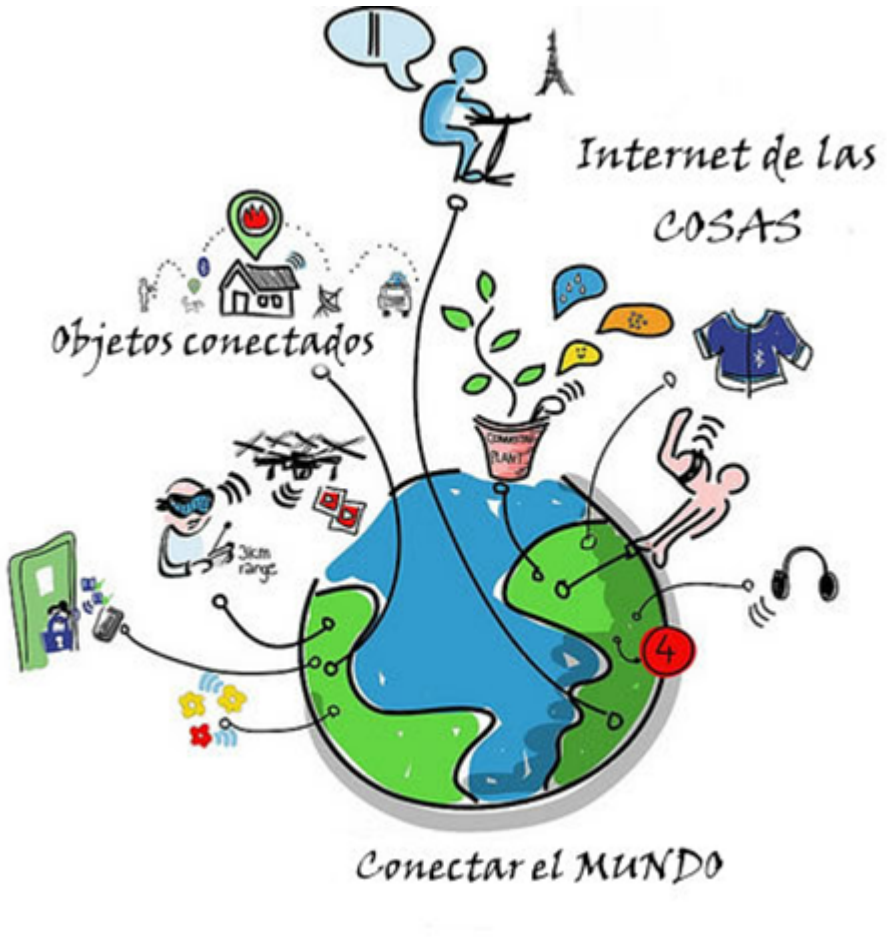
Internet of Thinks (IoT)

¿Qué es IoT?

Se le llama **Internet de las cosas**, en inglés **Internet of Things (IoT)** a la posibilidad de interconexión y transmisión de datos entre objetos cotidianos e internet.



Los aparatos electrónicos y los dispositivos digitales cotidianos tienen circuitos y sensores que les permite recolectar y compartir datos con la internet sin la intervención de personas.

¿Cómo funciona el IoT?

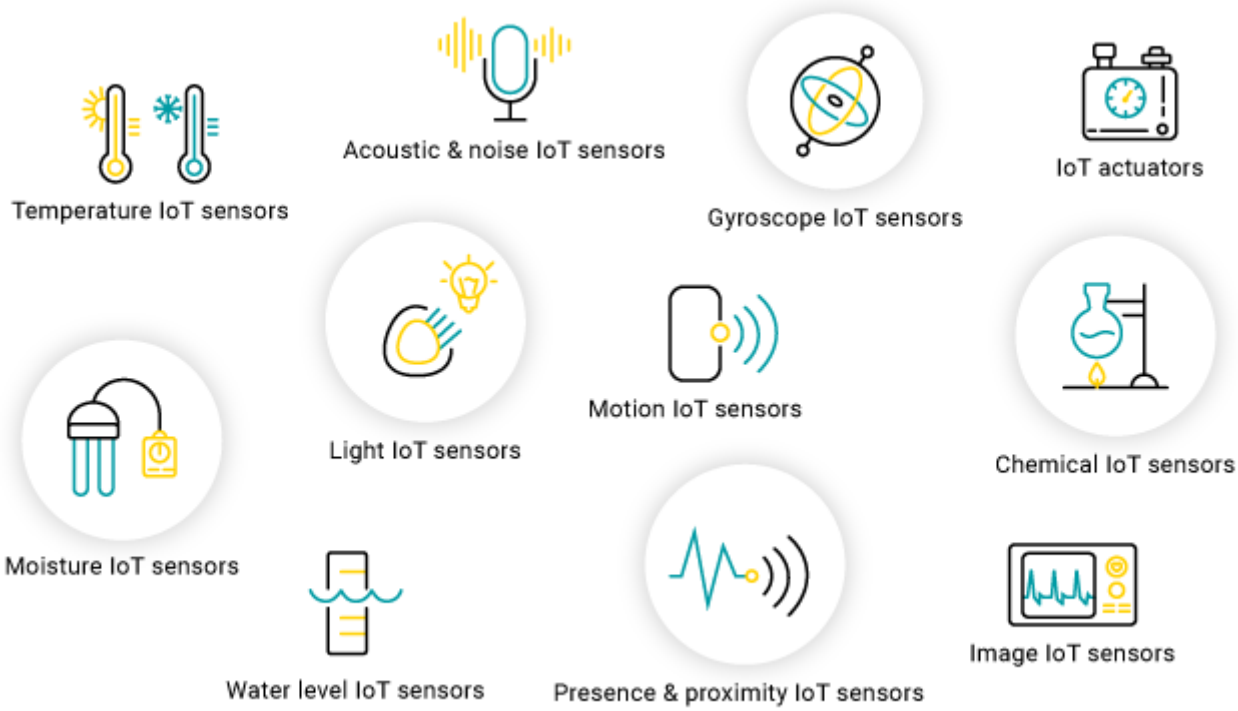
Los **dispositivos IoT** se conectan con un proceso llamado M2M (machine to machine, o máquina a máquina) en el que dos dispositivos o máquinas cualesquiera se comunican entre sí utilizando cualquier tipo de conectividad (que puede ser cable, WiFi, Bluetooth, etc.), haciendo su trabajo sin la necesidad de que un humano intervenga. No deja de ser el mismo concepto que las conexiones Peer to Peer de los ordenadores para jugar online o compartir archivos.

Entonces nos encontramos que un dispositivo IoT recolecta, procesa y analiza una gran cantidad de información, entre la que se incluye información sobre los hábitos y preferencias de consumo de cada uno de los usuarios, así como por ejemplo poder gestionar su salud u otras utilidades para el día a día que no serían posibles si. Como, por ejemplo, si hay algún fallo en tu coche, el ordenador a bordo del mismo puede enviar un aviso para que lo lleves al taller.

Un ejemplo de dispositivo IoT es el altavoz inteligente Amazon de Alexa, el cual se encuentra conectado a la red de redes para obtener información e interactúa con otros dispositivos para darnos la capacidad de controlarlos a través de simples comandos de voz.

Dispositivos IoT

Un dispositivo IoT consiste en un **objeto** al que se le ha dotado de conexión a Internet y cierta inteligencia software, sobre el que se pueden medir parámetros físicos (mediante **sensores**) o actuar remotamente (**actuadores**) y que por tanto permite generar un ecosistema de servicios alrededor del mismo.



Para lograr que sensores y actuadores recolecten y envíen datos del mundo físico a Internet, los dispositivos IoT cuentan con un componente escencial: el microcontrolador. Los microcontroladores son **Sistemas Embebidos de Control** diseñados para realizar funciones dedicadas, normalmente en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general, que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas y por tanto forman parte de multitud de productos en los que se requiere una cierta potencia de proceso como electrodomésticos, vehículos, máquinas, etc.

En el link historia microprocesadores se muestra una referencia detallada desde el comienzo de los microprocesadores hasta la aparición de los microcontroladores al mercado. Este material, diseñado por el ing. Hugo Pailos, permite comprender cómo la tecnología de los microcontroladores fue evolucionando hasta lo que conocemos en la actualidad.

Ventajas del IoT

Algunas de las ventajas mas significativas del IoT:

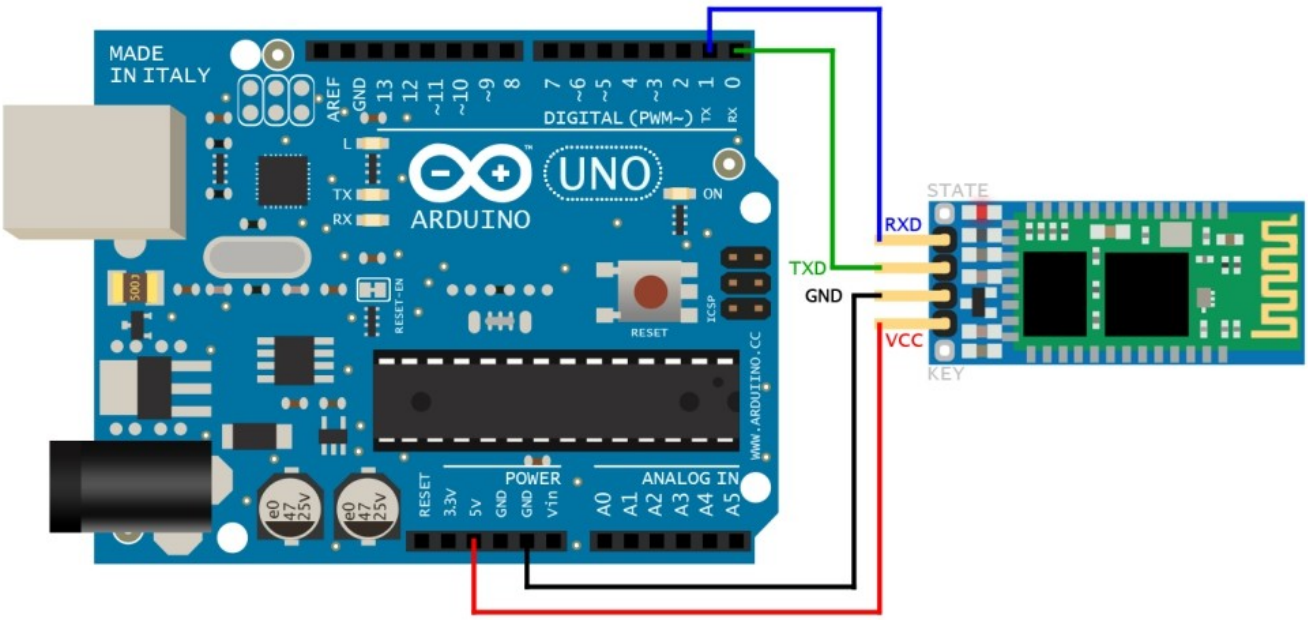
- **Conectividad y Comunicación**: El IoT permite la conexión de dispositivos y sistemas en red, lo que facilita la comunicación y la transferencia de datos en tiempo real entre ellos.

- **Automatización y Eficiencia**: Los dispositivos IoT pueden recopilar datos, analizarlos y tomar decisiones automáticas, lo que lleva a una mayor eficiencia operativa y a la automatización de tareas.
- **Monitoreo Remoto**: El IoT permite el monitoreo y control remoto de dispositivos y sistemas, lo que facilita la supervisión y gestión de procesos desde cualquier ubicación.
- **Mejora de la Productividad**: Al automatizar tareas repetitivas y optimizar procesos, el IoT puede mejorar la productividad en diversas industrias y actividades.
- **Personalización y Experiencia del Usuario**: El IoT permite la personalización de servicios y experiencias para los usuarios, adaptándose a sus necesidades y preferencias individuales.
- **Toma de Decisiones Basada en Datos**: Los datos recopilados por los dispositivos IoT proporcionan información valiosa para tomar decisiones informadas, lo que lleva a una toma de decisiones más precisa y efectiva.

Microcontroladores ARDUINO

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

Para poder entender este concepto, primero vas a tener que entender los conceptos de hardware libre y el software libre. El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas pero igualmente funcionales al partir de la misma base.



El **Arduino** es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

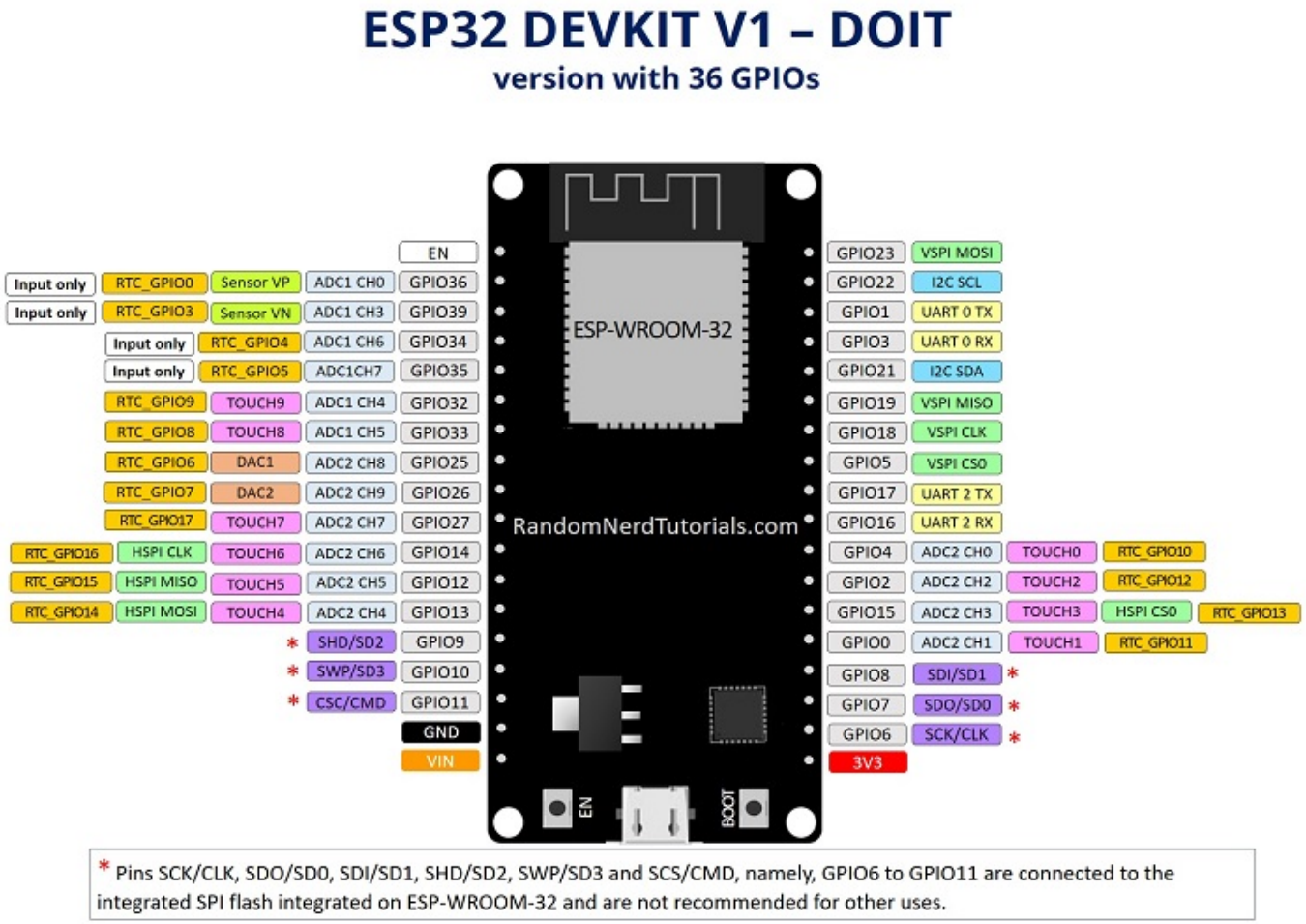
El tipo de periféricos que puedas utilizar para enviar datos al microcontrolador depende en gran medida de qué uso le estés pensando dar. Pueden ser cámaras para obtener imágenes, teclados para introducir datos, o diferentes tipos de sensores.

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados, pero también pueden ser otras placas o controladores.

Placas Esp32

ESP32 es una familia de microcontroladores de la empresa **Espressif Systems**. Su analogía más clara es la de un Arduino esteroides, que incluye Wifi, Bluetooth, altas velocidades de procesamiento, mayor capacidad de almacenamiento de datos, entre otras potentes características extra.

ESP32 es un microcontrolador de bajo costo y bajo consumo de energía conocido bajo la categoría de *SoC* (System on Chip)



Para mayor información consultar la hoja de datos (datasheet) del fabricante aquí: https://www.electrosoftcloud.com/wp-content/uploads/2021/04/esp32-wroom-32_datasheet_en.pdf.

Arquitectura del ESP32

Tensión de trabajo

El voltaje de funcionamiento de los microprocesadores ESP es de **3,3 V** en comparación con el voltaje de funcionamiento de Arduino de **5 V**. Si las placas se utilizan mientras están conectadas a la toma de corriente, no habrá diferencia en el consumo de energía porque la corriente se reducirá para recolectar la misma cantidad de energía. En el caso de un caso de uso alimentado por batería, la diferencia será mucho mayor porque si la curva de descarga de la batería cae por debajo del voltaje de funcionamiento, el microprocesador se apagará.

Por lo tanto, las placas basadas en ESP tendrán un tiempo de ejecución más prolongado, ya que estas placas podrían funcionar a 4 V, mientras que a 4 V las placas Arduino se apagaron.

Memorias: FLASH, SRAM, ROM y EEPROM

- La memoria flash (espacio del programa), es donde se almacena el programa escrito en C/C++ (hasta 4MB). También almacena el firmware del Sistema Operativo (RTOS).
- SRAM (memoria estática de acceso aleatorio) es donde el boceto crea y manipula variables durante la ejecución (hasta 520k).
- EEPROM es el espacio de memoria que los programadores pueden usar para almacenar información a largo plazo. La memoria flash y EEPROM no son volátiles (la información persiste después del apagado). SRAM es volátil y se perderá al encenderlo.
- ROM de 448 KB, para el bootloader del sistema operativo y otros códigos esenciales de sólo lectura.

Velocidad de reloj

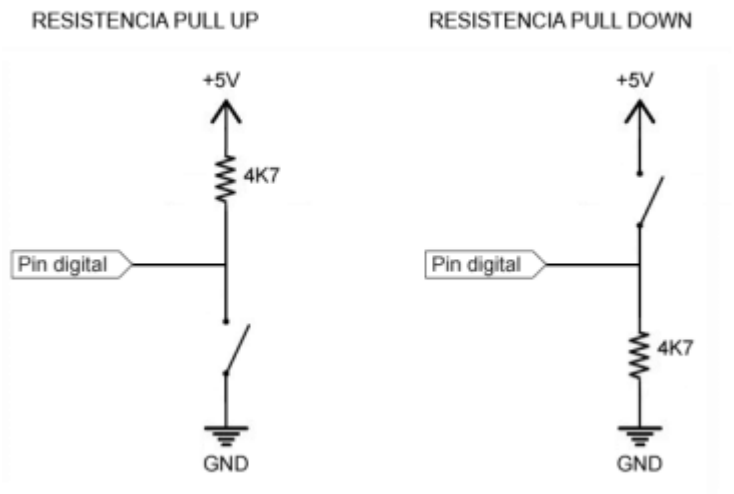
Todas las placas Arduino funcionan a 16 MHz, lo que significa que el microprocesador puede ejecutar hasta 16 millones de instrucciones por segundo. Eso puede parecer mucho, pero si considera que simplemente configurar el pin digital en alto puede tomar más de 50 ciclos de reloj. Las tarjetas basadas en ESP son mucho más rápidas con una velocidad de reloj de 52 MHz a 240 MHz. La placa que se utilizará en la cátedra cuenta con un procesador *Tensilica Xtensa LX6 de doble núcleo de 32 bits de hasta 240 MHz*.

Pines digitales de entrada y salida, PWM y analógicos

Pines de E/S digitales: la diferencia entre todas las placas cuando se trata de pines de E / S digitales es casi nula. La única diferencia es que las placas más grandes, como la NodeMCU ESP32 (36) y la placa más grande, la Arduino MEGA R3 (54) tiene muchos pines de E/S digitales.

Las entradas digitales o binarias de los microcontroladores tienen dos estados: **On / Off (true o false; HIGH o LOW)**. Un valor de tension alto (5v o 3.3v) equivale a un “1” lógico, mientras que uno bajo equivale a un “0”. Se pueden conectar:

- pulsadores
- sensores discretos
- llaves / interruptores



Así como las entradas digitales permiten sensar actividades que tienen dos estados, las salidas digitales permiten controlar actividades que tienen dos estados. Con las salidas digitales se puede encender o apagar el dispositivo que se conecte al pin.

Se pueden conectar:

- Leds
- Pantallas LCD
- Transistores
- Relés

Pines PWM: las placas basadas en ESP tienen una mejor relación de pines de E/S digitales a pines PWM es utilizado por un pin digital. Todos los pines que pueden utilizarse como salida soportan PWM excepto los pines del 34 al 39.

PWM

PWM son las siglas de **P**ulse **W**idth **M**odulation (Modulación por ancho de pulso). Para transmitir una señal, ya sea analógica o digital, se debe modular para que sea transmitida sin perder potencia o sufrir distorsión por interferencias.

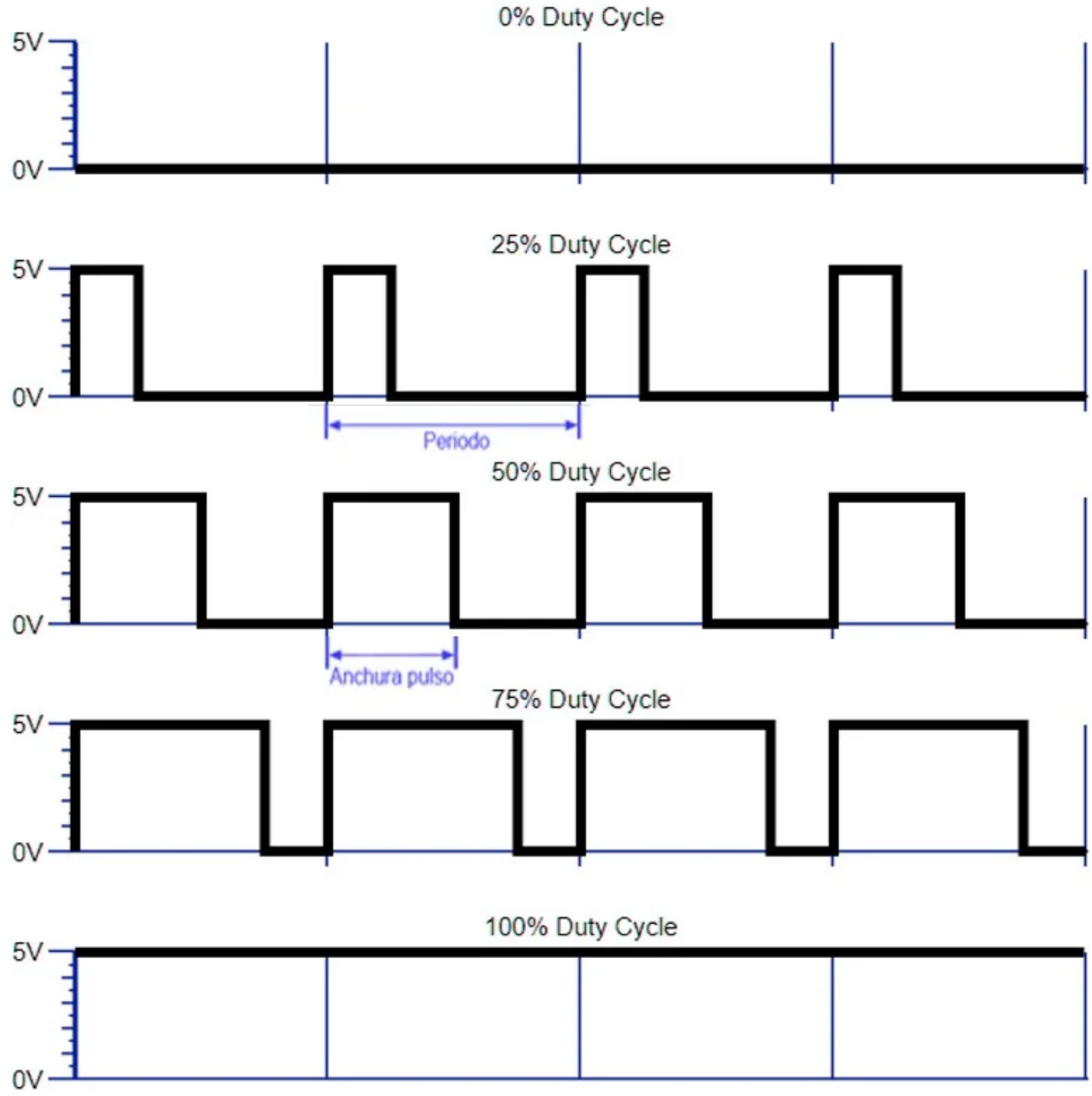
PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. En esta técnica se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. El ciclo de trabajo (duty cycle) de una señal periódica es el ancho de su parte positiva, en relación con el período. Está expresado en porcentaje, por tanto, un duty cycle de 10% indica que está 10 de 100 a nivel alto.

> Duty cycle = t / T
>
> t = tiempo en parte positiva
>
> T = Periodo, tiempo total

Básicamente, consiste en activar una salida digital durante un tiempo y mantenerla apagada durante el resto, generando así pulsos positivos que se repiten de manera constante. Por tanto, la frecuencia es constante (es

decir, el tiempo entre disparo de pulsos), mientras que se hace variar la anchura del pulso, el duty cycle. El promedio de esta tensión de salida, a lo largo del tiempo, será igual al valor analógico deseado.

La siguiente imagen muestra diferentes ciclos de trabajos para un rango de voltajes de 0 a 5v.



Por ejemplo, si con una tensión Vcc de 5V queremos una señal PWM de 1V, se generará una señal que el 20% del tiempo valdrá 5V y el 80% restante 0V.

Esta modulación es muy usada para controlar la cantidad de energía que se envía a una carga, es una técnica utilizada para regular la velocidad de giro de los motores, regulación de intensidad luminosa, controles de elementos termoeléctricos o controlar fuentes conmutadas entre otros usos.

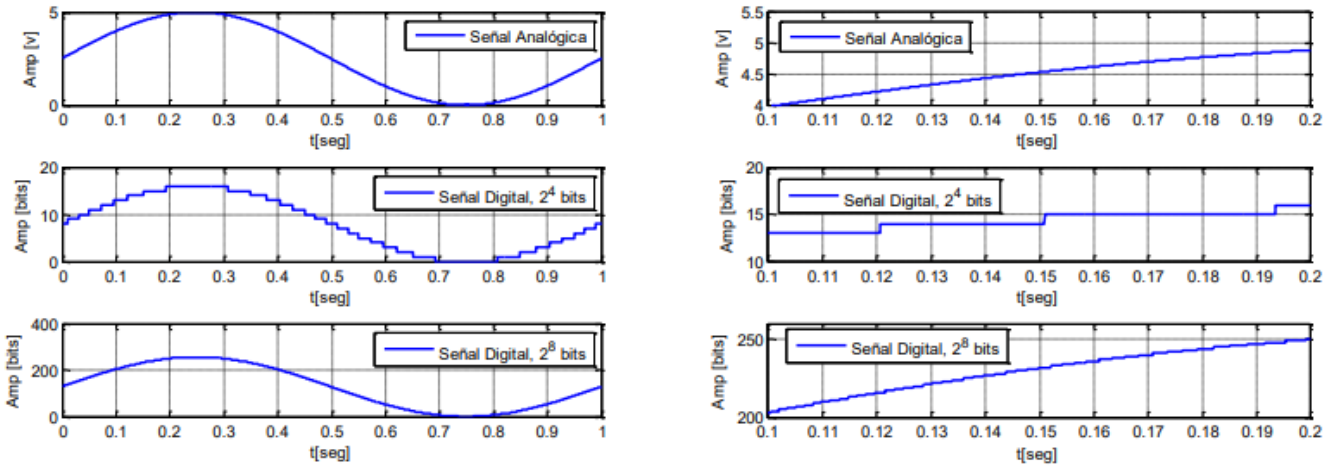
La mayoría de los automatismos, incluido Arduino, no son capaces de proporcionar una señal analógica. Sólo pueden proporcionar una salida digital de -Vcc o Vcc. (por ejemplo, 0V y 5V). Entonces, para conseguir una señal analógica, la mayoría de los automatismos usan PWM. Se usa esta técnica porque como se ve en los ejemplos anteriores, no siempre quieres un valor digital de la señal (ON/OFF), si no que necesitaremos proporcionar un valor analógico de tensión que usarán para las aplicaciones deseadas.

Para comprender más sobre el uso de esta técnica se deja el siguiente video.

Pines analógicos: Una entrada analógica es una que le permite al microcontrolador leer un voltaje variable, típicamente desde los 0V hasta el voltaje de alimentación del microcontrolador. Un puerto de entrada analógica convierte un valor de tension en un número entero.

La placa cuenta con 16 pines que soportan entradas analógicas que son mapeadas a valores de tensión de 0 - 3.3v mediante un conversor analógico-digital (ADC) con una resolución de 12 bits (es decir con valores de 0 a 2075).

Puertos de entrada analógica



Por ejemplo si queremos leer un valor de tensión variable desde un pin 34 utilizamos:

```
int valor = analogRead(34);
```

Podemos configurar la cantidad de bits utilizados mediante...

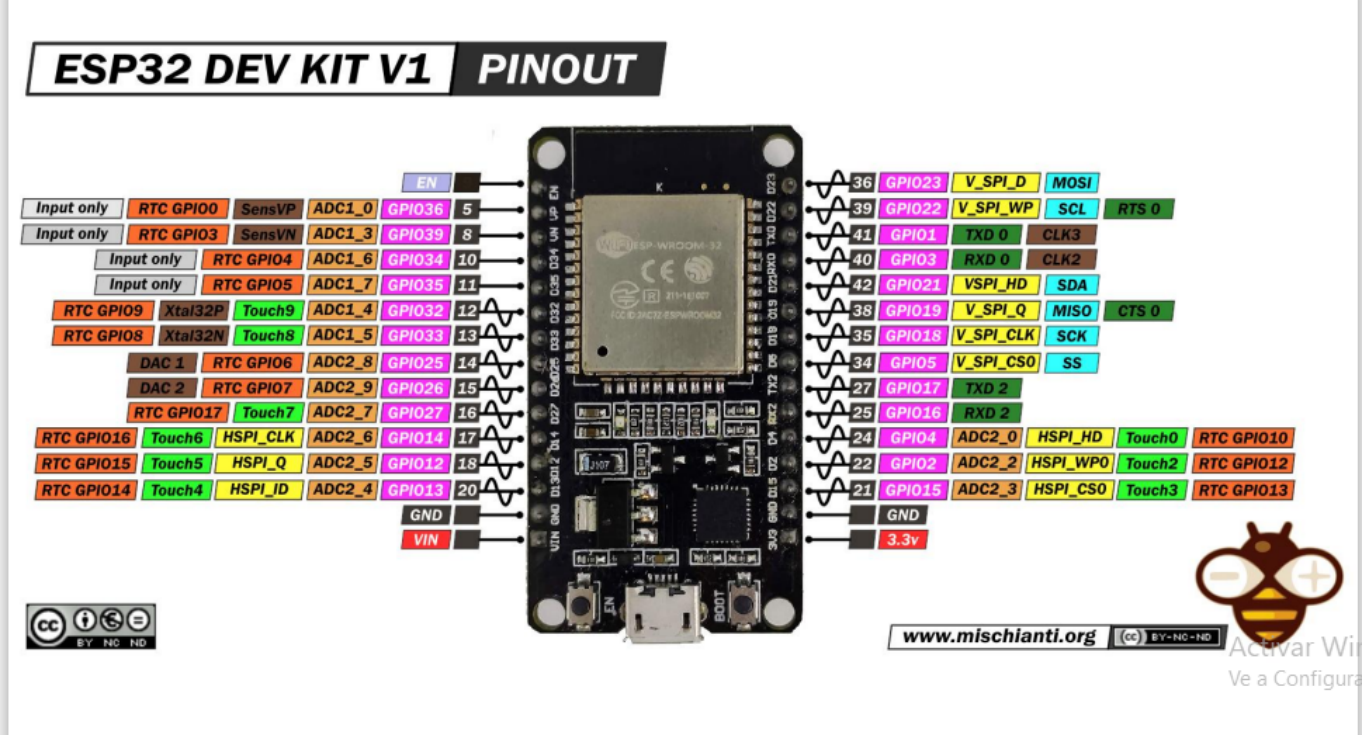
```
analogReadResolution(10);
```

Finalmente para leer un valor variable de voltaje:

```
float voltaje = (analogRead(34)/256)*3.3
```

En el camino inverso, la placa cuenta solo con dos pines que pueden utilizarse como salidas analógicas: GPIO25 y GPIO26 mediante un Conversor Digital Analógico (DAC) con una resolución de 8 bits (con valores de 0 a 255).

La siguiente imagen sintetiza la distribución de conectores de la placa.



- Conector micro-USB para la alimentación y comunicación
- 30 ó 36 Conectores GPIO para entrada y salida de señales digitales
- 3 interfaces serie SPI (HW específico)
- 2 interfaces I2C (HW específico)
- 2 interfaces I2S
- 2 canales de conversor analógico a digital de 12 bits (ADC)
- 2 canales de conversor digital a analógico (DAC)
- 16 canales de salida PWM
- 3 interfaces UART
- 10 GPIO de detección capacitiva (Touch)
- Botones ENABLE y BOOT
- Conectividad Wi-Fi y Bluetooth
- LED integrado

Protocolos de comunicación

- **SPI** (Serial Peripheral Interface) para enviar datos entre microcontroladores. Es un bus de datos síncrono, es decir, utiliza un reloj para regular la transferencia de datos.
- **I2C** más utilizado para enviar y recibir datos de otros dispositivos como pantallas OLED, sensores de presión barométrica, etc.
- **I2S** (Inter-IC Sound), es un estándar de interfaz de bus serie eléctrico que se utiliza para conectar dispositivos de audio digital entre sí.
- **UART** (Receptor / Transmisor Asíncrono Universal) no es un protocolo de comunicación como SPI e I2C, sino un circuito físico en un microcontrolador. El objetivo principal es transmitir y recibir datos en serie.

Comunicaciones inalámbricas

El módulo ESP32 es una solución de Wi-Fi/Bluetooth todo en uno, integrada y certificada en un solo chip. Además, el ESP32 dispone de conexión **BLE**. BLE o **Bluetooth Low Energy** es adecuada para conectar dos

dispositivos cercanos y para transmitir poco volumen de datos, de manera no continua. Para transmitir de manera continua es más adecuado el bluetooth tradicional.

Entorno de trabajo para Esp32

Para comezar a trabajar con ESP32 es necesario instalar:

- Arduino IDE como herramienta de desarrollo.
- Driver de la placa instalado en Arduino IDE.

Para obtener instrucciones paso a paso de cómo instalar el Entorno de Desarrollo (IDE) de Arduino se sugiere visitar el enlace: https://www.arduino.cc/en/Guide, donde encontrará una guía con los pasos necesarios para instalar este software, así como también documentación de referencia y ejemplos. Actualmente se encuentra en su versión 2.X que a diferencia de la versión anterior se incorporar características muy requeridas de desarrollo como el autocompletado, la integración con los repositorios de código GIT y el debbuger, entre otros. Para mayor información visitar este video: https://www.youtube.com/watch?v=7gymFdvJQ2s.

Otra alternativa es trabajar con el Editor Visual Studio Code e instalar las extensión particulares para hacer desarrollos sobre Arduino. https://www.luisllamas.es/arduino-visual-studio-code/

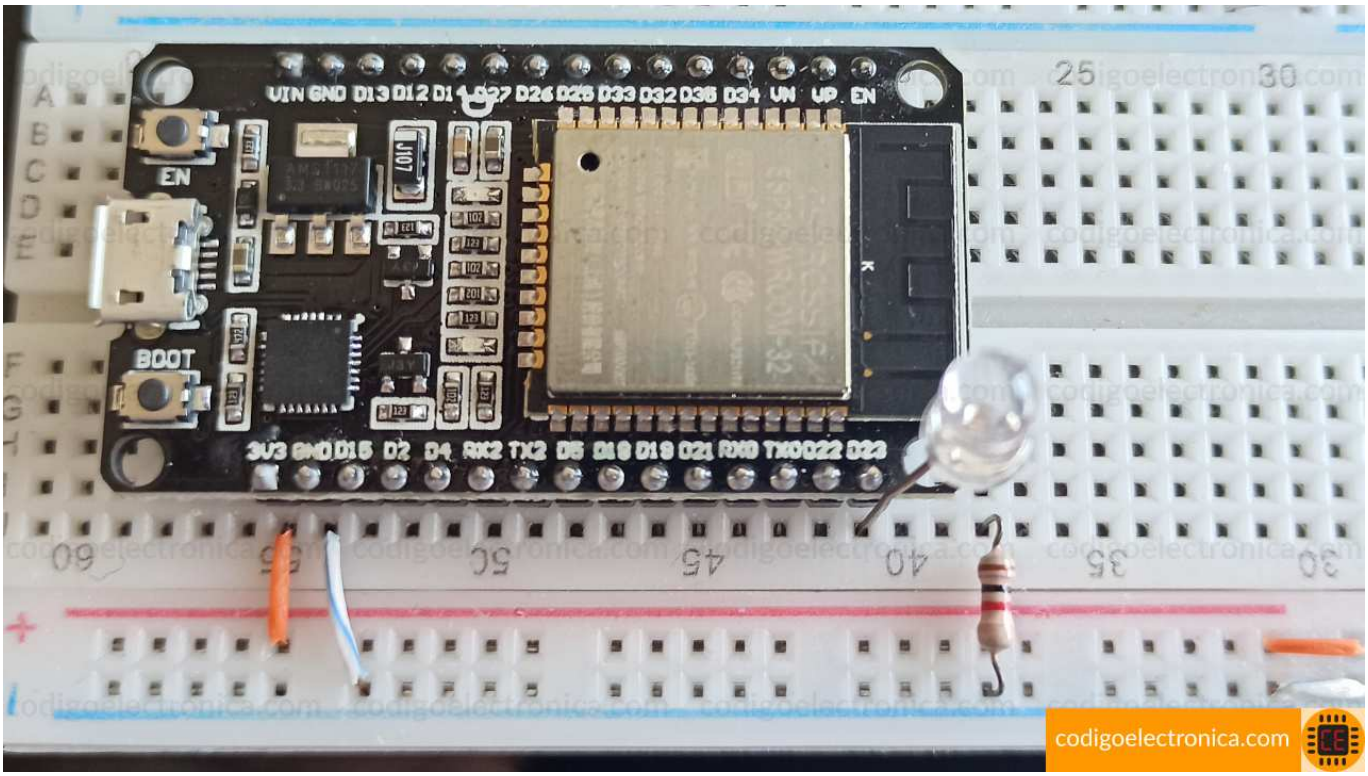
Una vez instalado **Arduino IDE** es necesario instalar el controlador de la placa. Para ello se deja el siguiente enlace con los pasos necesarios: https://www.electrosoftcloud.com/esp32-configuracion-y-primeros-pasos/. Es también de gran ayuda ver el siguiente video que explica cómo averiguar qué driver es necesario instalar para Win10 según el modelo de la Esp32: https://www.youtube.com/watch?v=JmDxP4O4Trk. En la descripción del video están los enlaces a los drives mencionados.

Primer ejemplo: Blink

> Para comenzar con un primer ejemplo de aplicación se va trabajar con el clásico ejemplo de encender y apagar un led usando el esp32 como controlador.

Para comenzar en necesario disponer de los siguientes materiales:

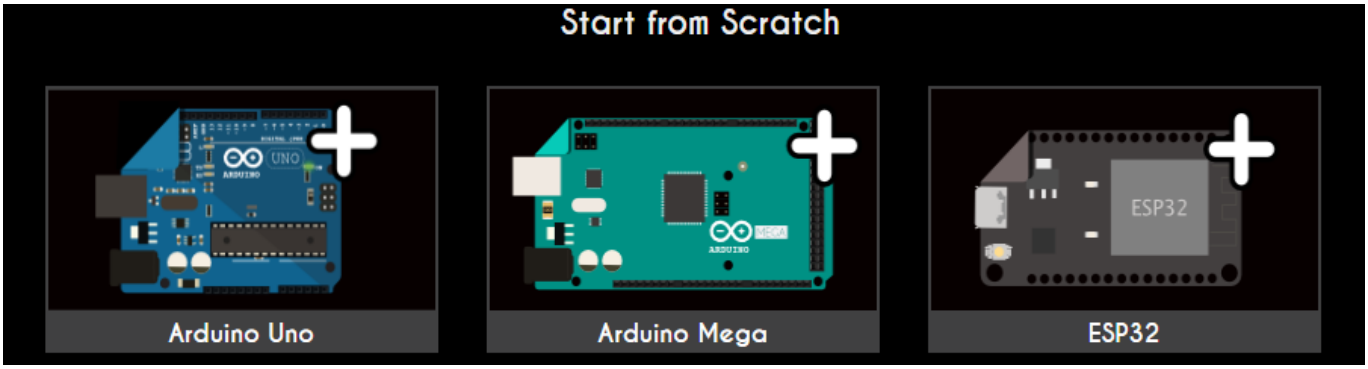
- Placa Esp32
- Led
- Resistencia de 1k (opcional ya que todos los pines de la placa tiene una resitencia Pull-Down y Pull-Up integradas)



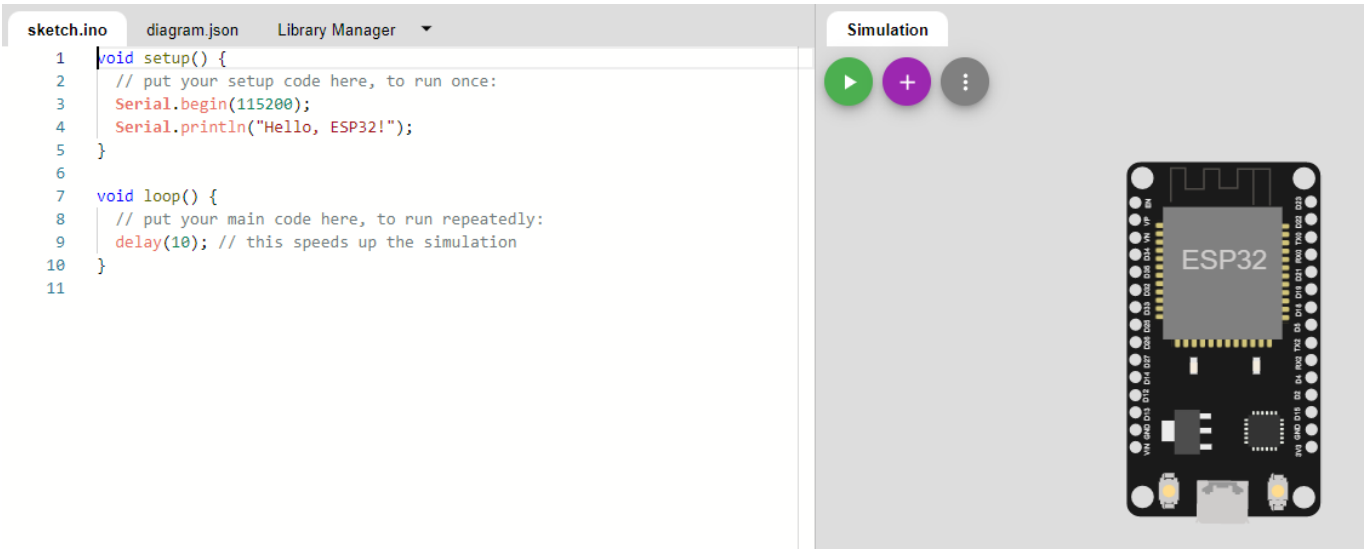
Diseñar lógicamente el circuito con WokWi

Wokwi es un simulador de electrónica en línea que permite simular desarrollos sobre Arduino, ESP32 y muchas otras placas, piezas y sensores populares. Para comenzar a utilizarlo solo es necesario crear una cuenta de usuario clásica a partir de un correo electrónico personal.

Una vez creado el usuario, en el home del sitio se selecciona la placa Esp32 desde la opción *start from Scratch*:



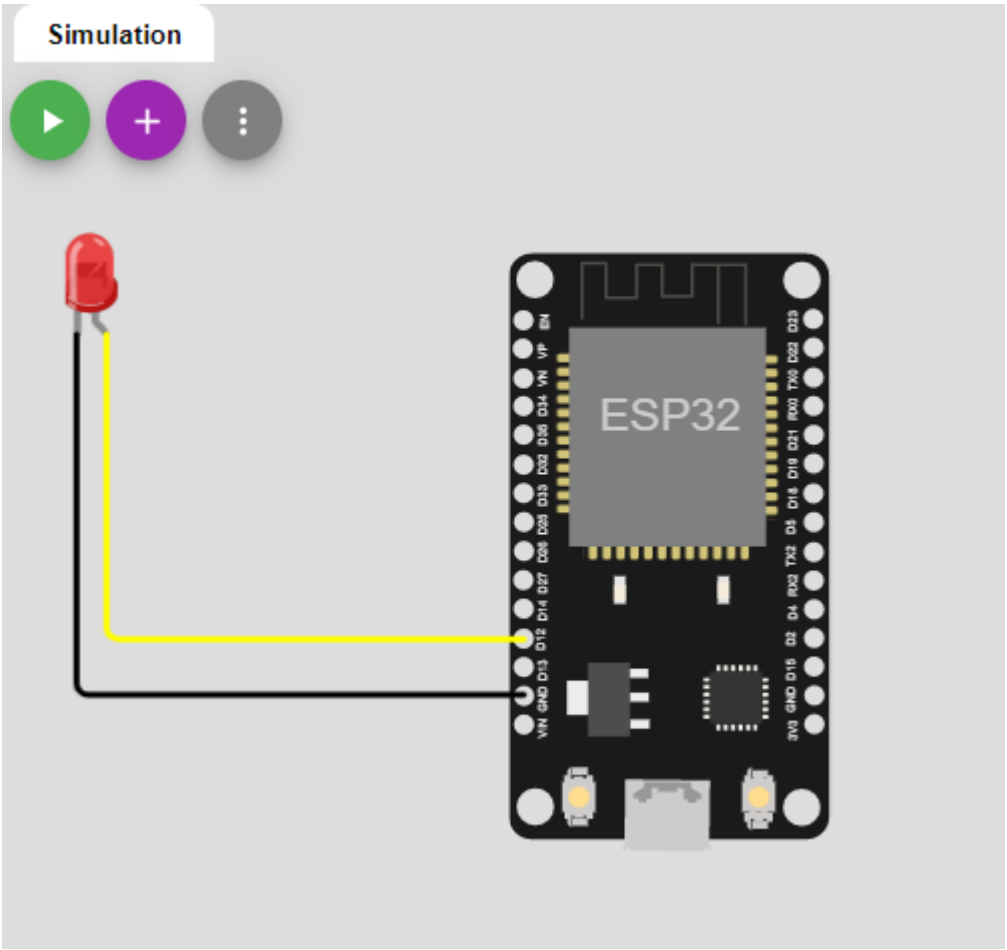
Una vez seleccionada la placa se muestra:



En la parte izquierda se habilita un editor para escribir las líneas de código C/C++, al igual que en Arduino IDE. En la parte derecha se muestra gráficamente la placa y es posible incorporar mediante la opción (+) los dispositivos electronicos como leds, pulsadores, potenciometros, pantallas, etc. Luego se conectan los pines de la placa a los dispositivos agregados y la herramienta actualiza la información de conexionados mediante un archivo JSON que puede gestionarse desde la solapa *diagram.json*. Es posible definir identificadores, colores y posiciones de los elementos dibujados.

De ser necesario, la pestaña *Library Manager* permite cargar librerías especificias de desarrollo según los componentes agregados al desarrollo.

Retornando al ejemplo del **Blink**, el diagrama Wokwi resulta:



Donde el conector cátodo (-) del led se conecta a la señal **GND** del Esp32, mientras el pin **D12** se conecta con el ánodo o conector positivo (+).

Finalizado el conexionado de la placa, se escribe el código que será instalado y ejecutado. Para compilar, instalar y ejecutar el sketch en la Esp32 se selecciona la opción **Play (>)**.

Desde la opción **Guardar** es posible descargar tanto el código en formato .ino como el archivo de diagrama en formato .json con la configuración de los dispostivos y la placa.

Para ver el código de la solución acceder a esta carpeta: Práctico 01