella. Hacer cosas prácticas siempre nos hace comprender mejor los conceptos. Con las placas descritas se hacen prototipos para ensayar nuevos desarrollos o utilizarlas para hacer grandes tiradas (series) pero en este último caso se hacen montajes más compactos y miniaturizados.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

8. Bibliografía

[1] SOLECTRO. Placas de desarrollo. https://solectroshop.com/es/10-placas-de-desarrollo
[2] PROJECT HUB. Top 10 Arduino projects. https://www.youtube.com/watch?v=9ItEPmwfBqg
[3] WEB OFICIAL DE ARDUINO https://www.arduino.cc/en/Guide/Introduction
[4] Productos ARDUINO. https://www.arduino.cc/en/Main/Products
[5] ARDUINO UNO R3 https://store.arduino.cc/products/arduino-uno-rev3
[6] Programar fácil. NodeMCU tutorial paso a paso desde cero. https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/
[7] Arduino web editor https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-on-various-platforms-4b3e4a?f=1
[8] Arduino software (IDE) https://www.arduino.cc/en/Guide/Environment
[9] Arduino create agent. https://create.arduino.cc/getting-started/plugin/welcome
[10] Tinkercard. https://www.tinkercad.com/

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Módulo 3. INTERFACES DE USUARIO Y PLATAFORMAS DE DESARROLLO

3.4 EJEMPLOS PRÁCTICOS

Por Beatriz Prieto Campos

Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada.

1. Introducción

En esta unidad vamos a ver con detalle algunos ejemplos prácticos utilizando las herramientas de la unidad 3.3. El objetivo de esta unidad es desarrollar proyectos sencillos de IoT, para comprender mejor su funcionamiento.

2. Ejemplo 1: Control domótico basado en la temperatura

El primer ejemplo es una aplicación domótica, es decir, de Internet de las cosas aplicada al control del hogar. La idea es cambiar el color de la luz de una bombilla según sea el valor de la temperatura leída por un sensor. Este valor, se podría utilizar, además de para cambiar el color de la luz de una bombilla, para interactuar con otros elementos del entorno como encender o apagar la climatización, subir o bajar una persiana un toldo, etc.

En el caso particular de nuestro ejemplo, vamos a considerar que si la temperatura captada por el sensor es inferior a 10 grados la bombilla emitirá una luz azul, si está entre 10 y 40 la luz será amarilla y si es mayor de 40 la luz se pondrá roja.

Una vez definido el objetivo, hay que seleccionar los componentes necesarios:

- Placa ARDUINO UNO R3
- 1 sensor de temperatura
- 1 LED RGB
- 3 resistencias
- Cables para conectar el led, el sensor y las resistencias

El sensor de temperatura lee la temperatura ambiente y transforma ese valor en una señal eléctrica que va al procesador de la placa ARDUINO. El sensor tiene 3 patas, una de ellas hay que conectarla a tierra, otra a la alimentación (Potencia) y la última es la señal eléctrica en la que se ha trasformado el valor de la temperatura (V_{out}), que habrá que conectar a una entrada analógica de la placa, porque el sensor es un dispositivo analógico (figura 1).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

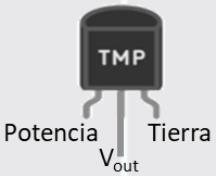


Figura 1. *Sensor de temperatura.*

Un LED RGB que es un tipo de led que emite el color resultante de la suma de una cierta cantidad de rojo, verde y azul. Este LED tiene 4 patas. Una va a tierra (cátodo) y, a través de las otras tres patas se le da la cantidad de rojo, verde y azul (figura 2). Para que no le llegue demasiada corriente al led y se funda, es necesario utilizar una resistencia. En esta ocasión hay que colocar una resistencia en cada pata (salvo la del cátodo).

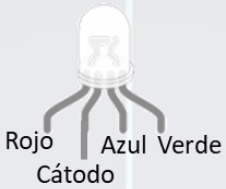


Figura 2. *LED RGB.*

Para simular este ejemplo vamos a utilizar Tinkercad, ya que no necesitamos disponer de ningún material físico (hardware). Escribimos la URL de Tinkercad en el navegador (http://www.tinkercad.com/) y nos identificamos para poder acceder a nuestra cuenta con nuestros circuitos y creamos uno nuevo (figura 3).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

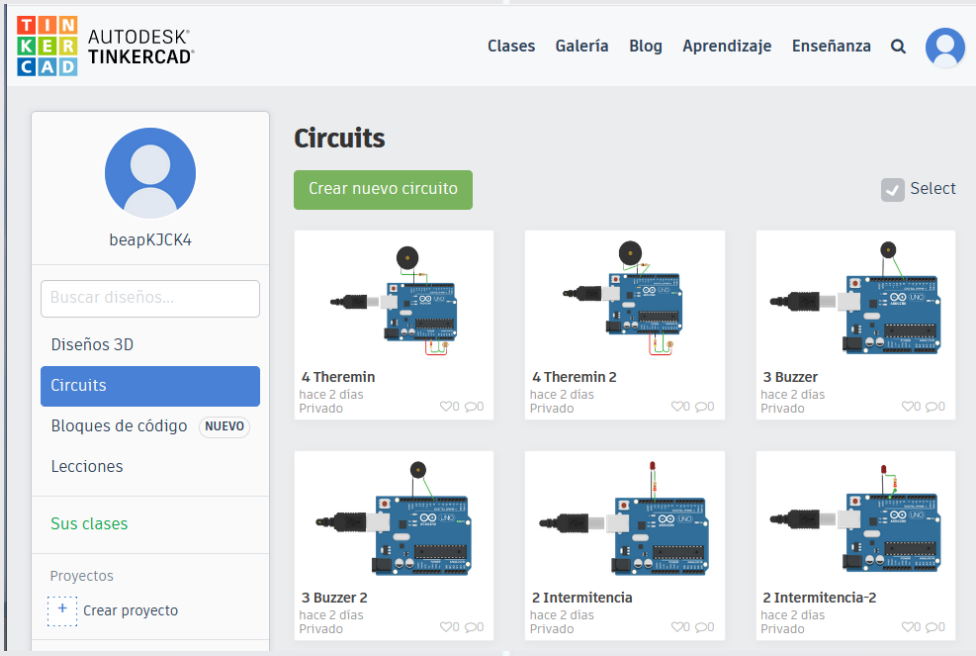


Figura 3. Entorno Tinkercad.

A continuación, hay que ir seleccionando los diferentes componentes del menú de la derecha y se van incorporando al diseño. En la figura 4 muestra el proceso que hay que hacer para encontrar cada uno de los componentes.

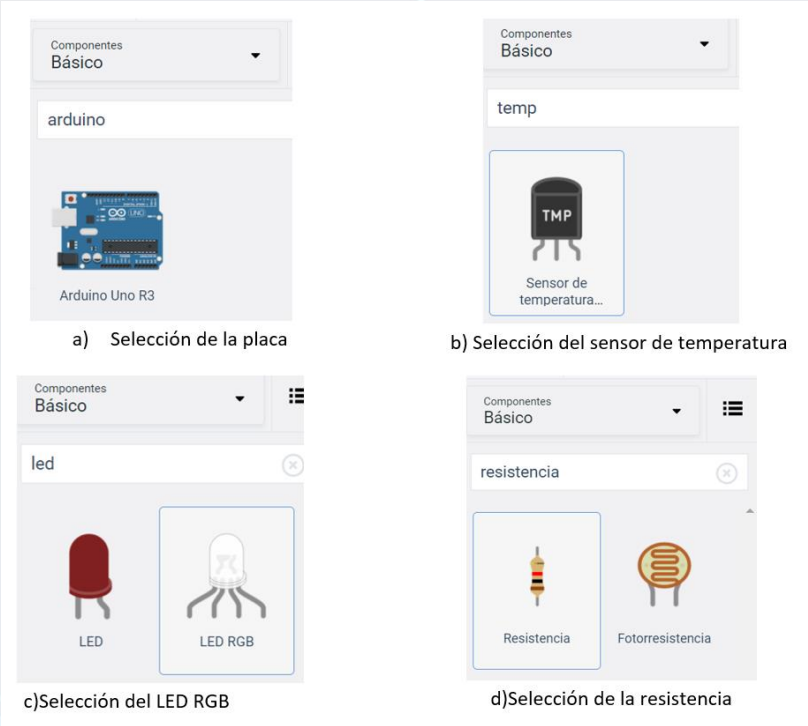


Figura 4. *Selección de componentes en Tinkercad*

UNIVERSIDAD DE GRANADA
abiertaugr
la formación abierta de la UGR
CC BY NC ND

A continuación, hay que conectar los componentes tal y como se indica en la figura 5. En el sensor de temperatura hay que conectar la pata de alimentación (potencia) al pin de 5V de la placa, la pata V_{out} a la entrada A0 y la tierra a GND. En el LED RGB hay que conectar la tierra a GND, y cada una de las otras tres patas a una resistencia. La salida de la resistencia que está conectada a la pata "rojo" se conecta a la salida digital D11, la verde a la D9 y la azul a la D10.

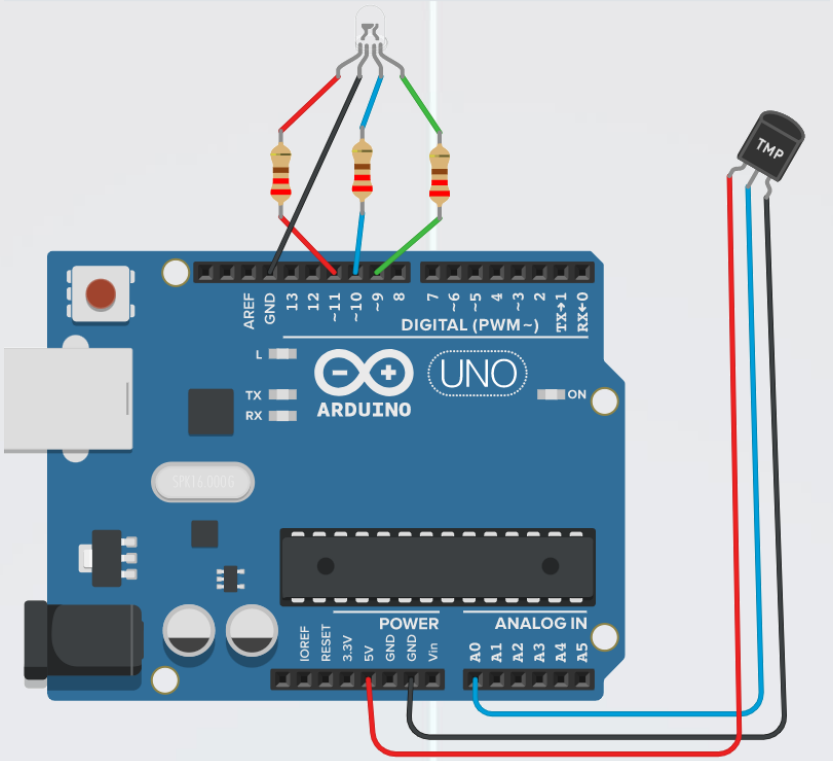


Figura 5. *Conexión de los componentes a la placa.*

El siguiente paso es escribir el programa. Vamos a utilizar la opción de programación visual por bloques. Recordemos que tenemos bloques para especificar operaciones de entrada (Entrada), operaciones de salida (Salida), acciones de control (Control), operaciones matemáticas (Matemáticas), definir variables (Variables) y poner comentarios (Notación).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

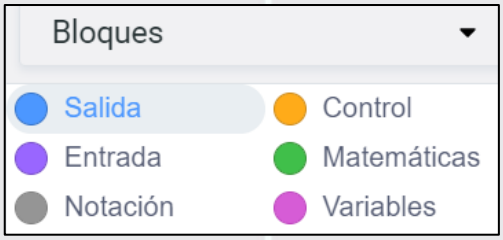


Figura 6. Tipos de bloques para escribir un programa.

Vamos a ir construyendo el programa poco a poco hasta conseguir nuestro objetivo. Primero se define el LED RGB. Como es una salida, seleccionamos bloques de salida y escogemos el bloque de definición de LED RGB. Tenemos que indicarle a qué salidas está conectado el LED. La pata rojo (R) a D11, la verde (G) a D9 y la azul (B) a D10 y el color por defecto será verde (figura 7).

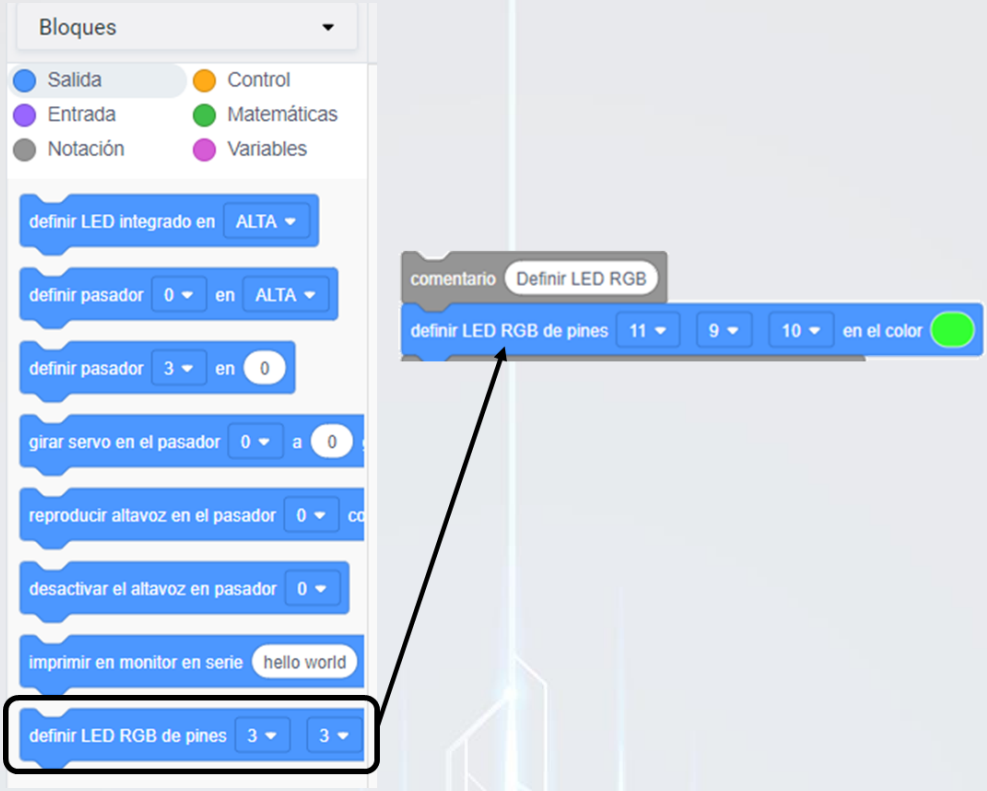


Figura 7. *Definición del LED.*

Es necesario definir una variable, que podemos llamar ValorTemp, para guardar el valor de la temperatura. La definimos utilizando el bloque de Variables, y le indicamos que queremos leer el sensor de temperatura que está conectado a la entrada (pasador) A0 y que el valor se de en grados centígrados (figura 8).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

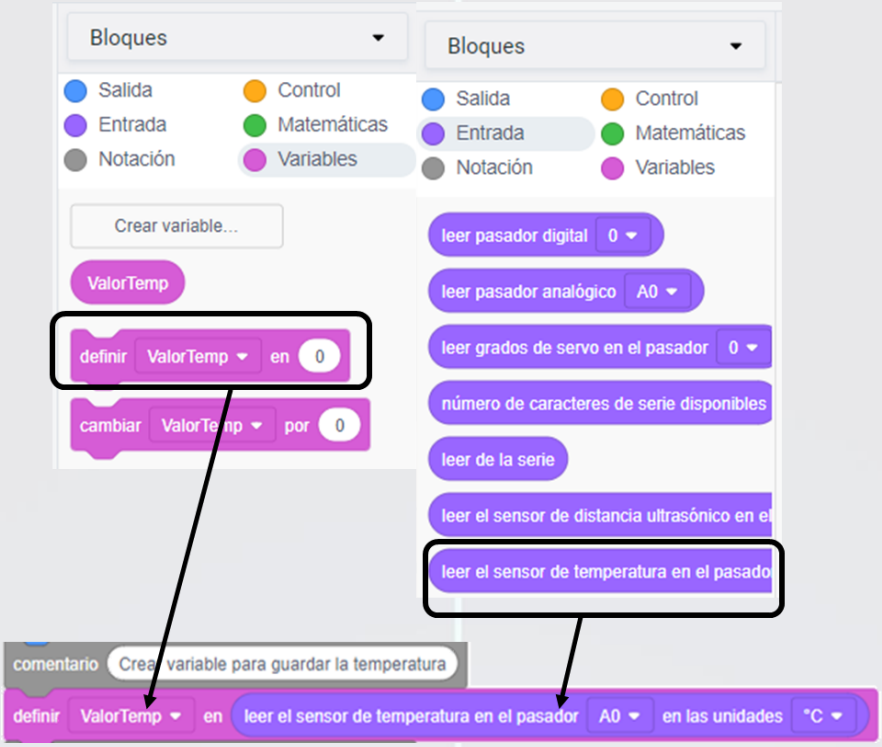


Figura 8. *Definición de la variable de temperatura.*

Una vez almacenado el valor de la temperatura actual en nuestra variable, tenemos que ver en qué rango se encuentra, para que el led se ponga de un color u otro. Es decir, según el valor de la temperatura hay que hacer una cosa diferente. Lo primero es comprobar si la temperatura es inferior a 10 grados, y en ese caso, la luz debe ponerse azul. Para expresar las condiciones hay que utilizar un bloque de control "si…entonces…". Como hay que hacer una comparación matemática (comprobar si ValorTemp <10) vamos al bloque de Matemáticas y seleccionamos el bloque de comparación. A continuación le decimos que queremos comparar la variable ValorTemp con 10 y, por último añadimos el bloque de salida para indicar que el led RGB se ponga azul (figura 9).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

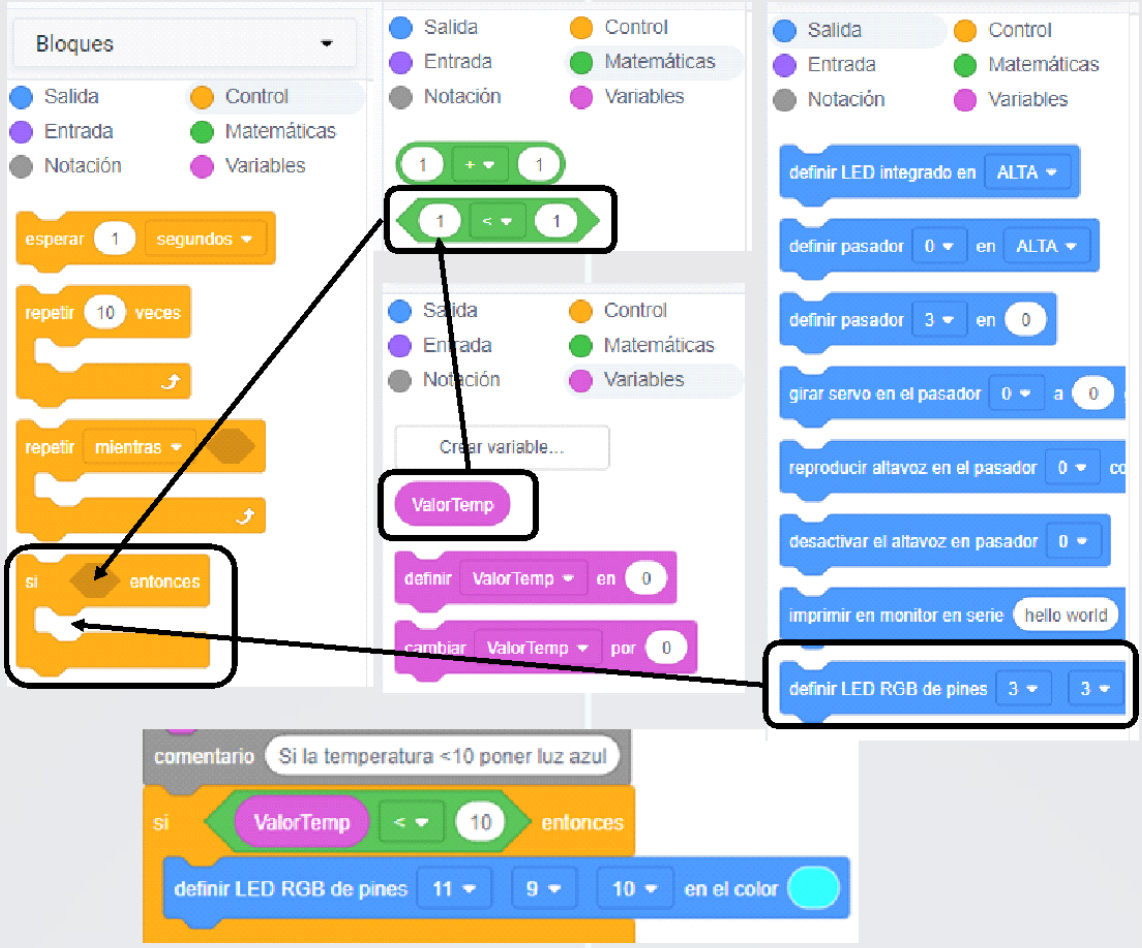


Figura 9. *Si ValorTemp < 10 poner el led azul.*

En el siguiente intervalo tenemos que comparar con dos valores, por lo que hay que utilizar el bloque matemático que contiene "y". Con este bloque y el de comparación expresamos que si ValorTemp ≥ 10 y ValorTemp ≤ 40, hay que poner el led amarillo (figura 10).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Figura 10. *Si 10 ≥ ValorTemp ≤ 30 poner el led amarillo.*

Por último, de forma similar al primer caso, le decimos que, si ValorTemp es mayor que 40, se ponga la luz roja (figura 11).

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

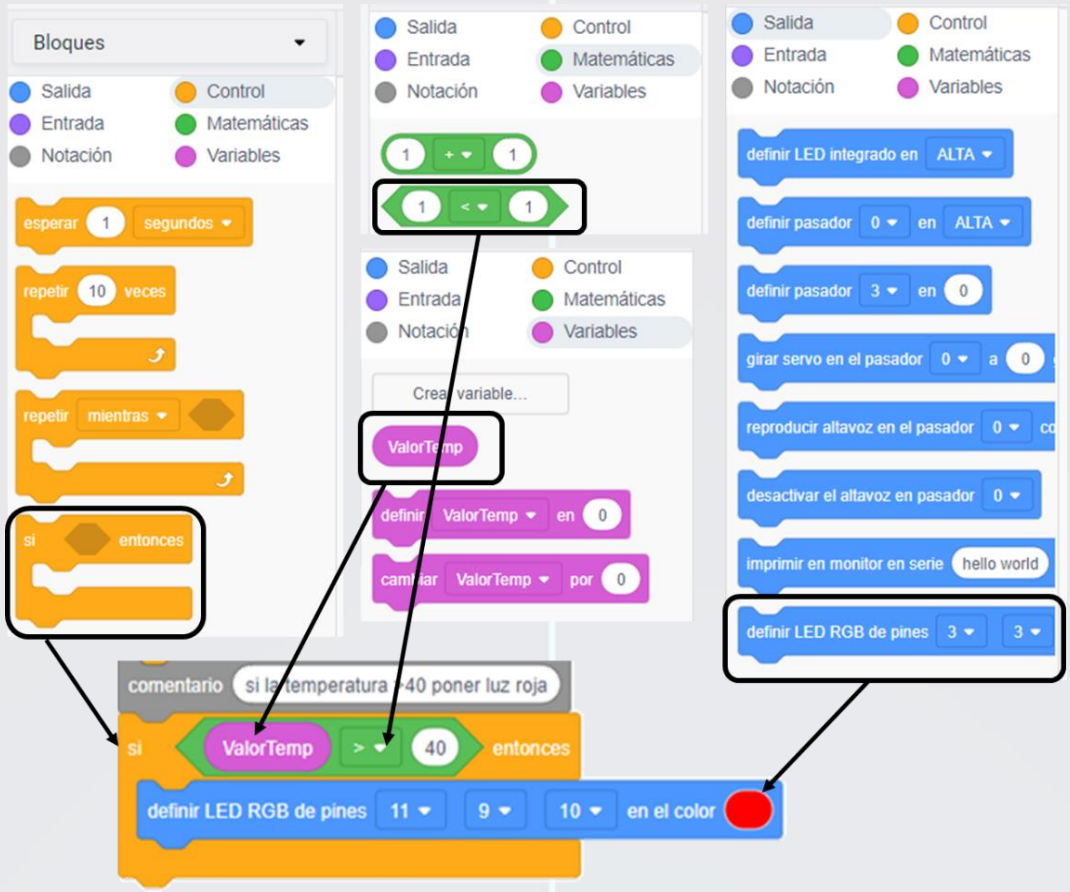


Figura 11. *Si ValorTemp > 40 poner el led rojo.*

En la figura 12 se puede ver el programa completo.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

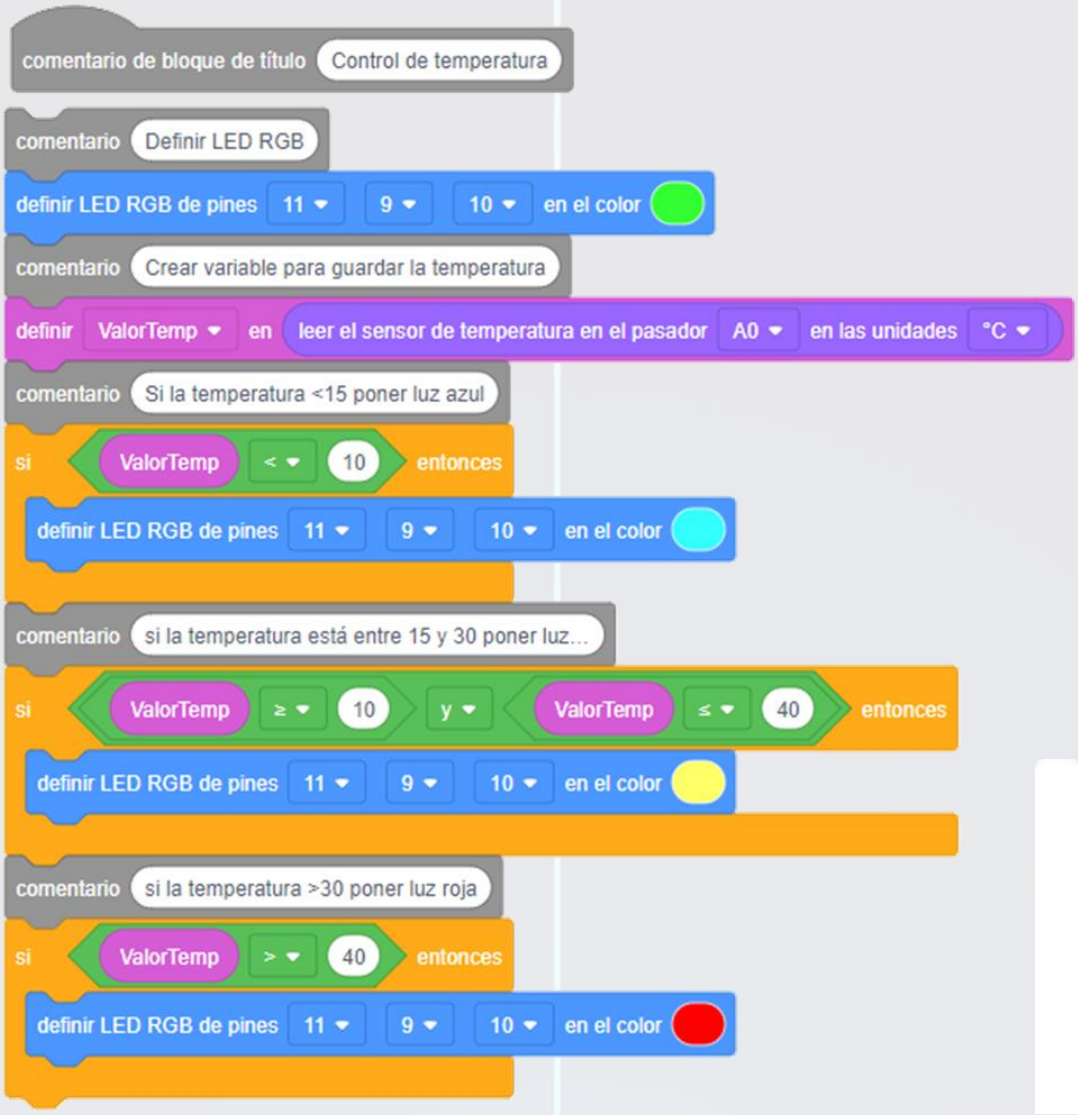


Figura 12. *Programa completo.*

El último paso es simular la ejecución y comprobar que funciona correctamente. Para ello, ocultamos el código y pulsamos en Iniciar simulación (figura13).

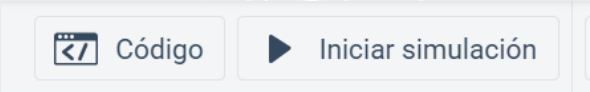


Figura 13. Botones de Código e Inicio de simulación.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Durante la comprobación hay que ir variando el regulador de temperatura (figura 14) para simular cambios de temperatura y verificar que el color del led cambia según sea este valor.

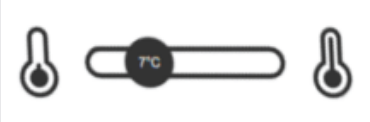


Figura 14. *Regulador de temperatura.*

En la figura 15 se observa el resultado de la simulación. Cuando la temperatura es de 7 grados (figura 15a), que es menor que 10, la luz es azul. Cuando la temperatura es de 26 grados (figura 15b), como está entre 10 y 40, es amarilla y, por último, como la temperatura es de 57 grados (figura 15c), que es superior a 40 grados, el led estará rojo.

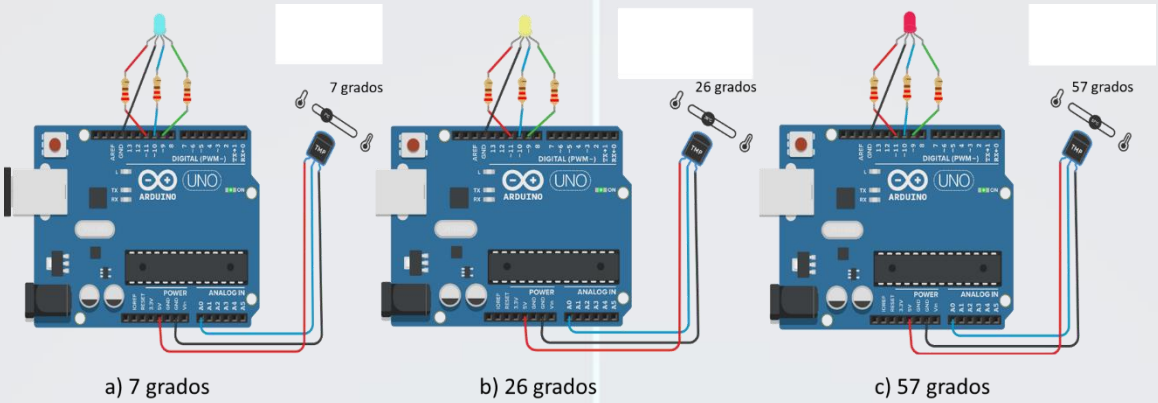


Figura 15. *Resultado de la simulación.*

3. Ejemplo 2: Control de un led a través de la Wifi

En este segundo ejemplo estudiaremos otro proyecto de IoT que consiste en controlar el apagado y encendido de un led, vía WIFI, a través de una aplicación instalada en nuestro Smartphone.

Figura 16. Control de luz vía WIFI.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Se trata de instalar y configurar la aplicación del teléfono, montar la placa con el led, escribir el programa y, por último, comprobar que funciona.

Para este proyecto necesitamos:

- Instalar y configurar una aplicación en nuestro Smartphone
- Placa NODEMCU V3, un led, una resistencia, un par de cables, una placa de trabajo y un cable USB para conectar la placa con el ordenador.
- Entorno de desarrollo de programas como Arduino IDE

Instalación y configuración de la aplicación en el Smartphone

Para este proyecto necesitamos un Smartphone que va a ser la interfaz a través de la que controlaremos el encendido y apagado de una luz (led) con una conexión WIFI. Para poder hacerlo hay que instalar una aplicación que permita enviar la orden de encendido o apagado desde nuestro teléfono a la placa, donde está conectado el led. La aplicación que vamos a utilizar es Blynk [2]. Blynk es una plataforma diseñada para IoT y permite controlar hardware de forma remota, haciendo uso de diferentes tecnologías de conexión inalámbrica o alámbrica (Ethernet, WIFI, USB, GSM, Bluetooth y BLE).

¿Cómo funciona Blynk? Vamos a explicarlo con nuestro ejemplo. La idea del proyecto es crear un botón, con la App Blynk, que al pulsarlo encienda y apague un led. Cuando esto ocurra, se enviará un mensaje que va a la nube de Blynk. Desde allí se busca nuestra placa y se le envía un mensaje para que se encienda o se apague el led. Es decir, Blynk es una plataforma que permite conectar y controlar dispositivos a través de la nube, de forma transparente al usuario.

Figura 17. *Comunicación Smartphone – placa vía BLYNK.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

En primer lugar, vamos a instalar y configurar la aplicación [3]. Buscamos BLYNK en la App Store o en Google Play, según tengamos un teléfono Apple o Android. Una vez instalada la aplicación hay que crear una cuenta (figura 18).

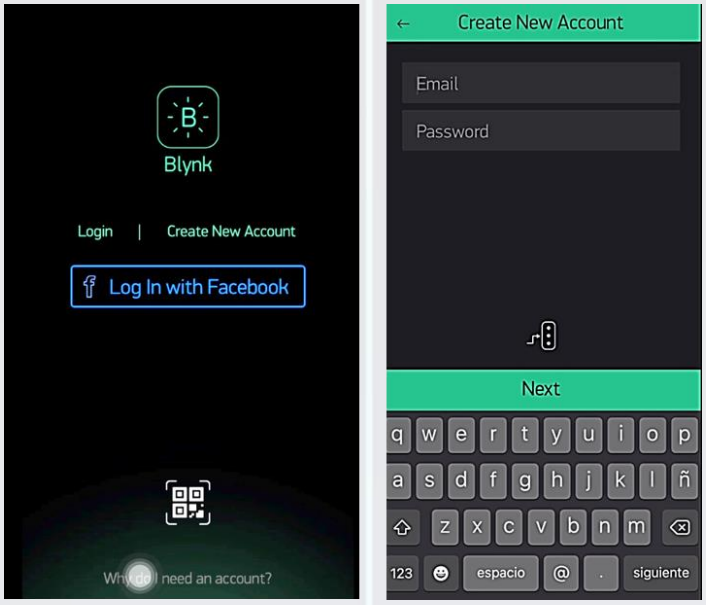


Figura 18. *Creación una cuenta en la App BLYNK.*

¿Cómo sabe la nube de Blynk cuál es mi placa? Cuando creamos una cuenta nos envían un correo electrónico con lo que denominan un "token de autenticación" (figura 19), que consiste en un código único que identificará nuestro hardware y que luego utilizaremos en el programa que carguemos en la memoria de nuestra placa. De esta forma, lo que hace Blynk es buscar el hardware que contenga ese código.

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
BY NC ND

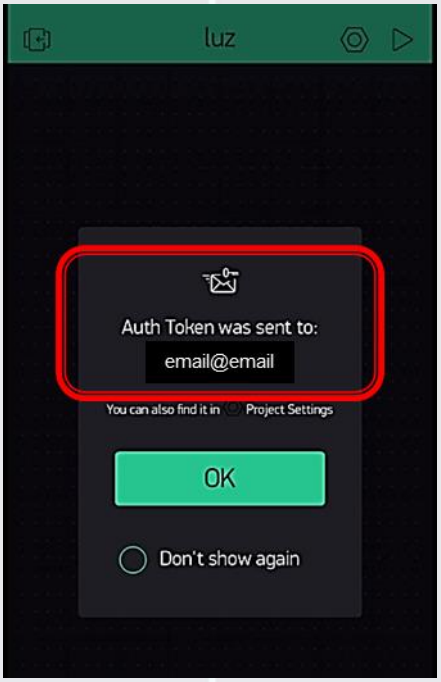


Figura 19. *Envío del Auth Token.*

Una vez abierta la aplicación, lo primero es crear un nuevo proyecto (figura 20). A continuación, le asignamos un nombre, seleccionamos el nombre de la placa, que en nuestro caso será NodeMCU y que dispone de un módulo WIFI. También hay que indicar el tipo de conexión: WIFI.

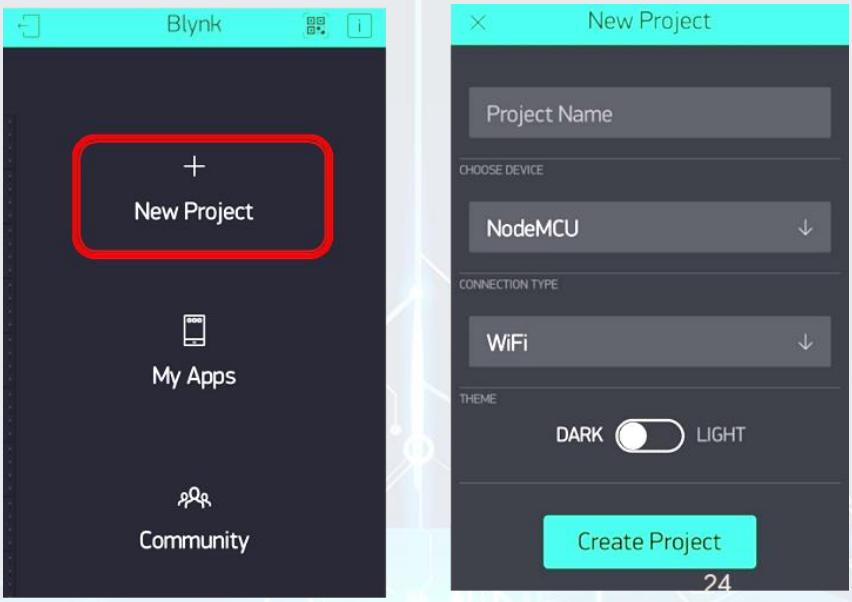


Figura 20. *Creación de un nuevo proyecto en la App BLYNK.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Creado el nuevo proyecto, entramos en el entorno que nos permite configurar el aspecto de la interfaz. En la figura 21 se pueden observar las diferentes opciones del editor. Hay un botón para ver el listado de proyectos que hemos creado o salir de la App. Hay otro botón de configuración, y otro para cambiar entre diseño y ejecución de la interfaz. Por último, tenemos la pantalla central, que al tocar muestra un listado de los elementos que podemos añadir a nuestra interfaz.

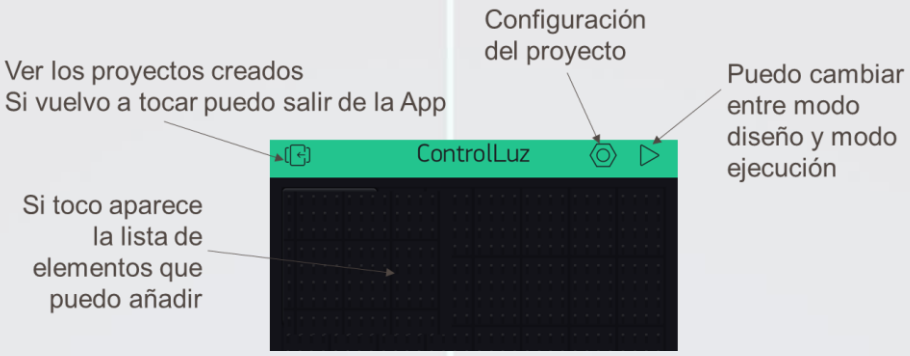


Figura 21. *Entorno de desarrollo de la interfaz.*

Lo único que necesitamos para nuestro proyecto es añadir un botón que simule apagar y encender la luz. Tocamos la pantalla principal y aparecen los elementos que podemos añadir. Seleccionamos "Button", lo arrastramos y colocamos en la zona de la pantalla que queramos. También se puede cambiar de tamaño. Una vez insertado el botón, si lo tocamos, es posible configurar su color, el tipo (push o switch) y, lo que es muy importante, la entrada de la placa a la que va a estar conectado el led, para mandar a esa entrada de la placa la orden de encender o apagar (figura 22). Si ponemos una entrada diferente, no funcionará.

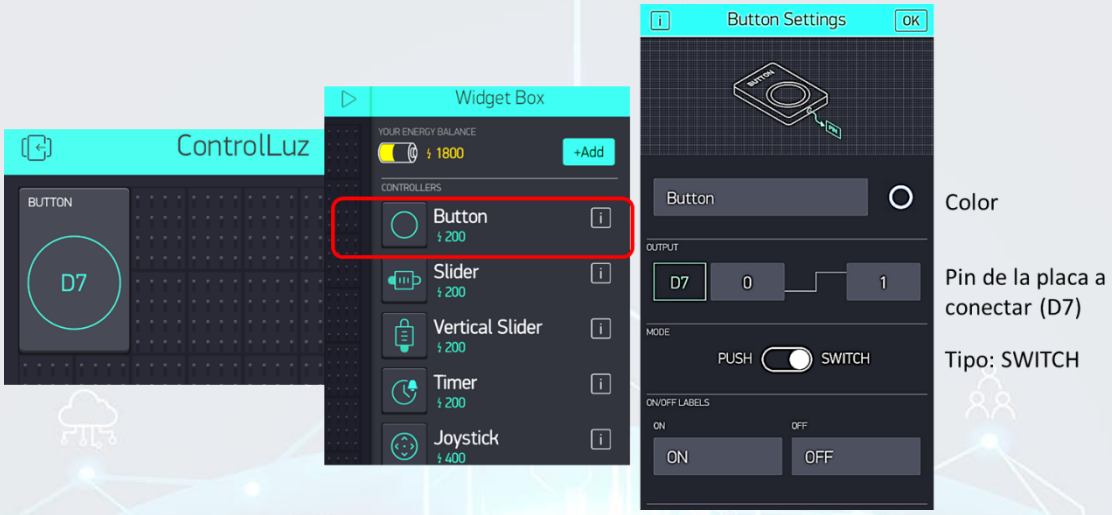


Figura 22. *Selección y configuración del botón.*

UNIVERSIDAD DE GRANADA
abiertaugr
la formación abierta de la UGR
BY NC ND

Ya tenemos lista la aplicación en el teléfono.

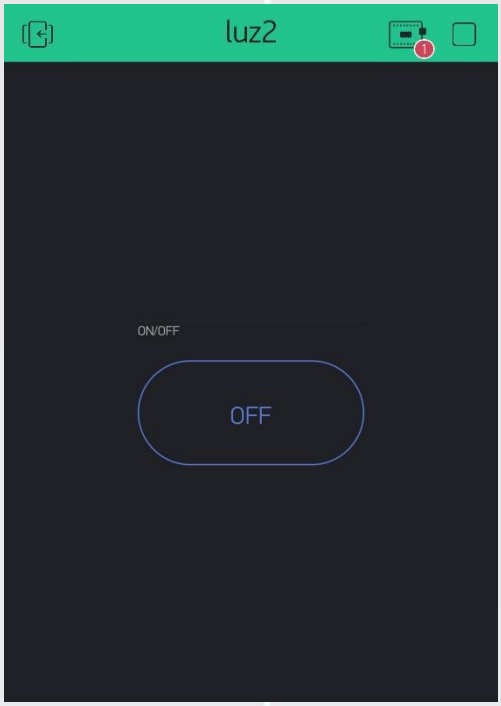


Figura 23. *Aspecto de la interfaz de la App.*

Montaje del hardware y redacción del programa.

El montaje hardware es muy sencillo. Tal y como se ve en la figura 24, sólo necesitamos conectar la placa NODEMCU V3 con el módulo WIFI a un led con una resistencia igual que la que hemos utilizado en otros ejemplos. Una de las patas del led irá a tierra y la otra a la resistencia que a su vez irá a la salida D7, tal y como hemos configurado la App Blynk.

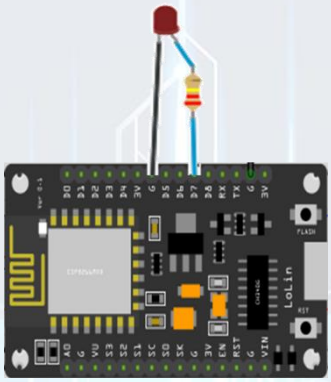


Figura 24. *Montaje hardware.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

A continuación, hay que escribir el programa. En este caso no se puede hacer con bloques por lo que hay que escribirlo en un lenguaje de alto nivel. Para ello se ha utilizado ARDUINO IDE (figura 25). El programa tiene varias partes. Inicialmente, se incluyen unas llamadas a las librerías que contienen las funciones necesarias para hacer uso de la comunicación WIFI de la placa. Hay que incluir el token de autenticación que identifica nuestra placa y permite la comunicación con ella. Luego hay que indicar el nombre del router o punto de acceso que va a dar la cobertura WIFI así como la contraseña para acceder a la misma. El resto consiste en abrir el monitor serie para ver una serie de mensajes de Blynk que indican si se ha establecido la conexión y, por último, la llamada a la nube de Blynk para enviar y recibir los mensajes necesarios para encender o apagar el led.

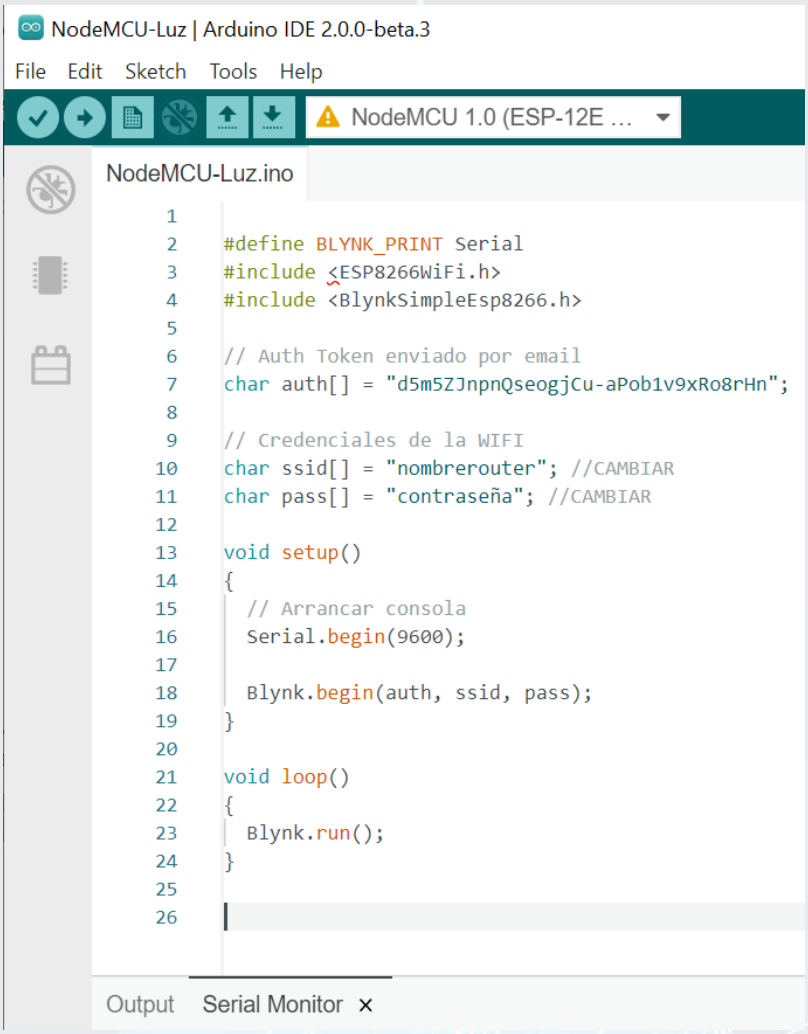


Figura 25. *Programa en Arduino IDE.*

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND

Lo importante de este programa es que toda la gestión de la comunicación entre la App del móvil y nuestro hardware es totalmente transparente a nosotros, es un servicio que Blynk hace por nosotros facilitando enormemente la labor de conexión entre la interfaz, en este caso el Smartphone, y el hardware (la placa con el led).

El último paso es conectar la placa al ordenador mediante un cable USB para transferir el programa. Si todo ha ido bien, en el monitor serie de Arduino IDE debe aparecer un mensaje de confirmación de conexión (figura 25).

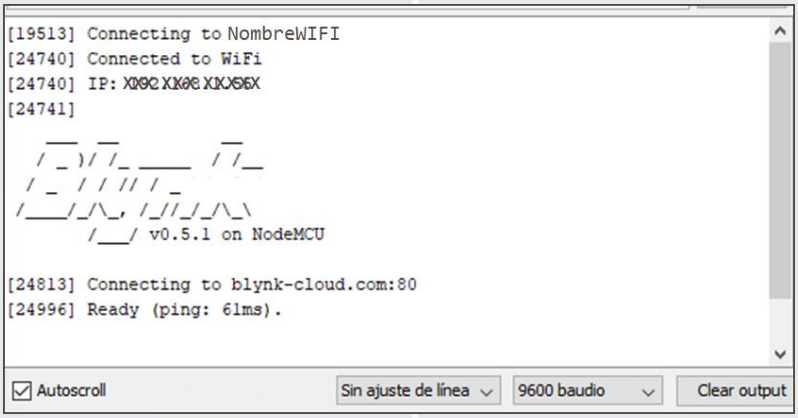
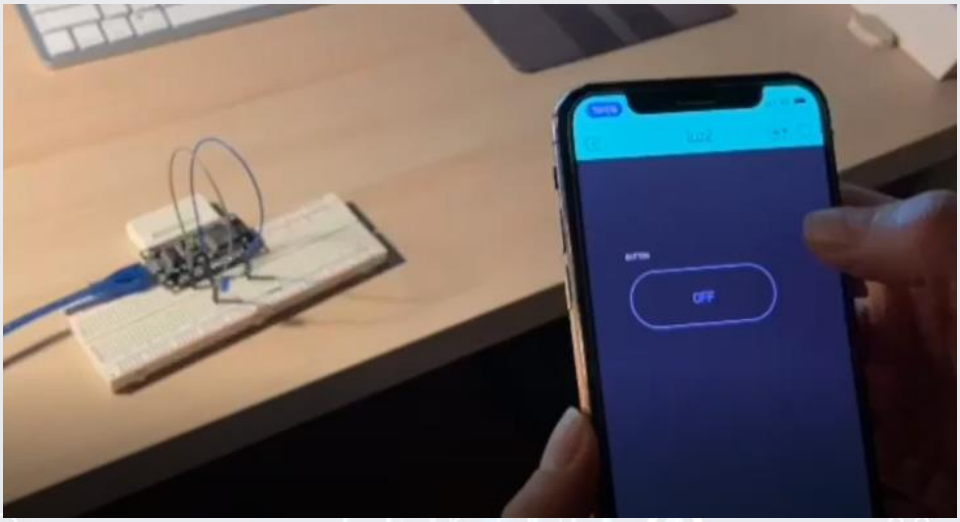


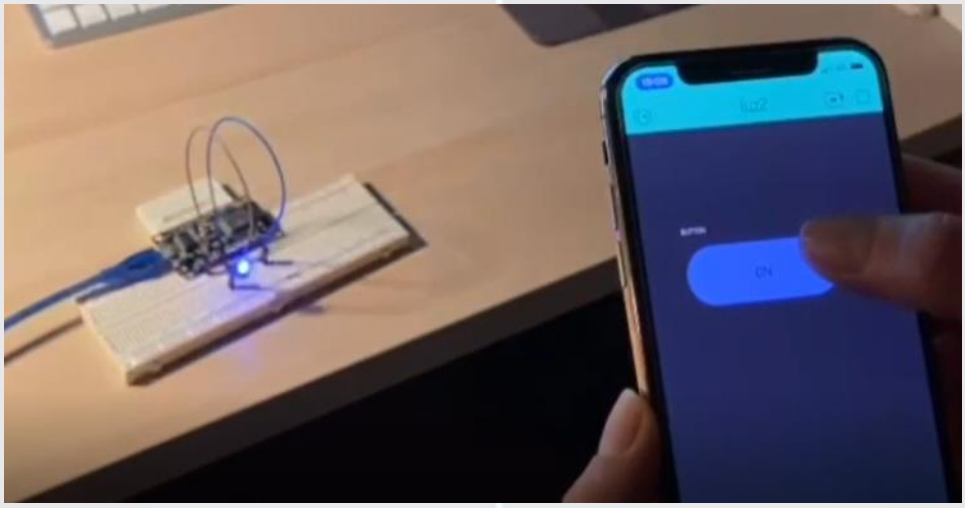
Figura 26. *Mensaje del monitor serie.*

Una vez establecida la conexión cuando pulsemos el botón de nuestra App se encenderá o apagará el LED, y lo podremos hacer desde cualquier lugar en el que tengamos cobertura de esa WIFI (figura 25).



a) Botón OFF, led apagado.

UNIVERSIDAD DE GRANADA
abiertaugr
la formación abierta de la UGR
BY NC ND



b) Botón ON, led encendido.

Figura 27. *Funcionamiento del encendido/apagado de un led vía WIFI.*

4. Conclusiones

En esta unidad se han descrito con detalle el desarrollo de dos proyectos sencillos de IoT. En el primer ejemplo se lleva a cabo una acción a partir de la información obtenida de un sensor. En concreto, según sea la temperatura hacemos que un led adopte un color diferente. Este mismo procedimiento se puede aplicar a multitud de circunstancias diferentes, en las que, a partir de valores suministrados por sensores, se actúa sobre algún objeto.

En el segundo ejemplo, ha sido posible controlar un objeto cotidiano, un led, desde una aplicación en el móvil, a través de un servidor en la nube con una conexión inalámbrica WIFI.

En ambos casos, se constata que IoT puede ser algo cercano, al alcance de cualquiera, existiendo infinidad de herramientas que nos ayudan a que todo este proceso sea mucho más sencillo de lo que imaginamos.

5. Bibliografía

[1] Plataforma Tinkercard. https://www.tinkercad.com/
[2] Aplicación BLYNK https://blynk.io/
[3] Instalación y configuración de BLYNK https://blynk.io/en/getting-started

UNIVERSIDAD DE GRANADA
abiertaugr la formación abierta de la UGR
CC BY NC ND