



UNIVERSIDAD
POLITECNICA
DE VALENCIA

UNIDAD 5

ANDROID THINGS: LAS COSAS

IoT Y VISIÓN ARTIFICIAL

Master Aplicaciones Android

Jesús Tomás

jtomas@upv.es

Gandia 16,23 y 30 de mayo de 2018

TEMAS DEL CURSO

Tema 1

Análisis Imagen en Android usando OpenCV

Tema 2

Programación en código nativo

Temas 3

Android Things

Temas 4

Ingeniería inversa en Android



- Profesor: Salvador Santonja y Jesús Tomás
- Del 16 de mayo al 13 de junio
- Temario:

U5: **Las cosas**

- Visión general
- Instalación en Raspberry Pi 3
- Tipos de entradas / salidas (GPIO, PWM, I2C, UART,...)

U6: **La Comunicación**

- Offline: Bluetooth, Nearby Connections
- Online: servicios Web, MQTT
- Servicios IoT: Firebase, Google Cloud IoT



MATERIAL PARA EL CURSO

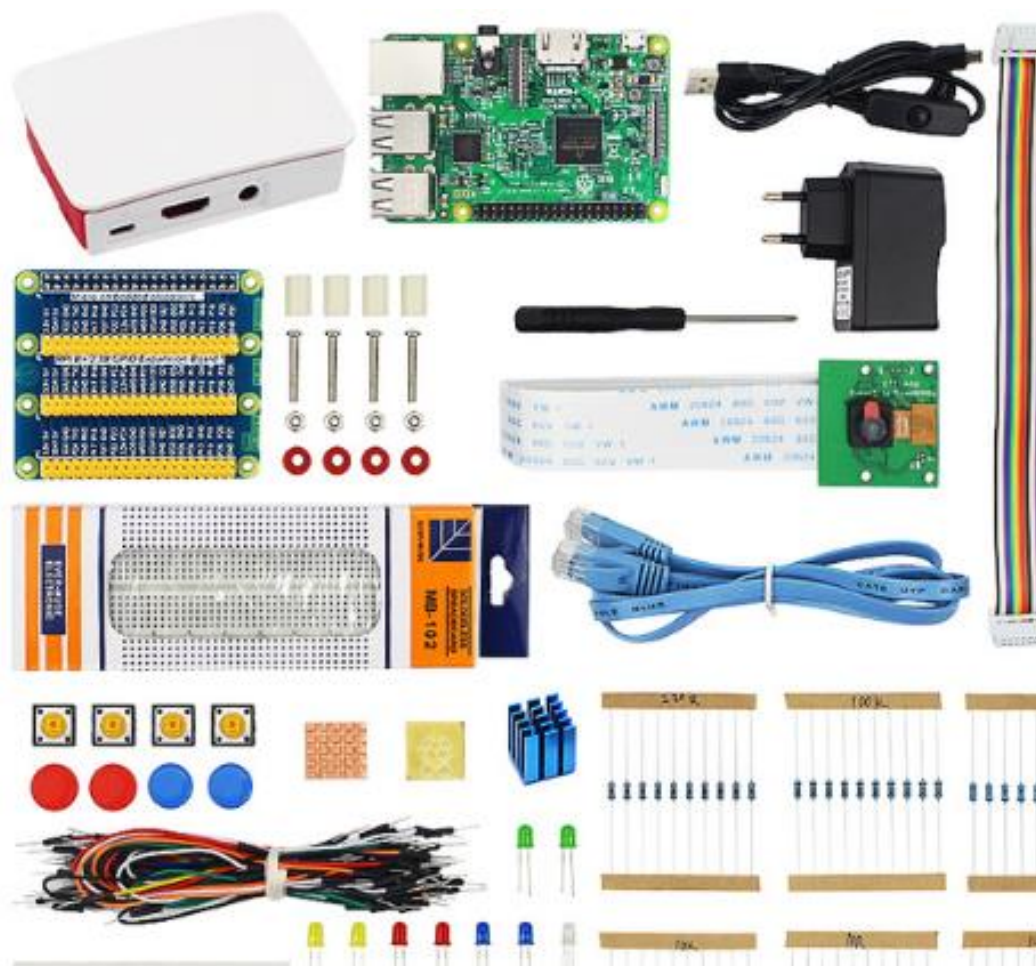
- **Raspberry Pi 3B** (NO 3B+) (RS 35€ (5€ envío) AliExpress 32,7 PC Componentes 35€ +envío Media Markt 36,6 +2€ envío)
- **Micro SD** (8G mínimo) (PC Componentes 5€)
- **Alimentador micro USB** (2,5A), HDMI, Ethernet
- **Placas de prototipos** (1,92)
- **Cables, resistencias, LED, pulsador** (8,02)(7,2)(13,35)(9,35)
- **Cámara:** Oficial (16), China (5,5), USB (3,84)
- **Conversor A/D compatible I2c** (2,48)
- **Kit sensores actuadores** Kit 16 sensores (9,22)
- **Entrada salida audio por USB** (3,08)
- **Arduino Nano** (3,22)



MATERIAL PARA EL CURSO

Opción completa:

- Raspberry Pi+caja+SD+cámara+placa+alimentador+cables... (58,16)



MATERIAL PARA EL CURSO

Por partes (si no se compra el ítem anterior):

- Raspberry Pi 3B (o 3B+) (RS 35€ (5€ envío) AliExpress 32,7 PC Componentes 35€ +envío Media Markt 36,6 +2€ envío)
- Micro SD (8G mínimo) (5€) o ver opciones DealExtrem
- Placa, cables, resistencias, LED, pulsador (8,02) (7,2) (13,35) (9,35)
- Cámara: China (5,5)
- (opcional) Raspberry pi breakout board (+ placa pequeña (4,3), + placa grande +cables+resistencias (6,3))



MATERIAL PARA EL CURSO

Otros componentes:

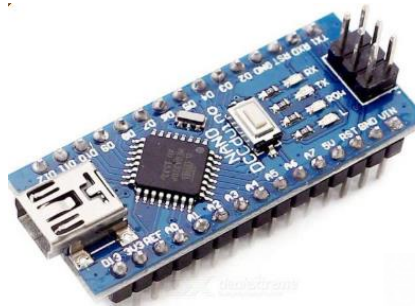
- Kit 16 sensores ([9,22](#))



- Conversor A/D compatible I2c, con 3 sensores ([2,48](#))



- Arduino Nano ([3,22](#))

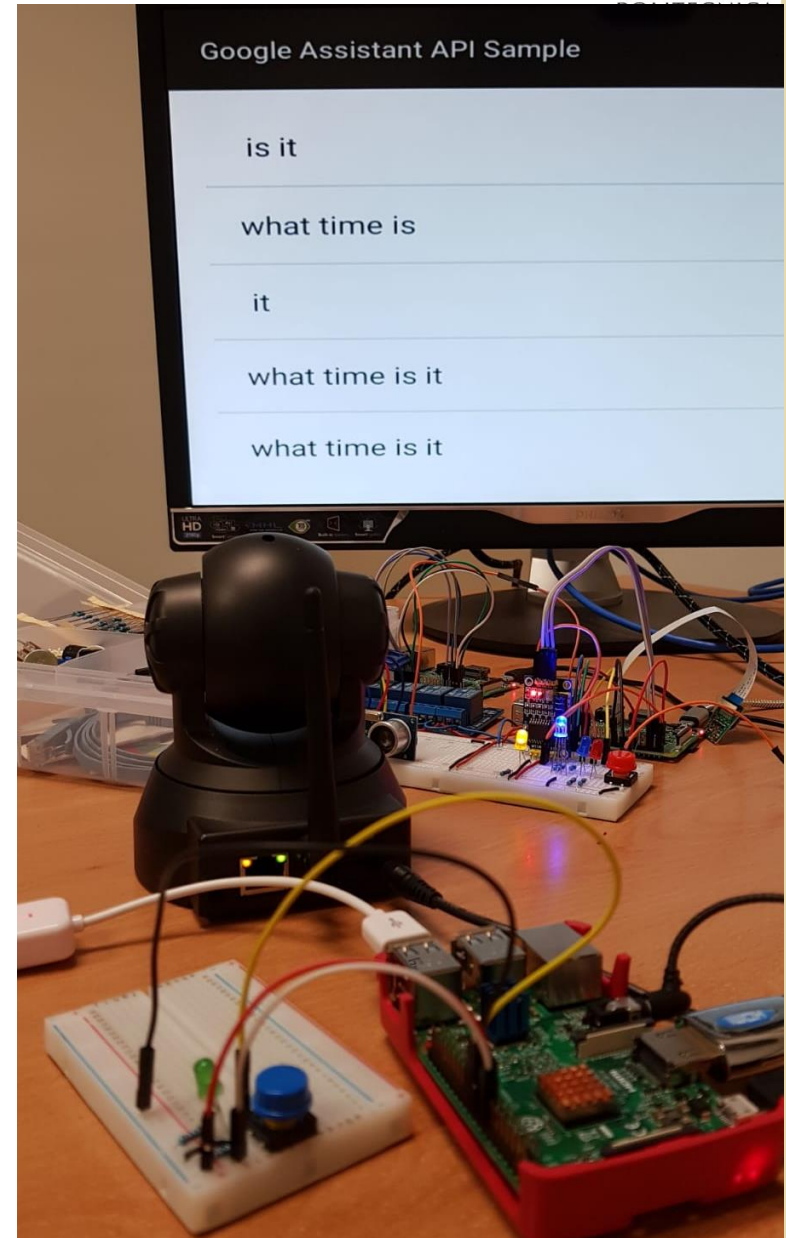


- Entrada salida audio por USB ([3,08](#))



LABORATORIO REMOTO

- Si alguien no dispone de hardware, puede realizar las prácticas de forma remota.
- <http://www.androidcurso.com/index.php/75-slider/882-laboratorio-virtual-para-android-things>



U.5 - ANDROID THINGS: LAS COSAS

- 1 - Internet de las cosas
- 2 - Android Things
 - Solución completamente administrada
 - Modelo de actualizaciones
 - Plataformas hardware soportadas
 - SDK
- 3 - Raspberry Pi
- 4 - Instalación de Android Things
- 5 - Algunos conceptos de electrónica
- 6 - Entradas / Salidas en Android Things
- 7 - Usar un microcontrolador Arduino como esclavo
- 8 - Escribir controladores
- 9 - Integrar Google Assistant SDK



U.5 - ANDROID THINGS: LAS COSAS

- 1 - Internet de las cosas
- 2 - Android Things
- 3 - Raspberry Pi
- 4 - Instalación de Android Things
- 5 - Algunos conceptos de electrónica
- 6 - Entradas / Salidas en Android Things
 - GPIO, PWM, I2C, UART, SPI
 - Conversor A/D y D/A por medio de I2C
 - Uso del medidor ultrasónico de distancia
- 7 - Usar un microcontrolador Arduino como esclavo
- 8 - Escribir controladores
- 9 - Integrar Google Assistant SDK



1 - INTERNET DE LAS COSAS

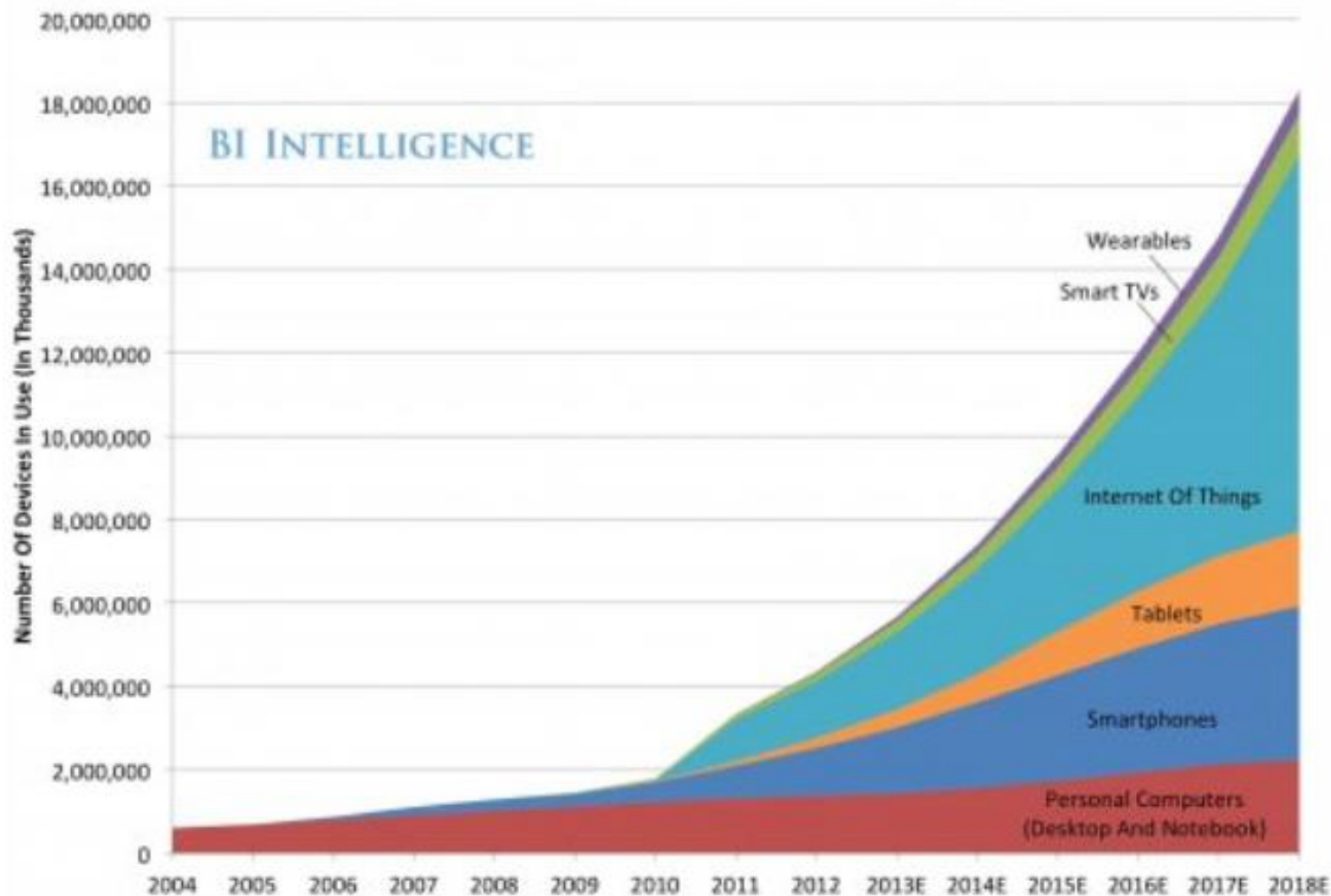


¿QUÉ ES INTERNET DE LAS COSAS?

- Puede ser la tecnología más activa en la próxima década.
- Se prevé un impacto en nuestras vidas, similar al que han tenido los dispositivos móviles.
- La primera definición en 1999 por Kevin Ashton, “El IoT es el mundo en el que cada objeto tiene una identidad virtual propia y capacidad potencial para integrarse e interactuar de manera independiente en la red con cualquier otro individuo, ya sea una máquina o un humano.”
- IoT incluye todos los objetos que pueden conectarse a Internet e intercambiar información.



Global Internet Device Installed Base Forecast



ALGUNOS EJEMPLOS

Amazon Dash Button



Amazon Key

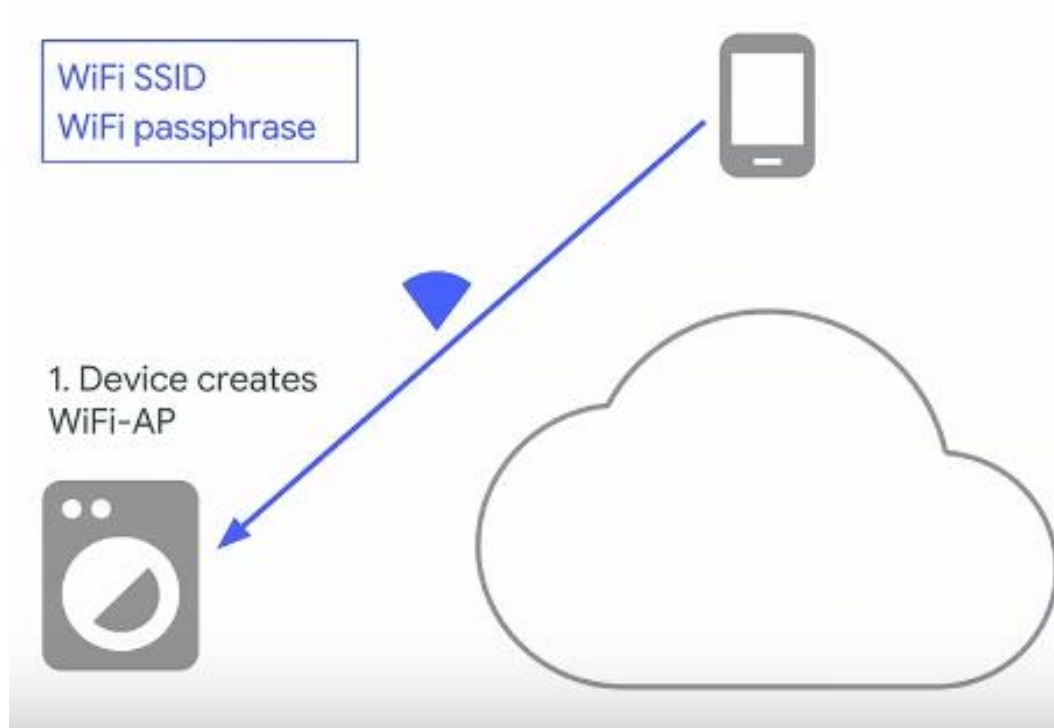


Electrodomésticos, wearables, ciudad inteligente,
automóvil, industria ...



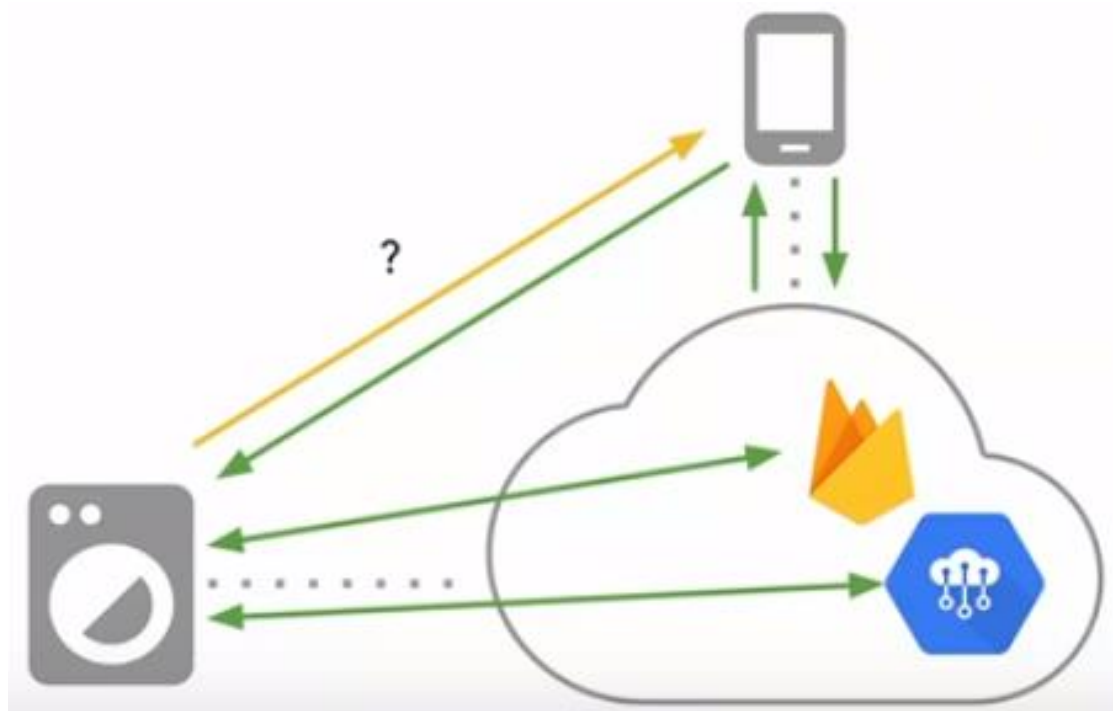
UN DISPOSITIVO DE IoT TÍPICO

- El dispositivo ha de conectarse a Internet



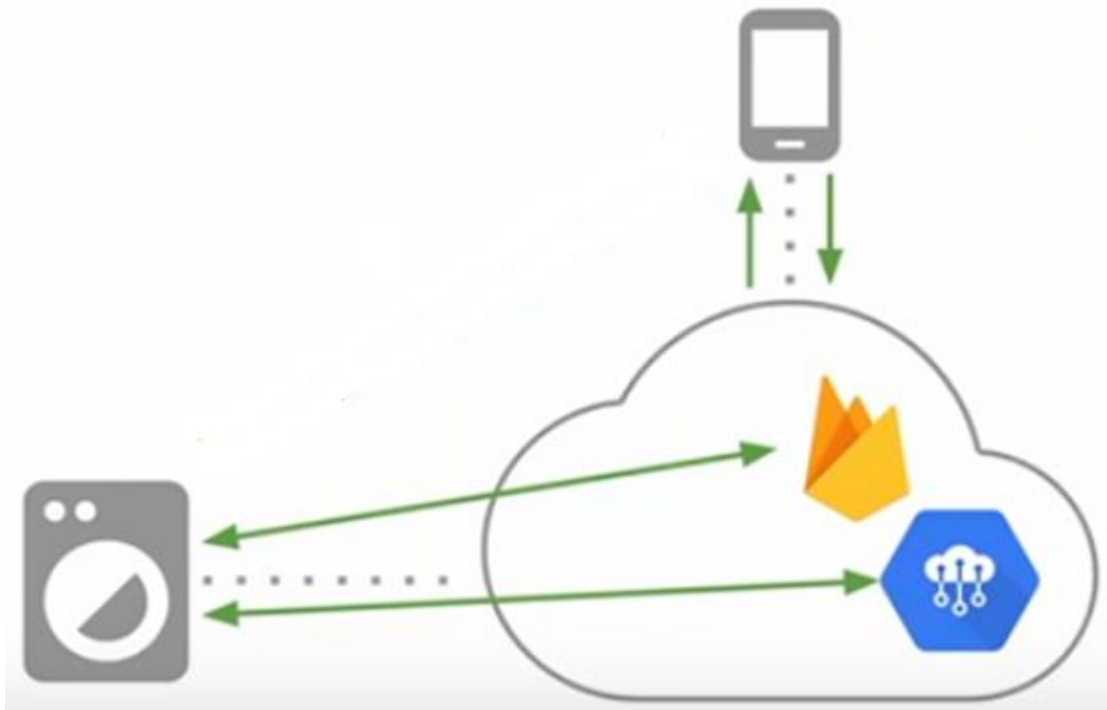
UN DISPOSITIVO DE IoT TÍPICO

- El dispositivo ha de autenticar los usuarios que tienen acceso:



UN DISPOSITIVO DE IoT TÍPICO

- El dispositivo ha de poder enviar y recibir datos:



UN DISPOSITIVO DE IoT TÍPICO

- Seguridad:



2 – ANDROID THINGS



UNIVERSIDAD
POLITECNICA
DE VALENCIA



¿QUÉ ES ANDROID THINGS?

- Plataforma de Google para el desarrollo en dispositivos incrustados, que trabajan en entornos de Internet de las Cosas.
- Historia:
 - Proyecto Brillo: Anunciado en Google I/O 2015
 - Primeros desarrollos diferencias con Android
 - No se programa en Java
 - En Google I/O 2018: versión 1.0
 - Android adaptado a microprocesadores
- A destacar:
 - Aprovechamos todo las herramientas disponibles en Android
 - Solución completamente administrada
 - Rapidez en poner en marcha un producto comercial



DIFICULTAD DESARROLLO Y COMERCIALIZACIÓN

- Desarrollar un producto IoT es muy complejo:
 - Involucra: electrónica, microprocesadores, comunicaciones y software.
- Dos problemas especialmente complejos:
 - Actualizaciones de software:
 - Errores en el desarrollo, agujeros de seguridad, mejoras de prestaciones, cambios de política de servicios o test A/B de usabilidad.
 - Seguridad:
 - Información muy sensible a los usuarios (cámaras, hábitos, ...).
 - Comunicación con la nube ha de ser 100% segura.
 - Hay que impedir software malicioso en el dispositivo.
Garantizando que el software no ha sido reemplazado. Incluso con acceso al dispositivo.
- Ciclo de desarrollo elevado y mantenimiento costoso



SOLUCIÓN COMPLETAMENTE ADMINISTRADA

- Google nos ofrece una solución con la que podríamos disponer de un prototipo en cuestión de semanas y llevarlo a producción en meses.
- Muchas partes ya resueltas:
 - no resulta imprescindible tener conocimientos sobre sistemas integrados ni seguridad.
- Está basada en una arquitectura de cuatro niveles:



SOLUCIÓN COMPLETAMENTE ADMINISTRADA



UNIVERSIDAD
POLITECNICA
DE VALENCIA



ACTUALIZACIONES EN LÍNEA (OTA)

- Juega un papel fundamental en la seguridad
- Podremos realizarlas desde la consola
- Pueden incluir nuestra aplicación o la imagen del sistema
- Google se compromete a hacerse cargo de las actualizaciones de forma gratuita durante los primeros tres años de soporte, realizándose de forma mensual.
- Las actualizaciones automáticas están habilitadas de manera predeterminada
- Después de tres años habrá "opciones para soporte extendido".



MODELO DE ACTUALIZACIONES TRADICIONAL

- En el modelo de actualizaciones tradicional de Android, las actualizaciones eran responsabilidad del fabricante.
- Ha de ser así, al modificar a su gusto el SO.
- A ningún fabricante le gusta un modelo de hardware interoperable (ej. PC). Prefieren diferenciarse personalizando el software.
- Problema, muchos fabricantes no realizaban actualizaciones o dejaban de hacerlo en modelos antiguos.
- Es más rentable que el usuario compre otro teléfono.



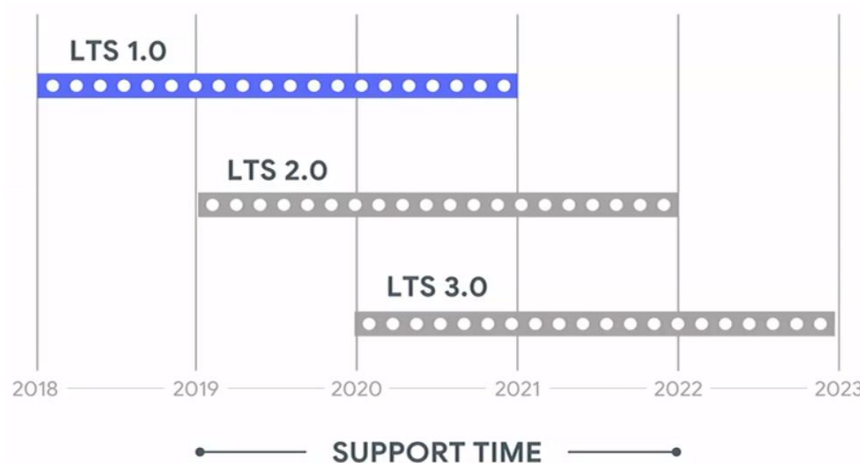
CAMBIO DE MODELO DE ACTUALIZACIONES

- Google quiere cambiar el modelo tradicional.
 - Puede hacerlo aprovechando su preponderancia en el mercado.
- Proyecto Treble (comenzó en Android O)
- La idea es separar el sistema operativo Android de los controladores y firmware específicos.
- De esta forma Google lanza actualizaciones de seguridad mensuales de Android directamente a los teléfonos, saltándose a los fabricantes.
- Google quiere utilizar este planteamiento en Android Things.
- Al ser un software abierto, también podemos ir por nuestra cuenta.



TEMPORIZACIÓN DE ACTUALIZACIONES

- Google garantiza las actualizaciones por 3 años.
- Podremos aprovechar esta infraestructura para actualizar nuestra app.
- Solo se actualizará de forma automática en la versión principal. cuando salga v2.0, no estaremos obligados a utilizarla, pero seguiremos recibiendo actualizaciones de v1.X.



PRODUCTOS BASADOS EN ANDROID THINGS

- En el Google I/O 2018 se presentó
- Altavoz inteligente LG WK7: primer producto basado en Android Things



Powered by
Android Things

Intelligent edge devices

Security updates

Rapid scaling to production



PANTALLAS INTELIGENTES

- Añade una pantalla a un altavoz inteligente como el Amazon Echo o Google Home
 - El primero fue [Amazon Echo Show](#),
- Marcas, como LG, Lenovo y JBL, lanzarán en verano sus modelos basados en Android Things.
- Breve periodo en tenerlos comercializables.



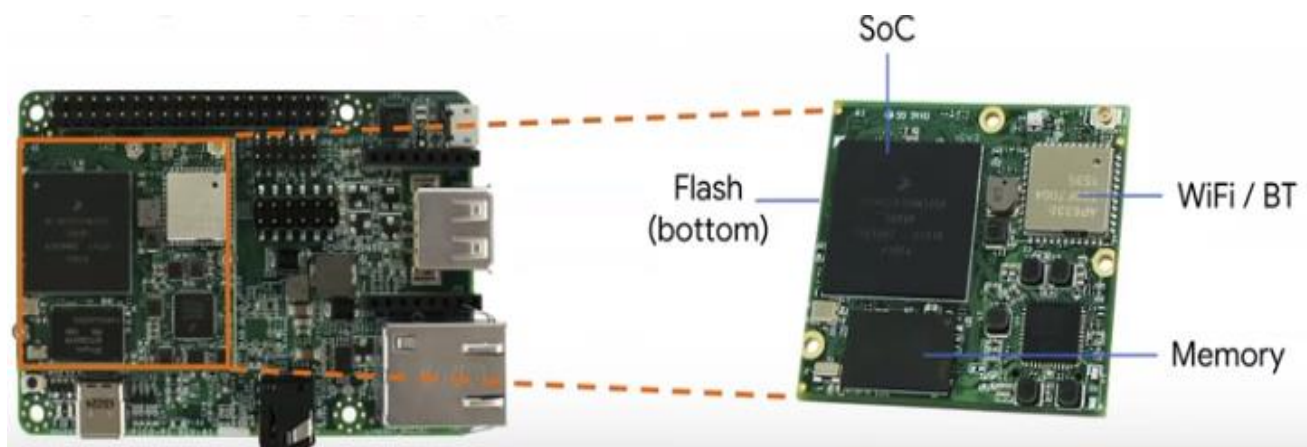
PLATAFORMAS DE HARDWARE SOPORTADAS

- Google trabaja con varios fabricantes para certificar que plataformas.
- Han de cumplir ciertas medidas de seguridad y son las únicas que podrán recibir actualizaciones directamente desde Google.
- Veamos tres conceptos:
 - SoC (System on Chip)
 - SoM (System on Module)
 - SBC (Single Board Computer)
- Los tres hacen referencia a tener un ordenador (CPU, GPU, RAM, almacenamiento, WiFi, E/S) en diferentes niveles de integración.



SoC, SoM Y SBC





- **SoC** (System on Chip): En un solo chip de silicio
- **SoM** (System on Module): Más grandes y más baratos al tener menor nivel de integración.
- **SBC** (Single Board Computer): suelen incorporar un SoC o un SoM y añaden E/S adicionales, conectores, alimentación, etc.



SoM SOPORTADOS



UNIVERSIDAD
POLITECNICA
DE VALENCIA



Platform	NXP i.MX8M  Learn More	Qualcomm SDA212  Learn More	Qualcomm SDA624  Learn More	MediaTek MT8516 
CPU & Memory	<ul style="list-style-type: none"> • NXP i.MX8M • 1.5Ghz quad-core ARM Cortex A53 • 1GB or 2GB RAM 	<ul style="list-style-type: none"> • Qualcomm Snapdragon™ 212 • Quadcore 1.267Ghz ARM Cortex A7 • 1GB RAM 	<ul style="list-style-type: none"> • Qualcomm Snapdragon™ 624 • Octacore 1.8Ghz ARM Cortex A53 • 2GB RAM 	<ul style="list-style-type: none"> • MT 8516 • 1.3Ghz quad-core ARM Cortex A35 • 512MB RAM
GPU	QC7000Lite	QC Adreno 304	QC Adreno 506	N/A
Storage	4GB eMMC	4GB eMMC	4GB eMMC	4GB eMMC
Display	MX8-DSI-OLED1	N/A	8-inch WXGA Innolux Display with Touch	N/A
Camera	OV5640 MIPI CSI	N/A	Omnivision OV5693 5MP sensor	N/A

SoM SOPORTADOS

Platform	NXP i.MX8M	Qualcomm SDA212	Qualcomm SDA624	MediaTek MT8516
Networking	10/100/1000 Ethernet Wi-Fi 802.11ac Bluetooth® 4.2	Wi-Fi 802.11ac (2.4/5.0GHz) Bluetooth® 4.2	Wi-Fi 802.11ac (2.4/5.0GHz) Bluetooth® 4.2	10/100/1000 Ethernet Wi-Fi 802.11ac (2.4/5.0GHz) Bluetooth® 5.0
USB	2x USB 3.0 Type C	2x USB 2.0 Host 1x USB 2.0 OTG	1x USB 3.0 Type C	1x USB 2.0 Host 1x USB 2.0 OTG
Size (width x length)	50.3mm x 50.3mm	50mm x 46.5mm	50mm x 46.5mm	N/A
Type	Physical	Physical	Physical	Virtual





SOM PARA PROTOTIPOS

Platform	NXP Pico i.MX7D	Raspberry Pi 3 Model B
		
CPU & Memory	<ul style="list-style-type: none"> • NXP i.MX7D • 1GHz dual-core ARM Cortex A7 • 512MB RAM 	<ul style="list-style-type: none"> • Broadcom BCM2837 • 1.2GHz quad-core ARM Cortex A53 • 1GB RAM
GPU	N/A	VideoCore IV
Storage	4GB eMMC	MicroSD card slot
Display	LCD8000-43T VL050-80128NM-C01	HDMI Raspberry Pi Touch Display
Camera	OV5640 MIPI CSI	RPi Camera module v2

- No cumplen con los requisitos de seguridad para la certificación:
 - clave de identificación
 - arranque verificado
- No reciben actualizaciones de seguridad
- Solo para prototipos



SOM PARA PROTOTIPOS

Platform	NXP Pico i.MX7D	Raspberry Pi 3 Model B
		
Networking	10/100/1000 Ethernet Wi-Fi 802.11ac (2.4/5.0GHz) Bluetooth® 4.1	10/100 Ethernet Wi-Fi 802.11n (2.4GHz) Bluetooth® 4.1
USB	1x USB 2.0 Host 1x USB 2.0 OTG	4x USB 2.0 Host
Size (width x length)	37mm x 40mm	85mm x 56mm (complete board)
Type	Physical	Physical



TIPOS DE SoM

- **SoM físico:** placa que puede extraerse físicamente.
- **SoM virtual:** diseño de referencia del fabricante y certificado por Google. Luego podemos integrar los componentes individuales de ese diseño directamente en un producto.
 - Para productos de gran volumen. Se necesita flexibilidad o componentes ajustados a dimensiones del producto.
- Para cada SoM Google nos proporciona un Board Support Package (BSP). no es necesario interactuar con los fabricantes.
- Migración sencilla.



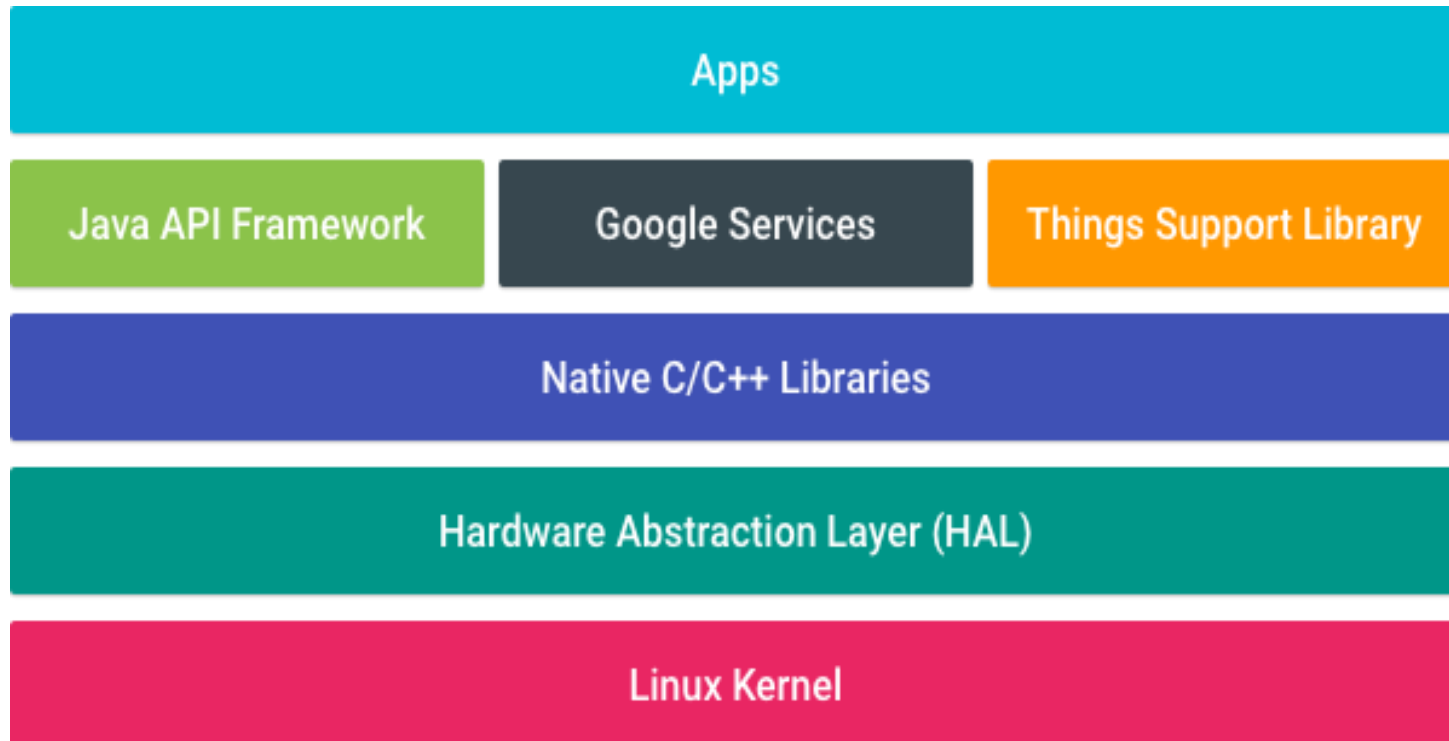
SDK DE ANDROID THINGS

- SDK de Android Things muy similar al de Android
 - Se añaden algunas librerías y se quitan otras
- Principales diferencias:
 - Acceso más flexible a E/S, periféricos y controladores.
 - Aplicaciones de sistema no presentes.
 - Los dispositivos solo ejecutan una aplicación, que se arranca al inicio. Un usuario no puede arrancar aplicaciones.



ARQUITECTURA

- Igual que en Android



THINGS SUPPORT LIBRARY

- **Bluetooth:** emparejamiento y conexión con dispositivos cercanos.
- **Device Updates:** actualizaciones de software basadas en OTA (Over The Air).
- **LoWPAN:** Acceso a redes de área personal inalámbricas de baja potencia.
- **NDK:** El Kit de Desarrollo Nativo se integra de forma predeterminada. Para desarrollar en C / C++.
- **Peripheral I/O:** Comunicación con sensores y actuadores utilizando interfaces estándar: GPIO, PWM, I2C, SPI y UART.
- **User Driver:** Los controladores de usuario nos permiten registrar nuevos drivers de dispositivo desde la aplicación. Podemos inyectar eventos de hardware al sistema, que otras apps podrán capturar.
- **Settings:** Para configurar pantalla, hora del sistema y configuraciones regionales.



CARACTERÍSTICAS NO SOPORTADAS

Característica	API
Interfaz de usuario (barra estado, navegación, quick settings)	<u>NotificationManager</u> <u>KeyguardManager</u> <u>WallpaperManager</u>
VoiceInteractionService	<u>SpeechRecognizer</u>
android.hardware.fingerprint	<u>FingerprintManager</u>
android.hardware.nfc	<u>NfcManager</u>
android.hardware.telephony	<u>SmsManager</u> <u>TelephonyManager</u>
android.hardware.usb.accessory	<u>UsbAccessory</u>
android.hardware.wifi.aware	<u>WifiAwareManager</u>
android.software.app_widgets	<u>AppWidgetManager</u>
android.software.autofill	<u>AutofillManager</u>
android.software.backup	<u>BackupManager</u>
android.software.companion_device_setup	<u>CompanionDeviceManager</u>
android.software.picture_in_picture	<u>Activity Picture-in-picture</u>
android.software.print	<u>PrintManager</u>
android.software.sip	<u>SipManager</u>



CAMBIOS DE COMPORTAMIENTO (I)

○ **Aplicaciones comunes no disponibles:**

- No se incluyen Contactos, Settings, Calendario, ...
- No podremos usar las intenciones comunes que utilizaban estas aplicaciones o proveedores de contenido estándar.

○ **Salida por pantalla opcional:**

- Algunos dispositivos disponen de salida HDMI.
- Podremos diseñar la salida por pantalla de igual forma.
- La aplicación ocupa toda la pantalla.
- No incluye barra de estado, ni botones de navegación.
- Muchos dispositivos no necesitan pantalla.
- Aunque sea así, la actividad siguen siendo el componente principal de la aplicación. Se sigue enviando los eventos de entrada a la actividad en primer plano.



CAMBIOS DE COMPORTAMIENTO (II)

○ Aplicación de arranque (home):

- No hay una aplicación de arranque (home), desde donde el usuario lanzar otras aplicaciones.
- Se espera que una de las aplicaciones instaladas asuma este rol y se ejecute tras el arranque del sistema.
- Filtro de intención **CATEGORY_DEFAULT** y **IOT_LAUNCHER**.

○ Declaración de permisos:

- Han de ser declarados en el manifiesto.
- Los permisos peligrosos se otorgan al arrancar el dispositivo sin la verificación del usuario.

○ Notificaciones:

- No son soportadas. Al no existir la barra de estado, no podrían mostrarse.



CAMBIOS DE COMPORTAMIENTO (III)

○ Soporte para servicios de Google:

- Soporte para muchas de las Apis de Google.
- Si requieren de una entrada directa del usuario NO.
- Cada versión de Android Things incluye la última versión estable de Google Play Services.
- No podrá actualizarse a través de Google Play.

Servicios soportados	Servicios no soportados
Awareness, Cast, Google Analytics for Firebase, Firebase Authentication, Firebase Cloud Messaging (FCM), Firebase Crash Reporting, Firebase Realtime Database, Firebase Remote Config, Firebase Storage, Fit, Instance ID, Location, Maps, Nearby Connections, Nearby Messages, Places, Mobile Vision, SafetyNet	AdMob, Android Pay, Drive, Firebase App Indexing, Firebase Dynamic Links, Firebase Invites, Firebase Notifications, Play Games, Sign-In



3 - Raspberry Pi 3

- *Low Cost Single Board Computers*
- ordenador barato, integrado en una pequeña placa para resolver problemas específicos.
- Fase de prototipo de un sistema embebido
- Ver comparativa con otras placas similares
 - Otros modelos pueden costar la mitad: Orange Pi o Raspberry Pi Zero
 - Pero Raspberry Pi 3 es el modelo más popular
 - Y es compatible con Android Things
- Se basa en SoC BCM2837 de Broadcom.
 - CPU: ARM de 64bits a 1.2GHz, quad-core
 - RAM: 1GB
 - GPU, GPIO, ...



LOW COST SINGLE BOARD COMPUTERS

Raspberry Pi 3 B (35 € PcCom-
ponentes, 30 € AliExpress)
4 x ARM Cortex-A53 a 1,2 GHz,
RAM 1G, Ethernet, WiFi, Bluetooth,
HDMI, 4xUSB 2.0, Jack audio,
ranura SD



Raspberry Pi Zero W (13,9 €
AliExpress)
Broadcom BCM2835 a 1 GHz,
RAM 512M, HDMI, 1x microUSB,
ranura SD



Orange Pi Plus 2 (12,3 € AliEx-
pres)
H3 Quad-core|Cortex-A7 a 1,6
GHz, H.265/HEVC 4K, RAM 2G,
Ethernet, WiFi, Bluetooth, HDMI,
4xUSB 2.0, ranura SD, 16 G Flash



Intel Edison (91 €)

2 x Atom Silvermont a 0,5GHz + 1 x
Quark a 0,1GHz, RAM 4G, WiFi,
Bluetooth, USB 2.0, ranura SD

[https://software.intel.com/es-
es/iot/hardware/edison](https://software.intel.com/es-es/iot/hardware/edison)



RASPBERRY PI 3 (+)

- 1.- 4 x USB 2,0 hub
- 2.- Ethernet 10/100 Mbps (+ 1000)
- 3.- Alimentación microUSB ($\geq 2,5A$)
- 4.- HDMI (full HD)
- 5.- cámara CSI
- 6.- Jack 3,5mm audio/v. compuesto
- 7.- tarjeta de memoria micro SD
- 8.- conector de pantalla DSI
- 9.- WiFi 802.11n y Bluetooth 4.1 (+ 802.11ac y Bluetooth 4.2 LS BLE)
- 10.- CPU Broadcom BCM2837 64bit quad-core 1,2GHz, 1 GB RAM
(+ BCM2837B0 a 1,4GHz)
- 11.- Controlador USB/Ethernet
- 12.- Conector de 40 pines con salidas GPIO



NOTA: En Raspberry Pi 3+ de momento no se puede instalar Android Things

ALTERNATIVAS PARA EL SISTEMAS OPERATIVOS (I)



UNIVERSIDAD
POLITECNICA
DE VALENCIA

- SO oficial Raspbian, creado para el dispositivo. Distribución Linux basada en Debian. Fedora o Arch Linux, tienen versiones especiales para Raspberry Pi.
- No hay una imagen oficial de Android. Existen alternativas de terceros: RaspAnd, podremos ejecutar aplicaciones y juegos de Android. Incorpora el gestor de contenido Kodi y Firefox.
- Gestor de contenido multimedia: OSMC o OpenElec, ambos basado en Kodi. Pi MusicBox solo música.
- Almacenamiento conectado en red (NAS). Ver foro. El chip que controla USB y Ethernet es el mismo.



ALTERNATIVAS PARA EL SISTEMAS OPERATIVOS (II)

- Video consola: RetroPie, permite ejecutar videojuegos antiguos (+50 sistemas). Además, permite instalar Kodi como reproductor multimedia.
- Kano, sencillo sistema operativo para que los niños tengan un primer contacto con la informática.
- Desarrollo de dispositivos de Internet de las cosas: además de Linux, puedes instalar Windows 10 IoT Core, la propuesta de Microsoft para este mundo. Necesitarás un PC con Windows y Visual Studio. Y por supuesto, Android Things.



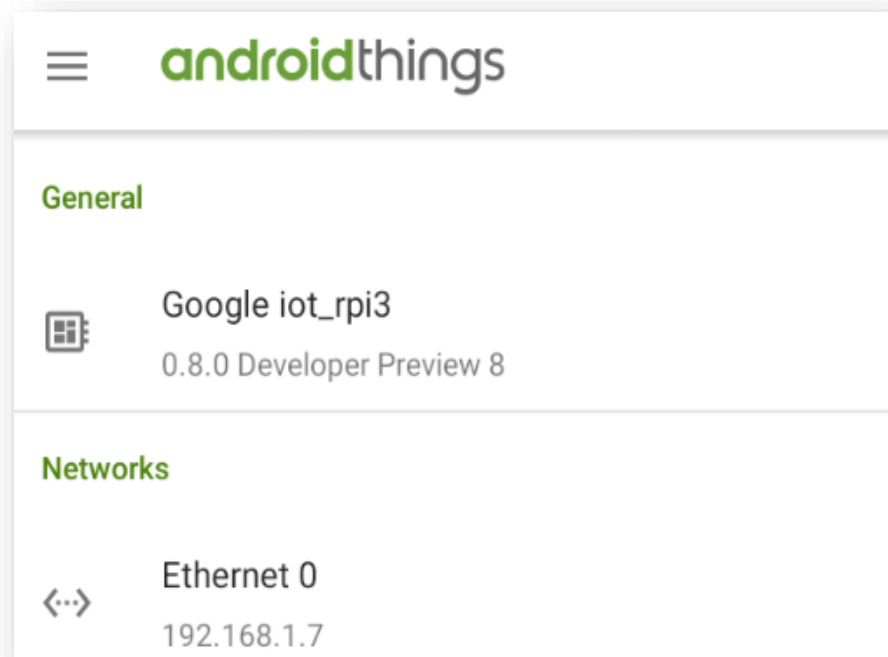
4 - INSTALACIÓN DE ANDROID THINGS

- Desde la consola entra en *Tools / DOWNLOAD*.
<https://developer.android.com/things/preview/download.html>
- Ejecuta **Android Things Setup Utility** y sigue las instrucciones
- Necesitarás una tarjeta microSD al menos 8GB y adaptador para leer tarjeta microSD en el PC
- Grabación 20 minutos y verificación 20 min
- Inserta la tarjeta en la ranura de la Raspberry Pi.



CONFIGURAR INTERNET CON MONITOR

- *Con monitor y teclado*
- *Configura WiFi o conecta un cable Ethernet*



CONFIGURAR ETHERNET SIN MONITOR

- Conecta *un cable Ethernet*
- La Raspberry Pi obtendrá una IP por DHCP
- Descubre cual es la IP
 - Conéctate a tu switch
 - Utiliza un programa de escaneo de IP (por ejemplo: <http://angryip.org/>).
- Desde la línea de comando escribe:
`adb connect <dirección-ip>`



CONFIGURAR WiFi CON SETUP UTILITY

- Ejecuta la herramienta Android Things Setup Utility
- Selecciona:
 - 2 - Set up Wi-Fi on an existing Android Things device
- Te pedirá que conectes el dispositivo con un cable Ethernet.
- Introduce el nombre SSID de la red y la contraseña



CONFIGURAR WIFI CON ADB

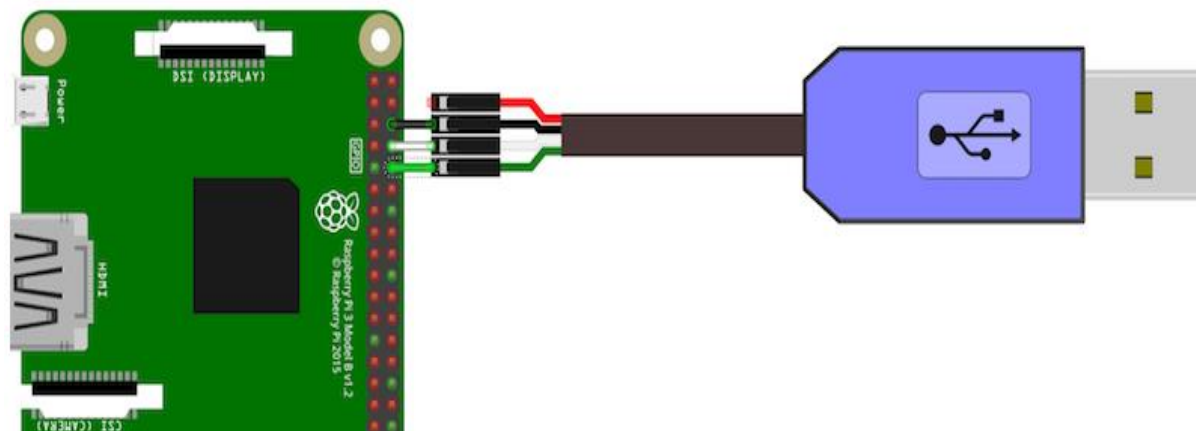
- Asigna una IP usando un método alternativo.
- Para abrir una consola Shell, elige una opción:
 - Ejecuta el comando: **adb connect <dirección-ip>**
 - Conecta un cable serie por *USB*. (ver siguiente ejercicio)
- Ejecuta el comando: **adb shell**
Se abrirá un shell Linux en el dispositivo.
- Envía una intención para arrancar el servicio de Wi-Fi:

```
$ am startservice \  
-n com.google.wifisetup/.WifiSetupService \  
-a WifiSetupService.Connect \  
-e ssid <SSID_de_tu_red> \  
-e passphrase <contraseña>
```



CONEXIÓN POR CABLE USB

- Para abrir un Shell Linux desde tu PC
 - Consigue a un cable USB a UART
 - Conéctalo como se muestra en la figura y al PC:

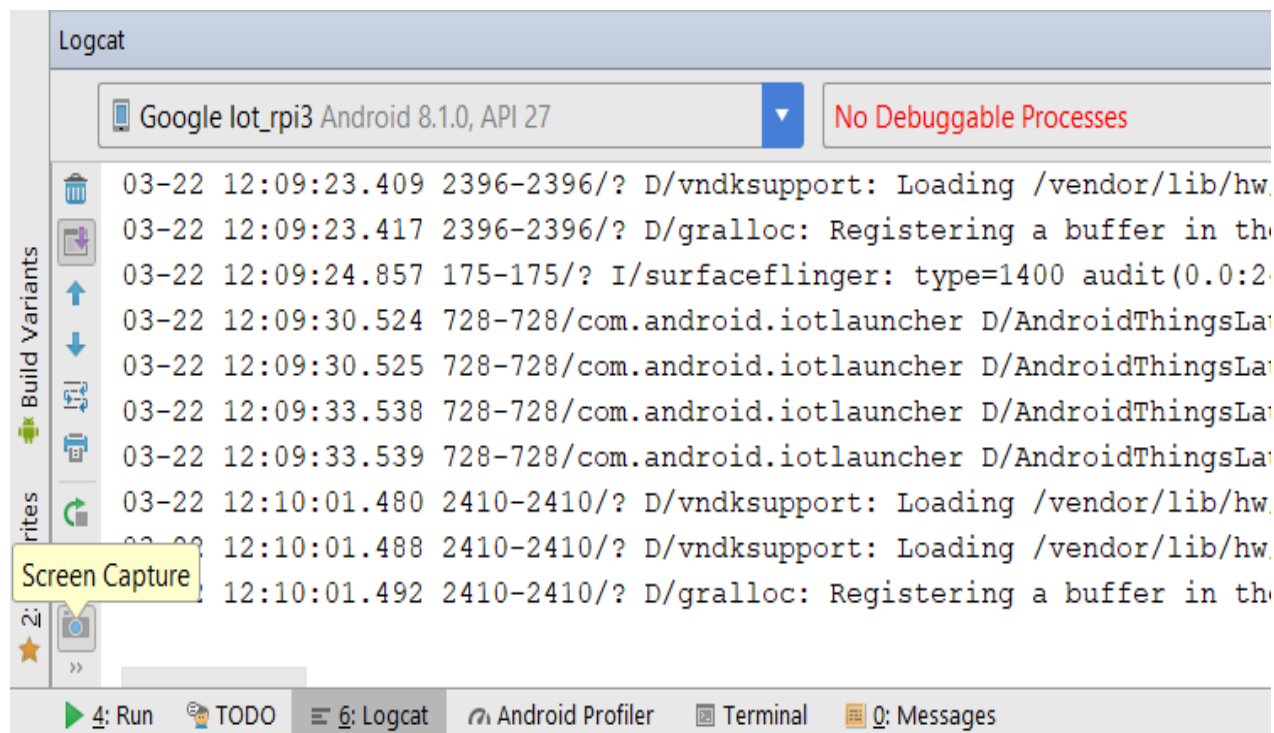


- Abre un programa de terminal como PuTTY (Windows), Serial (Mac OS) o Minicom (Linux).
- Parámetros del puerto serie - velocidad: 115200 baudios, bits de datos: 8, paridad: ninguna, bits de parada: 1.



ACCEDER AL DISPOSITIVO DESDE ANDROID STUDIO

- Ejecuta el comando: **adb connect <dirección-ip>**
- En la pestaña *Logcat* y selecciona el dispositivo



- *Screen Capture* : ver pantalla
- *Android Profiler* : ver estado CPU, memoria y red



PRIMER PROYECTO ANDROID THINGS

- Crea un nuevo proyecto con los siguientes datos:

Application name: *Android Things*

Package name: `org.example.androidthings`

☐ Phone and Tablet

☒ Android Things

Minimum SDK: API 27 Android 8.1 (Oreo)

Add an activity: *Android Things Empty Activity*

Activity Name: `MainActivity`

☒ Generate a UI layout File

Layout Name: `activity_main`

- Navega por los ficheros generados

- No se han añadido recursos para los iconos.



OTRAS DIFERENCIAS

○ *AndroidManifest.xml*

```
<uses-library android:name="com.google.android.things" />
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.IOT_LAUNCHER" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

○ *build.gradle (Module: app)*

```
dependencies {
    ...
    <u>compileOnly 'com.google.android.things:androidthings:+'</u>
}
```



PERMISOS



UNIVERSIDAD
POLITECNICA
DE VALENCIA

- Declara los permisos en el manifiesto.
- Todos los permisos normales se otorgan en el momento de la instalación.
- Los permisos peligrosos no se otorgan hasta el siguiente reinicio del sistema.



EJEMPLOS DE PROYECTOS

- o <https://developer.android.com/samples/?technology=iot>

JAVA IOT

Bluetooth Audio

This sample demonstrates the use of Android Bluetooth APIs for audio from an Android...

JAVA IOT

Bluetooth GATT Server (Java)

This application demonstrates accessing the Android API from within an Android Things...

KOTLIN IOT

Bluetooth GATT Server (Kotlin)

This application demonstrates accessing the Android API from within an Android Things...

JAVA IOT

Button and LED (Java)

This Android Things sample demonstrates how to use a button input UserDriver to liste...

KOTLIN IOT

Button and LED (Kotlin)

This Android Things sample demonstrates how to use a button input UserDriver to liste...

JAVA IOT

Cloud Doorbell

The Android Things Doorbell sample demonstrates how to create a "smart" doorbell. The...



JAVA IOT

Cloud IoT Sensor Hub

This sample shows how to implement a sensor hub on Android Things that collects...

JAVA IOT

Driver

Samples for Android Things peripheral drivers located in the [https://github.com/androidthi...

JAVA IOT

Google Assistant API

This sample shows how to call the Google Assistant Service from Android Things using...



CONCLUSIONES: VENTAJAS DE ANDROID THINKG

- Desarrolla con Android SDK y Android Studio.
- Utiliza todo el potencial de herramientas creadas para Android (Firebase, TensorFlow, ...)
- Usa hardware de producción certificado.
- Añade pantallas, cámaras e interfaces de audio accesibles a través del API de Android.
- Integra periféricos adicionales a través de entradas/salidas estándar (GPIO, I2C, SPI, UART, PWM)
- Utiliza la consola de Android Things para enviar actualizaciones de seguridad online.



CONCLUSIONES: LIMITACIONES DE ANDROID THINKG

- Requiere hardware potente y caro.
- Solo para productos que no requiera administración de energía (por ejemplo, bajo consumo de energía cuando está inactivo).
- Ha de estar conectado a Internet a través de WiFi o Ethernet (actualizaciones OTA).

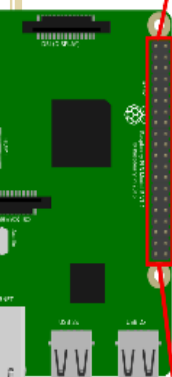


5 – USOS DE ENTRADAS / SALIDAS

- Se trata de la parte más característica de A.Things
- Nos van a permitir interaccionar con el entorno.
- Tipos: GPIO, PWM, I²C, UART y SPI
- Se controlan mediante una serie de registros del chip BCM2835. Pero al usar un S.O. estamos obligados a hacerlo a través de la API.
 - Este API puede ser demasiado lento para controlar algunos dispositivos
- Disponemos de un conector de 40 Pines
- Interesante utilizar extensor:



CONEXIONES DE ENTRADA / SALIDA



3.3V PWR	1		2	5V PWR
GPIO2 (SDA1 , I2C)	3		4	5V PWR
GPIO3 (SCL1 , I2C)	5		6	GND
GPIO4 (GPIO_GCLK)	7		8	(UART_TXD0) GPIO14
GND	9		10	(UART_RXD0) GPIO15
GPIO17 (GPIO_GEN0)	11		12	(GPIO_GEN1) GPIO18
GPIO27 (GPIO_GEN2)	13		14	GND
GPIO22 (GPIO_GEN3)	15		16	(GPIO_GEN4) GPIO23
3.3V PWR	17		18	(GPIO_GEN5) GPIO24
GPIO10 (SPI0_MOSI)	19		20	GND
GPIO9 (SPI0_MISO)	21		22	(GPIO_GEN6) GPIO25
GPIO11 (SPI0_CLK)	23		24	(SPI_CE0_N) GPIO8
GND	25		26	(SPI_CE1_N) GPIO7
ID_SD (I2C EEPROM)	27		28	ID_SC (I2C EEPROM)
GPIO5	29		30	GND
GPIO6	31		32	GPIO12
GPIO13	33		34	GND
GPIO19	35		36	GPIO16
GPIO26	37		38	GPIO20
GND	39		40	GPIO21

- **GND** (8) masa
 - **5V** (2) fusible cortocircuito.
 - **3,3V** (2) consumo < 1A
 - **GPIO** (24) entrada o salida. -
entrada: entre 0 y 3,3V.
- salida: consumo < 50mA
 - **PWM** (2 canales) salida
- E/S serie:
- **UART** (2)
 - **I²C** (4)
 - **SPI** (5)

(algunos con varias funciones)



CONEXIONES GPIO

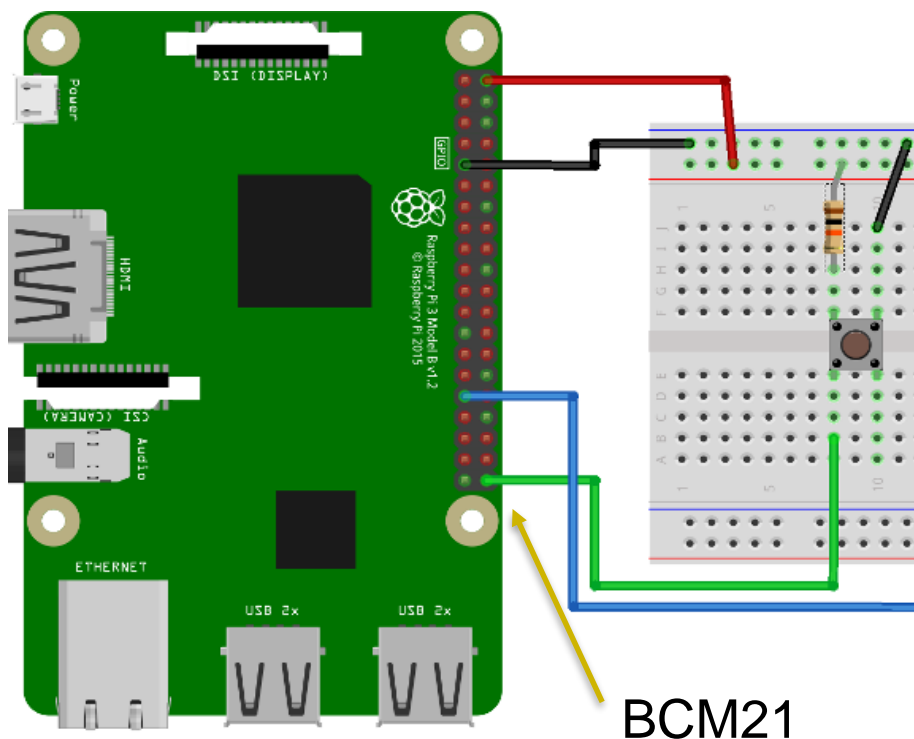
- General Purpose Input/Output
- para realizar una entrada o salida binaria, con algún dispositivo exterior.
- Disponemos de un máximo de 24
- Se programan como entrada o salida
- como salida:
 - nivel bajo: 0V, nivel alto: 3,3V.
 - consumo máximo 50mA.
- como entrada:
 - si el voltaje cercano a 0V -> 0 si cercano a 3,3V -> 1.
- No hay E/S analógicas



ENTRADA GPIO

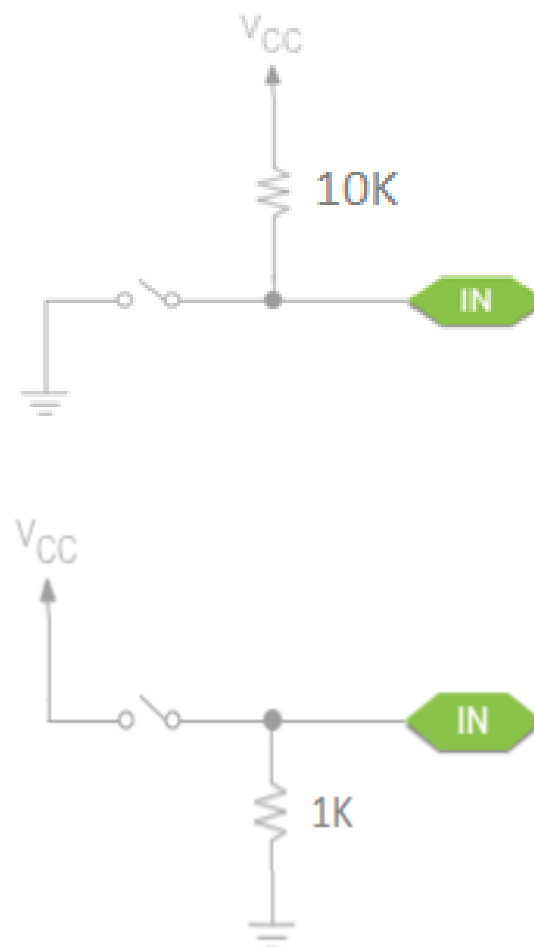
○ Conexiones:

- Pulsador 4 pines
- Resistencia pull-up de 10KΩ
- Tablero prototipos



QUE ES UNA RESISTENCIA PULL-UP / PULL-DOWN

- En el ejercicio hemos usado:
- Si eliminas la resistencia y el pulsador está abierto, cable desde la entrada, sin conectar a ningún sitio, no sería ni 0 ni 1 (alta impedancia).
- Podría actuar como una antena, introduciendo entras no deseadas.
- Al poner la resistencia, entrada a 1.
- Cuando se pulse entrada a 0
- Inconveniente, pequeño consumo (menor de 1mA) al pulsar
- Si queremos por defecto 0, ponemos resistencia de pull-down.



ENTRADA GPIO: CÓDIGO

```
private static final String BOTON_PIN = "BCM21"; //Puerto GPIO
private Gpio botonGpio;

@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    PeripheralManager manager = PeripheralManager.getInstance();
    try {
        botonGpio = manager.openGpio(BOTON_PIN); //crea conexión
        botonGpio.setDirection(Gpio.DIRECTION_IN); //es entrada
        botonGpio.setEdgeTriggerType(Gpio.EDGE_FALLING);
        // 3. Habilita eventos de disparo por flanco de bajada
        botonGpio.registerGpioCallback(callback); //registra callback
    } catch (IOException e) {
        Log.e(TAG, "Error en PeripheralIO API", e);
    }
}
```



ENTRADA GPIO: CÓDIGO (II)

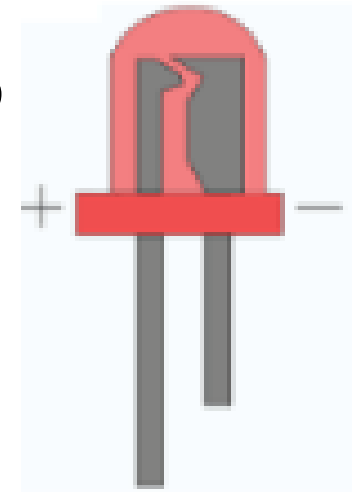
```
private GpioCallback callback = new GpioCallback() {
    @Override public boolean onGpioEdge(Gpio gpio) {
        try {
            Log.e(TAG, "cambio botón "+Boolean.toString(gpio.getValue()));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return true; // 5. devolvemos true para mantener callback activo
    }
};

@Override protected void onDestroy() {
    super.onDestroy();
    if (botonGpio != null) { // 6. Cerramos recursos
        botonGpio.unregisterGpioCallback(callback);
        try {
            botonGpio.close();
        } catch (IOException e) {
            Log.e(TAG, "Error en PeripheralIO API", e);
        }
    }
}
```



LED

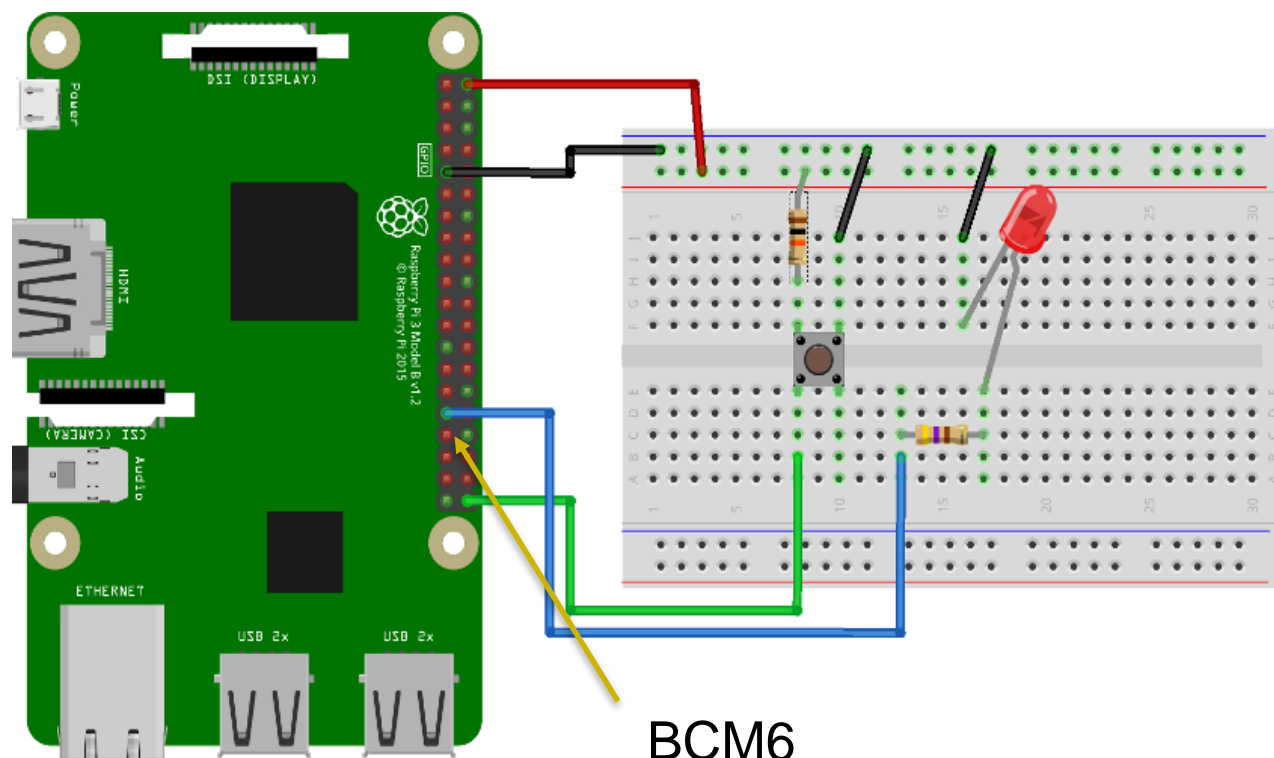
- Diodo emisor de luz (Light Emitting Diode)
- Diodo la corriente solo puede pasar del ánodo (pata larga) al cátodo (pata más corta)
- NUNCA conectes de 3,3/5V a GND.
- Limita la corriente utilizando una resistencia.
- Ley de Ohm: $V = I \cdot R$
- $I_{\max} = 20\text{mA}$ $V_{\text{LED}} = 1,8\text{V} - 2,1\text{V}$ rojo, amarillo, verde
 $V_{\text{LED}} = 3\text{V} - 3,8\text{V}$ azul, violeta o blanco
- LED rojo con una caída de 1,8V alimentado a 9V
$$R = V / I = (9\text{V} - 1,8\text{V}) / 17\text{mA} = 423,5\Omega \approx 470 \Omega$$
- Para una alimentación de 3,3V:
$$R = (3,3\text{V} - 1,8\text{V}) / 17\text{mA} = 88,2\Omega \approx 100\Omega \text{ (o } 120 \Omega)$$



SALIDA GPIO

○ Conexiones:

- LED rojo
- Resistencia para limitar voltaje ($120\ \Omega$)



fritzing



SALIDA GPIO: CÓDIGO

```
private static final int INTERVALO_LED = 1000; // Intervalo (ms)
private static final String LED_PIN = "BCM6"; // Puerto GPIO
private Handler handler = new Handler(); // Handler parpadeo
private Gpio ledGpio;
```

```
@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    PeripheralManager manager = PeripheralManager.getInstance();

    ...
    try {
        ledGpio = manager.openGpio(LED_PIN); // 1. Crea conexión GPIO
        ledGpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);
        // 2. Se indica que es de salida
        handler.post(runnable); // 3. Llamamos al handler
    } catch (IOException e) {
        Log.e(TAG, "Error en PeripheralIO API", e);
    }
}
```



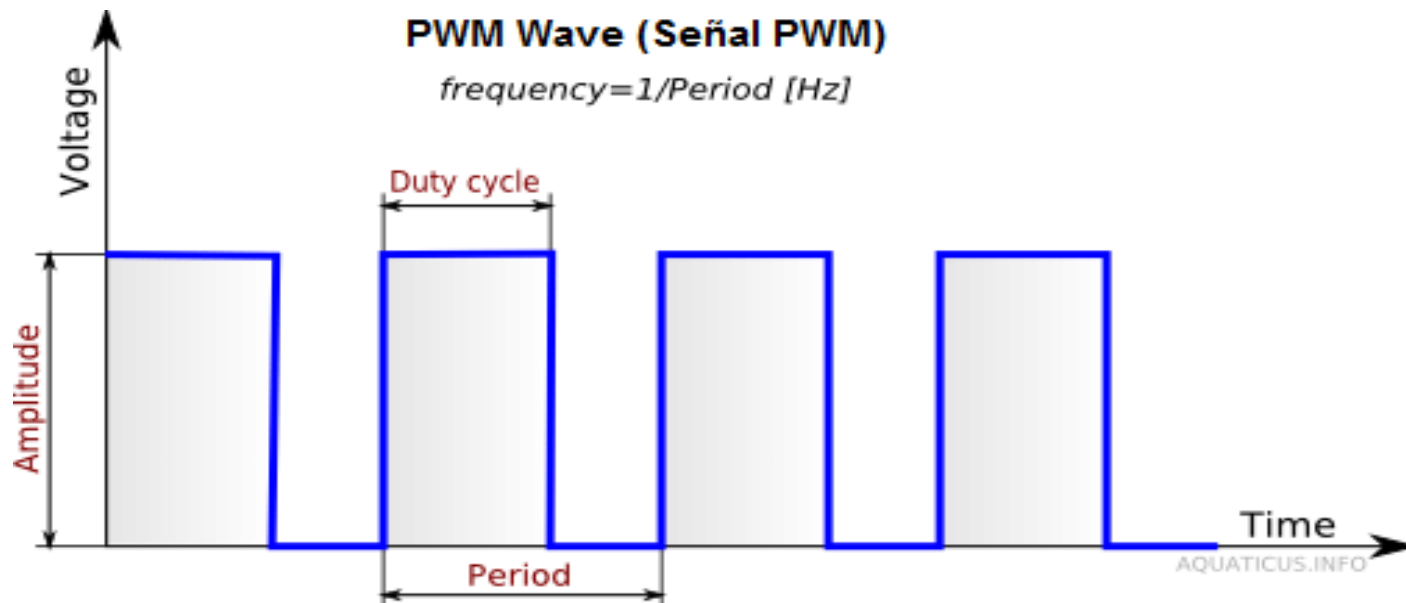
SALIDA GPIO: CÓDIGO (II)

```
private Runnable runnable = new Runnable() {  
    @Override  
    public void run() {  
        try {  
            ledGpio.setValue(!ledGpio.getValue()); //Cambio valor LED  
            handler.postDelayed(runnable, INTERVALO_LED);  
            //Programamos siguiente llamada dentro de INTERVALO_LED  
        } catch (IOException e) {  
            Log.e(TAG, "Error en PeripheralIO API", e);  
        }  
    }  
};
```



SALIDAS PWM

- Una salida GPIO solo puede tomar valores, 0 o 1
- Si quisiéramos bajar el brillo de un LED bajar su voltaje
- Como no tenemos salidas analógicas, podríamos aplicar una señal que esté a 1 solo una fracción del tiempo.
- Salida PWM (Pulse Width Modulation) (en motores paso a paso o controlar brillo pantallas LCD)



SALIDAS PWM: CÓDIGO

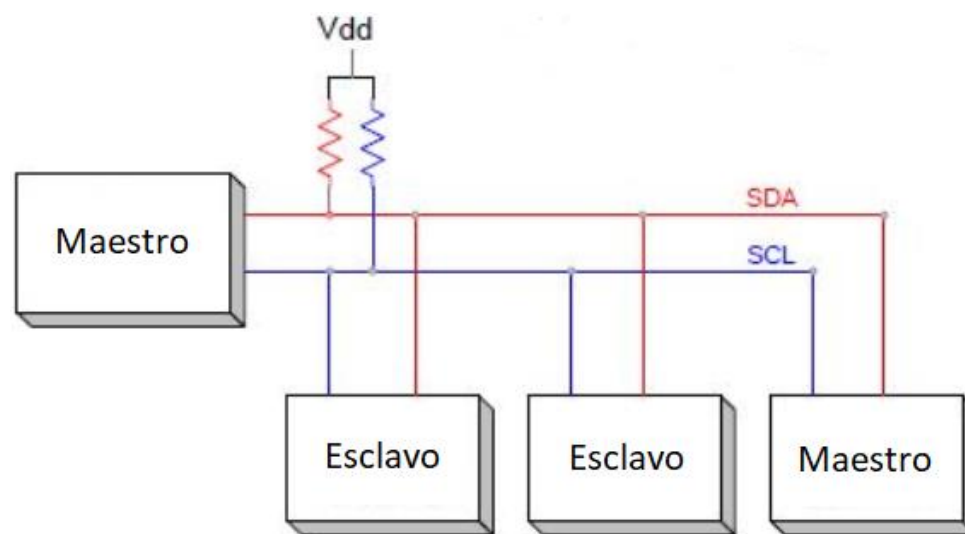
```
private static final int PORCENTAGE_LED_PWM = 25; // % encendido
private static final String LED_PWM_PIN = "BCM18"; // Puerto GPIO
private Pwm ledPwm;

@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    PeripheralManager manager = PeripheralManager.getInstance();
    ...
    try {
        ledPwm = manager.openGpio(LED_PWM_PIN); // 1. Crea conexión GPIO
        ledPwm.setPwmFrequencyHz(120);          // 2. Configuramos PWM
        ledPwm.setPwmDutyCycle(PORCENTAGE_LED_PWM);
        ledPwm.setEnabled(true);
    } catch (IOException e) {
        Log.e(TAG, "Error en al acceder a salida PWM", e);
    }
}
```



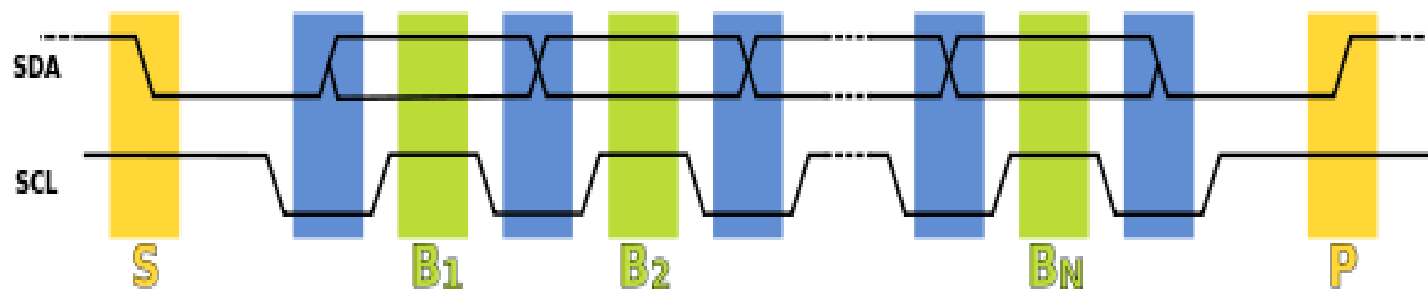
X.3 EL BUS I²C

- Permite interconectar varios dispositivos (no UART)
- Bus serie y síncrono
- Desarrollado por Philips en los 80, para reducir el número de pines en los chips
- Dos líneas: SDA – datos y SCL – reloj ()
 - resistencias pull-up, nivel alto por defecto.
- Roles: maestro / esclavo



SINCRONISMO DE TRAMA Y BIT EN I²C

- La señal de reloj es introducida por el maestro:
 - La velocidad puede variar según marque el maestro.
- valores preestablecidos: 0,1 0,4 1,0 3,4 y 5,0 Mbits/s
- La unidad de datos transferida siempre es un byte.

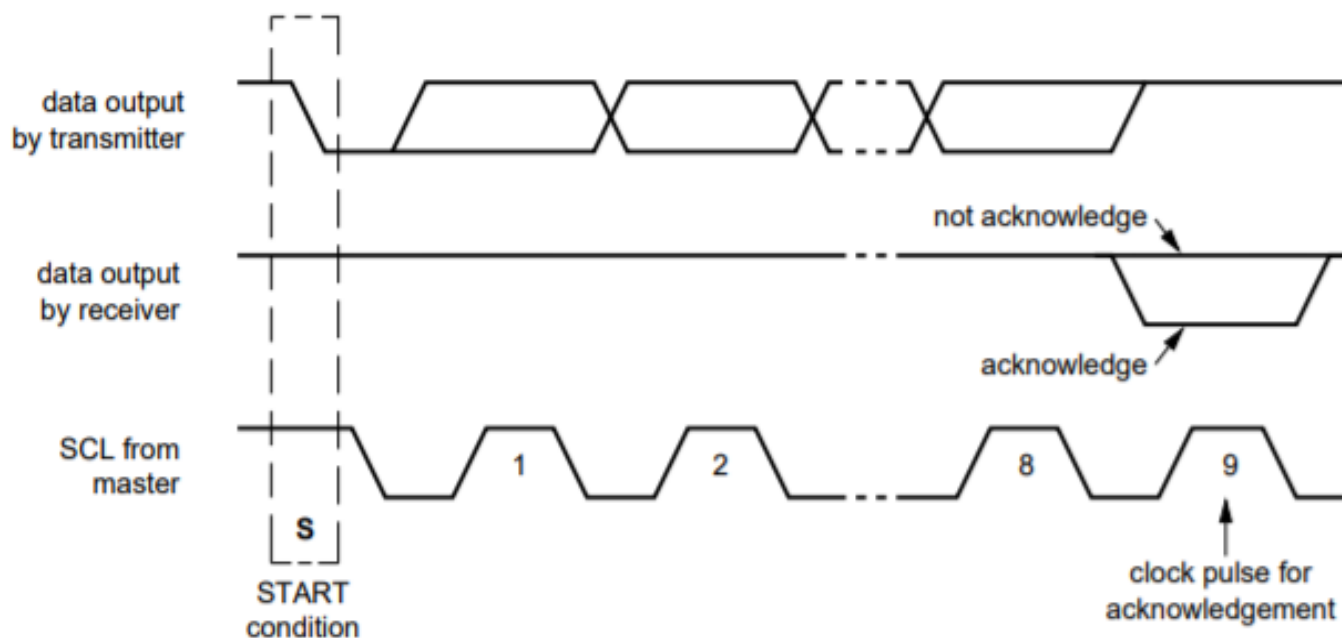


- **S**: arranque - flanco de bajada con reloj alto.
con reloj bajo cambiamos datos
- **B**: con reloj alto el dato no puede cambiar
- **P**: parada – flanco de subida con reloj alto



BIT DE RECONOCIMIENTO (ACK)

- Tras cada byte se espera un bit de reconocimiento por parte del receptor. 0 – positivo 1 – negativo



- Fin transferencia: tras el último byte leído por el maestro ha de indicar un Not ACK, luego un bit P.



DIRECCIONAMIENTO EN I²C

- El primer byte enviado por el maestro:
- Dirección 7 bits:
 - 4 más significativos establecidos por fabricante
 - 3 últimos pueden ser configurados por jumpers
 - hasta 8 (2^3) circuitos iguales en el bus
- Bit R/W:
 - 1: leer del esclavo 0: para escribir en esclavo.



7	6	5	4	3	2	1	0
MSB							LSB
1	0	0	1	A2	A1	A0	R/W

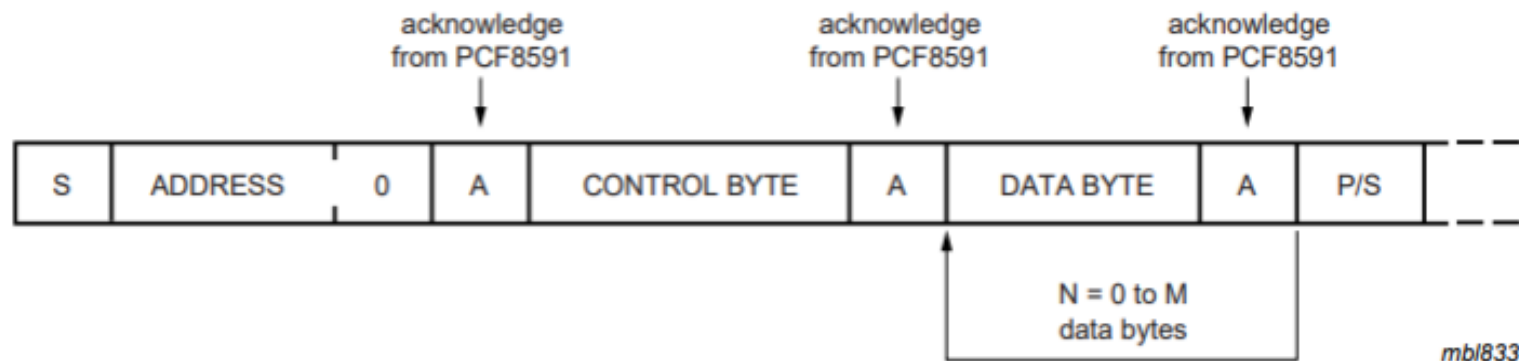


EL CONVERSOR A/D Y D/A PCF8591

- chip con 4 entradas A/D y una salida D/A
- resolución es de 8 bits
- se conecta como esclavo a un bus I²C.
- velocidad máxima de conversión dada por el bus
- Dirección: $1001J_2J_1J_0$



PCF8591: CONVERSIÓN D/A

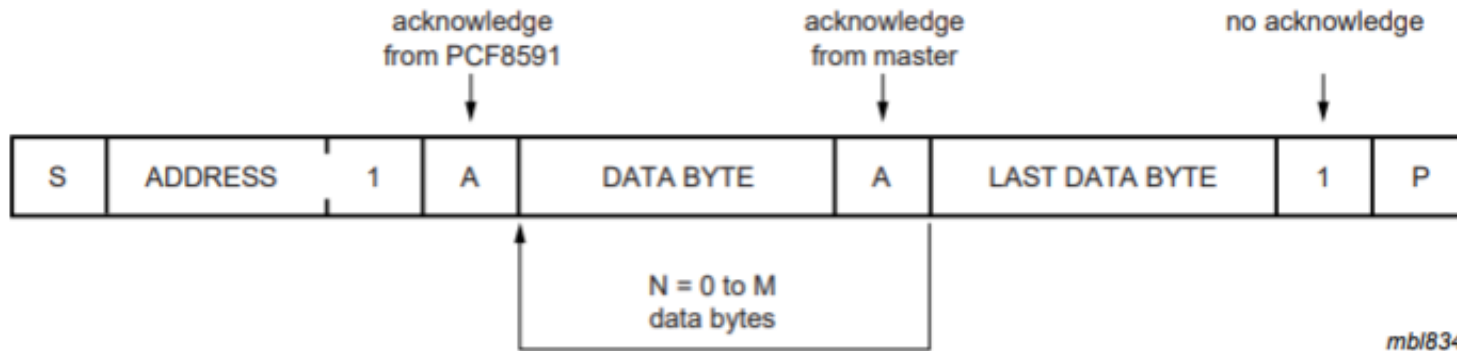


- dirección + R/W = 0 + byte de control + datos
- Byte de control:

bit	valor	significado
7	0	valor fijo
6	0/1	1 activamos la salida analógica
5,4	00	4 entradas analógicas AIN0-AIN3 respecto a GND
	01	3 entradas analógicas respecto a AIN3
	10	AIN0 y AIN1 respecto a GND, AIN2 respecto a AIN3
	11	AIN0 respecto a AIN1, AIN2 respecto a AIN3
3	0	valor fijo
2	0/1	1 activamos auto incremento
1,0	00	canal 0
	01	canal 1
	10	canal 2



PCF8591: CONVERSIÓN A/D



- dirección + R/W = 1 + datos
- los datos se configuran en el byte de control.
- la conversión se vuelve a mandar hasta que maestro mande reconocimiento negativo y bit de parada.
- si modo auto incremento, se enviarán los diferentes canales uno tras otro.
- data sheet: <https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf>



BUS I²C CON ANDROID THINGS (RAW)

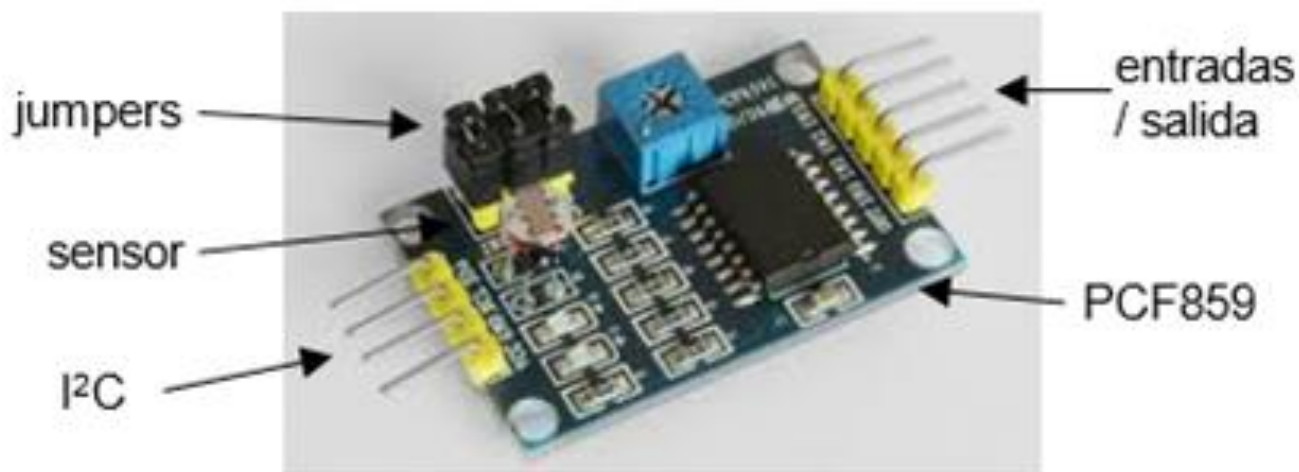
```
try {  
    I2cDevice i2c = manager.openI2cDevice("I2C1", 0x48);  
                                                // nombre y dirección  
  
    byte[] config = new byte[2];  
    config[0] = (byte) 0x00;                    // byte de control  
    config[1] = (byte) 0xFF;                    // valor de salida  
    i2c.write(config, config.length);           // escribimos 2 bytes  
  
    byte[] buffer = new byte[4];  
    i2c.read(buffer, buffer.length);            // leemos 4 bytes  
  
    i2c.close();                                // cerramos i2c  
    i2c = null;                                 // liberamos memoria  
} catch (IOException e) {  
    Log.e(TAG, "Error en al acceder a dispositivo I2C", e);  
}
```

- Consultar protocolo SMBus en documentación oficial



EJERCICIO BUS I²C RAW

- Conectar 5V, GND, SDA y SCL a la Raspberry Pi
- Conectar un LED en OUT (con resistencia)
 - Aplicar salida de 255, 128 y 0
- Conectar IN0...IN3 a GND, 3,3V, 5V y un potenciómetro o foto resistencia
- Probar modo auto incremento



USAR UN DRIVER PARA PCF8591

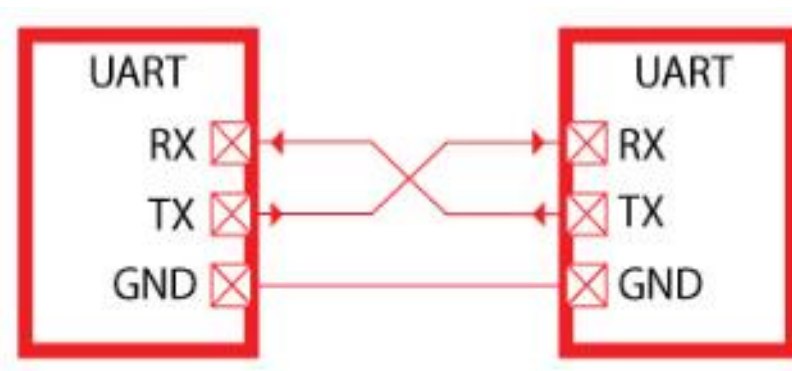
- Programar estos chips es complejo (data sheet)
- En muchas ocasiones existe un driver para manejarlo
<https://github.com/davemckelvie/things-drivers/tree/master/pcf8591>
- Añadir dependencia:
`'nz.geek.android.things:things-drivers:1.8.0'`
- Pedir permisos: `MANAGE_INPUT_DRIVERS` y `USE_PERIPHERAL_IO`
- Código:

```
I2cAdc.I2cAdcBuilder builder = I2cAdc.builder();
I2cAdc adc = builder.address(0).fourSingleEnded()
                .withConversionRate(100).build();
adc.startConversions();
adc.readChannel(0);
```
- Este driver no permite conversión D/A.



UART

- Universal Asynchronous Receiver-Transmitter: interfaz serie muy usado en GPS, pantallas LCD, consolas ... (predecesor del USB)
- asíncrono, no utiliza una señal de reloj.

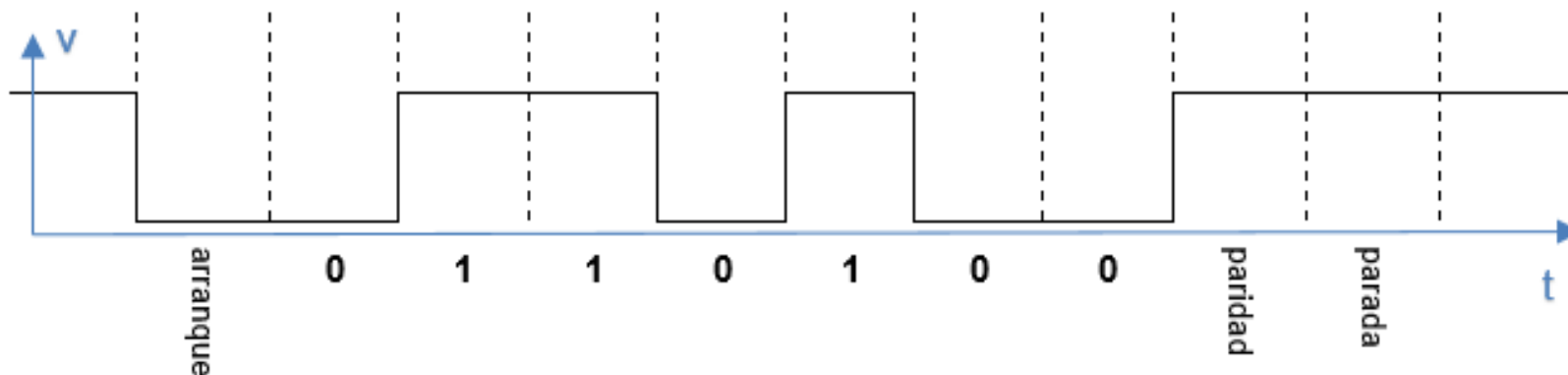


- Transmisión full-duplex, 2 dispositivos,
- Problema emisor y receptor han de fijar parámetros de transmisión



UART: TRANSMISIÓN

- La línea en reposo permanece a 1



- bit arranque:** 0
- bits de datos:** entre 5 y 8 (de LSB a MSB)
- bit de paridad:** *par* (la suma de “1” es par), *impar* (la suma de “1” es impar) o ninguna.
- bit parada:** 1 (1 o 2 periodos según Config.)
- velocidad:** 300, 1200, 4800, 9600, ..., 115200, ..., 1M, 2M baudios



UART: CONFIGURACIÓN

```
UartDevice uart;
```

```
...
```

```
try {  
    uart = manager.openUartDevice("UART0");  
    uart.setBaudrate(115200);  
    uart.setDataSize(8);  
    uart.setParity(UartDevice.PARITY_NONE);  
    uart.setStopBits(1);  
} catch (IOException e) {  
    Log.w(TAG, "Error iniciando UART", e);  
}
```

○ Para cerrarlo:

```
uart.close();
```



UART: ESCRIBIR / LEER

- Para escribir bytes utilizaríamos:

```
byte[] buffer = {...};
```

```
int bytesEscritos = uart.write(buffer, buffer.length());
```

- Si queremos escribir los caracteres de un String:

```
int bytesEscritos = uart.write(s.getBytes(), s.length());
```

- Los datos recibidos se almacenan en memoria FIFO.
Para extraerlos:

```
int bytesLeidos = uart.read(buffer, buffer.length);
```

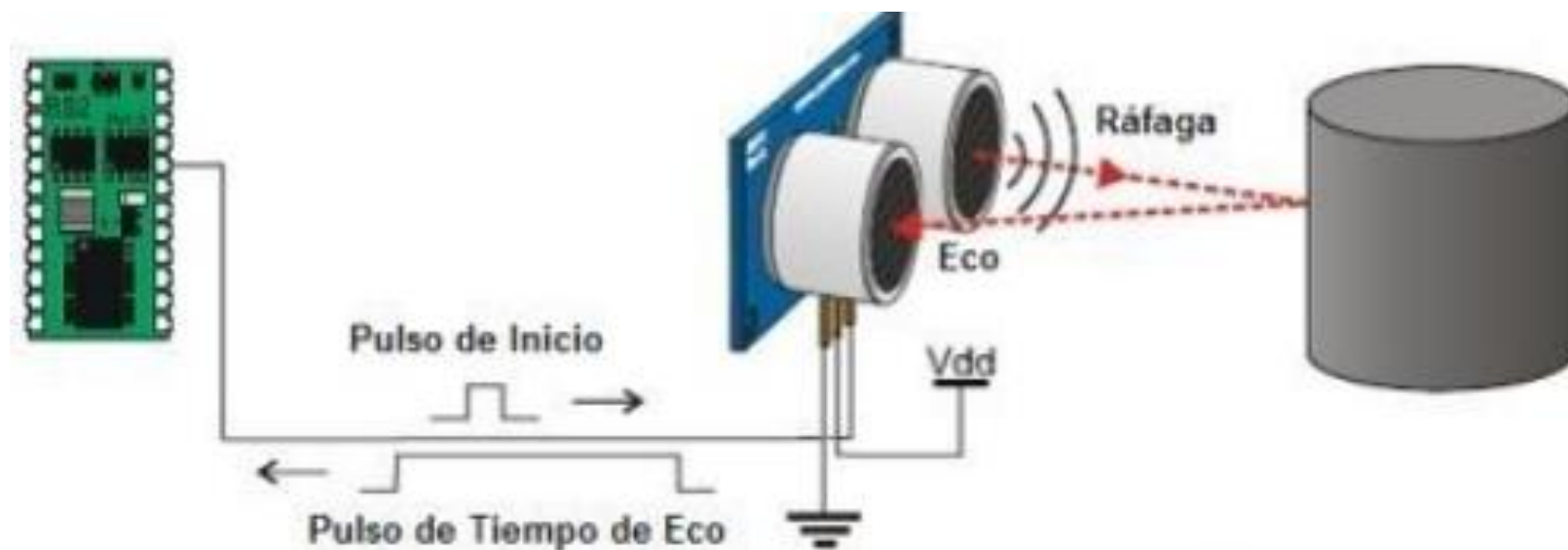
- Los datos recibidos se almacenan en memoria FIFO.
Para extraerlos:

```
uart.registerUartDeviceCallback( new UartDeviceCallback() {  
    @Override  
    public boolean onUartDeviceDataAvailable(UartDevice uart){  
    }  
}
```



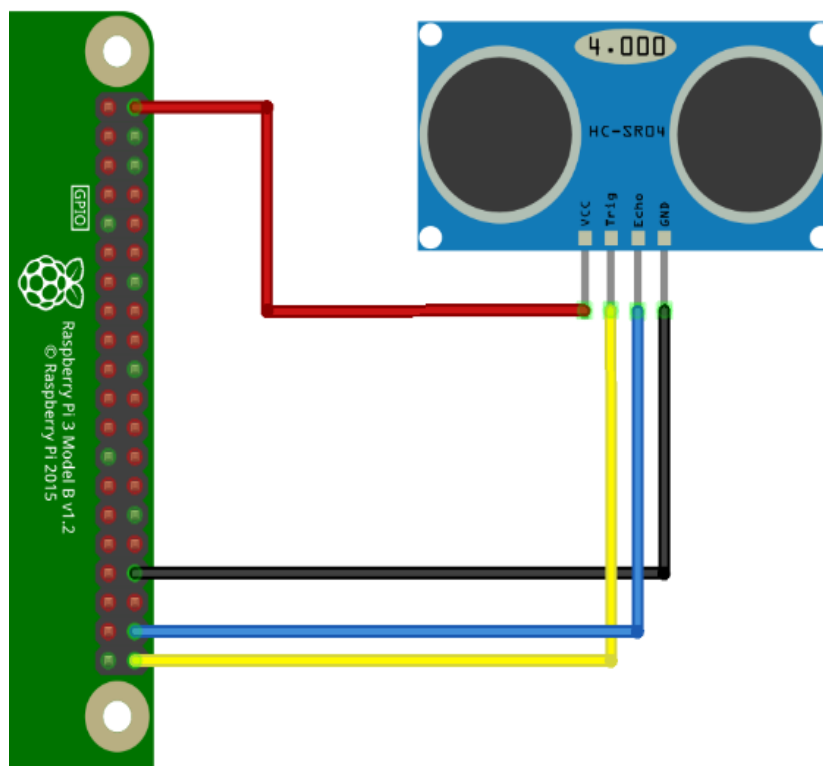
MEDIDOR DE DISTANCIA HC-SR04

- Funciona midiendo el tiempo que tarda un pulso (ping) de ultrasonido en rebotar en un objeto
- Consultar el [datasheet](#)



MEDIDOR DE DISTANCIA - ANDROID THINGS

- Una salida GPIO para el pulso de inicio
- Una entrada GPIO para medir tiempo de pulso de eco



MEDIDOR DE DISTANCIA - ANDROID THINGS



UNIVERSIDAD
POLITECNICA
DE VALENCIA

```
protected double leerDistancia() throws IOException,
                                   InterruptedException {

    mTrigger.setValue(false);
    Thread.sleep(0, 2000); // 2 msec
    mTrigger.setValue(true);
    Thread.sleep(0, 10000); //10 msec
    mTrigger.setValue(false);
    while (mEcho.getValue() == false) {
        hazAlgo = 0;
    }
    long tiempoIni = System.nanoTime();
    while (mEcho.getValue() == true) {
        hazAlgo = 1;
    }
    long tiempoFin = System.nanoTime();
    long anchoPulso = tiempoFin - tiempoIni;
    double distancia = (anchoPulso / 1000.0) / 58.23; //cm
    Log.i(TAG, "distancia: " + distancia + " cm");
    return distancia;
}
```



UTILIZANDO ANDROID THINGS

- Resultados algo pobres, poco fiables:
 - Al estar basado en un sistema operativo multiproceso, como Linux, no podemos asegurar que tenemos el procesador 100% disponible para nosotros.
- Si realizamos el control desde un microcontrolador (ej. Arduino), los resultados son mejores.
 - El procesador se encarga solo de esta tarea



PROBLEMAS AL CONTROLAR E/S

- No disponemos del procesador el 100% del tiempo.
- Las entradas/salidas GPIO en Android Things son muy lentas. Muchos sensores se controlan activando/desactivando una GPIO. En muchos casos la frecuencia necesaria no puede alcanzarse.
- No se dispone de E/S analógicas.
- La mayoría de sensores han sido diseñados para ser usados desde un microcontrolador. En sus especificaciones suelen incluirse código Arduino o librerías.



7 - USAR UN MICROCONTROLADOR ARDUINO COMO ESCLAVO

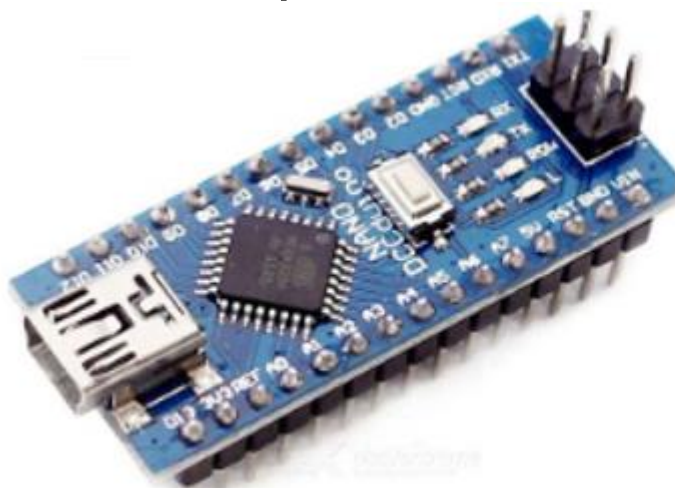


UNIVERSIDAD
POLITECNICA
DE VALENCIA



USAR UN MICROCONTROLADOR COMO ESCLAVO

- Resolvemos problemática anterior



Arduino Nano ([3,22 €](#))



ATTINY85 con USB ([1,34 €](#))

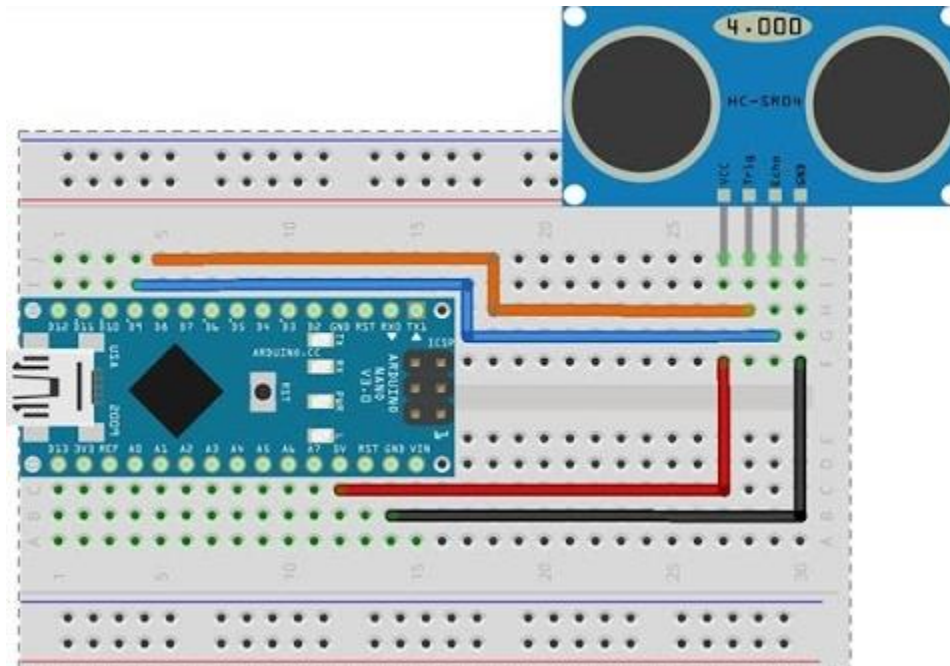
- Arduino Nano: microcontrolador ATmega328 16Mhz, 14 entradas/salidas (6 PWM y 6 A/D), 1KB EEPROM, 2KB de SRAM, 32KB flash.
- ATTINY85: 20Mhz, 5 entradas/salidas (2 PWM), 512B EEPROM, 512B SRAM (compat. Arduino)



MEDIDOR DISTANCIA CON ARDUINO



UNIVERSIDAD
POLITECNICA
DE VALENCIA



MEDIDOR DISTANCIA CON ARDUINO

```
const int EchoPin = 9;
const int TriggerPin = 8;

void setup() {
  Serial.begin(115200);
  pinMode(TriggerPin, OUTPUT);
  pinMode(EchoPin, INPUT);
}

void loop() {
  Serial.print("Distancia: ");
  Serial.println(distancia(TriggerPin, EchoPin));
  delay(1000);
}
```



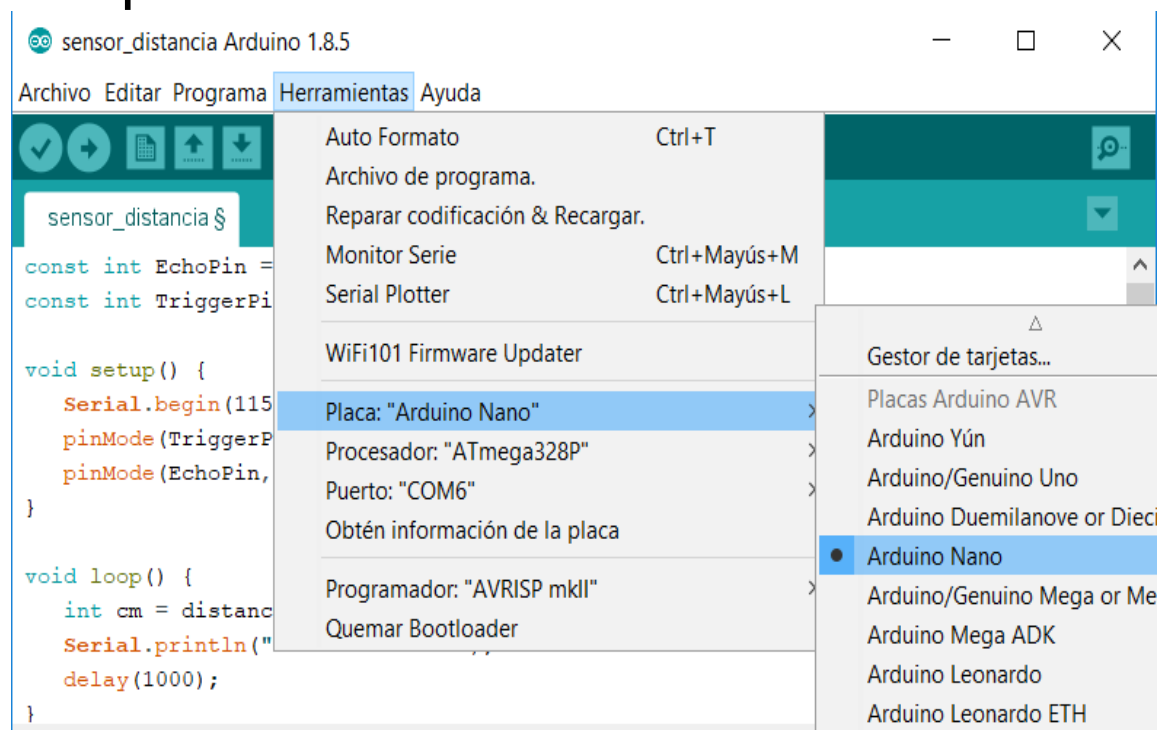
MEDIDOR DISTANCIA CON ARDUINO

```
int distancia(int TriggerPin, int EchoPin) {  
    long duracion, distanciaCm;  
    digitalWrite(TriggerPin, LOW); //nos aseguramos señal baja  
    delayMicroseconds(4);  
    digitalWrite(TriggerPin, HIGH); //generamos pulso de 10us  
    delayMicroseconds(10);  
    digitalWrite(TriggerPin, LOW);  
    duracion = pulseIn(EchoPin, HIGH); //medimos el tiempo pulso  
    distanciaCm = duracion * 10 / 292 / 2; //convertimos a distancia  
    return distanciaCm;  
}
```



INSTALACIÓN ARDUINO

- Instala de <https://www.arduino.cc/en/Main/Software>, busca la sección *Download the Arduino IDE*.
- Copia el código anterior
- Conecta la placa con un cable USB a tu ordenador.
- Selecciona placa:



INSTALACIÓN ARDUINO

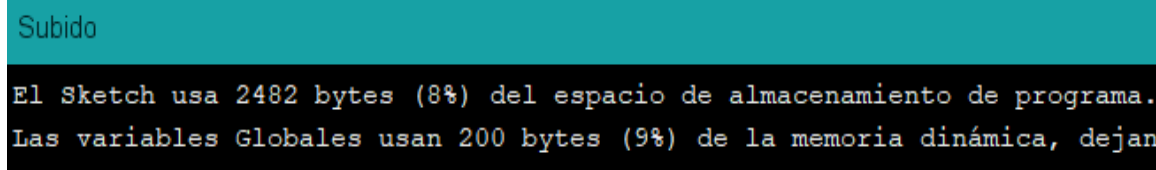
- Selecciona la opción Herramientas/Puerto/COMX

- Pulsa subir:

Archivo Editar Programa Herramientas Ayuda



- Si OK...

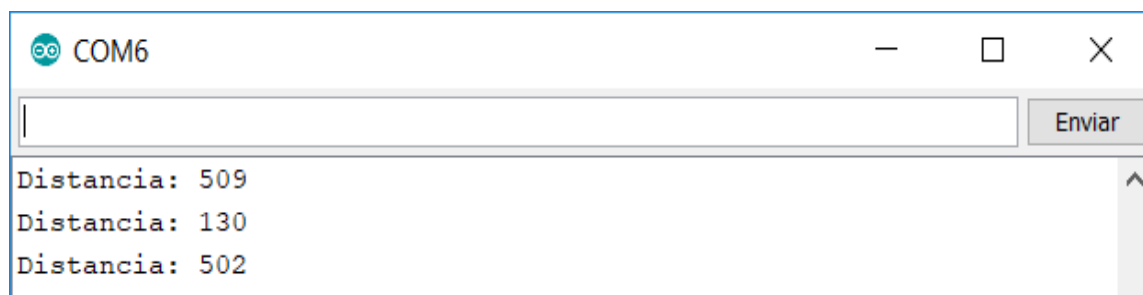


- Pulsa *Monitor Serie*:

Archivo Editar Programa Herramientas Ayuda



- Selecciona
115200
baudios



PROCESAR COMANDOS POR EL PUERTO SERIE

```
void loop() {  
    if (Serial.available() > 0) {  
        char command = (char) Serial.read();  
        switch (command) {  
            case 'H':  
                Serial.println("Hola Mundo");  
                break;  
            case 'D':  
                Serial.println(distancia(TriggerPin, EchoPin));  
                break;  
        }  
    }  
    Serial.flush();  
}
```



ARDUINO COMO ESCLAVO A TRAVÉS DE UART

```
public class ArduinoUart {  
    private UartDevice uart;  
    public ArduinoUart(String nombre, int baudios) {  
        try {  
            uart = PeripheralManager.getInstance().openUartDevice(nombre);  
            uart.setBaudrate(baudios);  
            uart.setDataSize(8);  
            uart.setParity(UartDevice.PARITY_NONE);  
            uart.setStopBits(1);  
        } catch (IOException e) {  
            Log.w(TAG, "Error iniciando UART", e);  
        }  
    }  
    public void escribir(String s) {  
        try {  
            int escritos = uart.write(s.getBytes(), s.length());  
            Log.d(TAG, escritos+" bytes escritos en UART");  
        } catch (IOException e) {  
            Log.w(TAG, "Error al escribir en UART", e);  
        }  
    }  
}
```



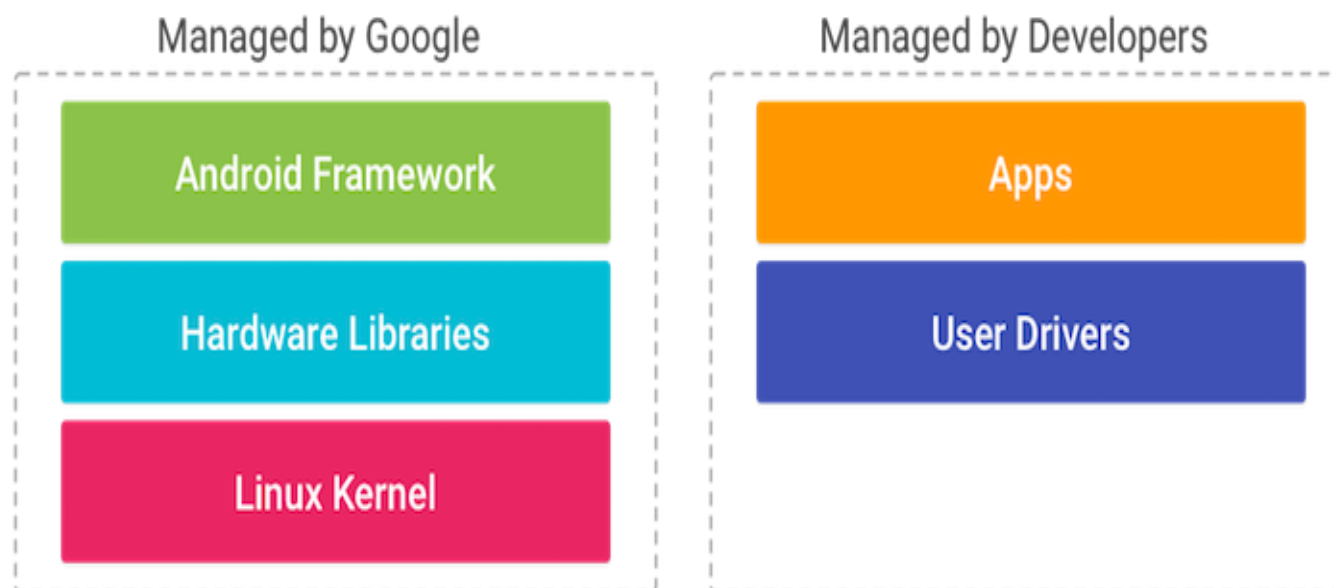
ARDUINO COMO ESCLAVO A TRAVÉS DE UART

```
public String leer() {  
    String s = "";  
    int len;  
    final int maxCount = 8; // Máximo de datos leídos cada vez  
    byte[] buffer = new byte[maxCount];  
    try {  
        do {  
            len = uart.read(buffer, buffer.length);  
            for (int i=0; i<len; i++) {  
                s += (char)buffer[i];  
            }  
        } while(len>0);  
    } catch (IOException e) {  
        Log.w(TAG, "Error al leer de UART", e);  
    }  
    return s;  
}  
  
public void cerrar() {  
    ...  
}
```



8 - CONTROLADORES DE USUARIO

- Interactuar con el chip PCF8591 ha sido complejo.
- Utilizar un driver simplifica mucho este trabajo.
- Además suelen estar bien escritos.
- Android Things introduce los controladores de usuario.
- Permiten interactuar con el sistema, generando eventos de teclado, posicionamiento,...



CONTROLADORES DE USUARIO

- Permitir que las aplicaciones interaccionen de forma directa con el framework
- Problemas:
 - puede hacer al sistema inestable
 - inseguro.
- Beneficios:
 - Muy necesarios en el marco de IoT, donde los tipos de dispositivos y periféricos es inmenso.
 - Esta flexibilidad nos va a permitir que nuestro código sea portable, reutilizable y fácil de integrar.



UTILIZAR CONTROLADORES (I)

1. Busca el driver para tu dispositivo

<https://github.com/androidthings/drivers-samples>

2. Añade la dependencia

```
dependencies {  
    ...  
    compile 'com.google.android.things.contrib:driver-button:1.0'  
}
```

3. Pide el permiso:

```
<uses-permission android:name=  
    "com.google.android.things.permission.MANAGE_INPUT_DRIVERS"/>
```



UTILIZAR CONTROLADORES (II)

4. Inicializa controlador

```
private ButtonInputDriver driverBoton;  
  
@Override protected void onCreate(Bundle savedInstanceState){  
    ...  
    try {  
        driverBoton = new ButtonInputDriver(  
            NOMBRE_DE_PIN,  
            Button.LogicState.PRESSED_WHEN_LOW,  
            KeyEvent.KEYCODE_SPACE);  
    } catch (IOException e) {  
        Log.e(TAG, "Error configurando pin GPIO", e);  
    }  
}
```



UTILIZAR CONTROLADORES (III)

4. Recuerda cerrar el recurso

```
@Override protected void onDestroy(){  
    super.onDestroy();  
    if (driverBoton != null) {  
        try {  
            driverBoton.close();  
        } catch (IOException e) { ... }  
    }  
}
```

5. Activarlo y desactivarlo cuando se necesario

```
@Override protected void onStart() {  
    super.onStart();  
    driverBoton.register();  
}  
  
@Override protected void onStop() {  
    super.onStop();  
    driverBoton.unregister();  
}
```



UTILIZAR CONTROLADORES (IV)

6. Utiliza el API estándar de Android:

```
@Override public boolean onKeyDown(int keyCode, KeyEvent event) {  
    if (keyCode == KeyEvent.KEYCODE_SPACE) {  
        // ...  
        return true;  
    }  
    return super.onKeyDown(keyCode, event);  
}  
  
@Override public boolean onKeyUp(int keyCode, KeyEvent event) {  
    if (keyCode == KeyEvent.KEYCODE_SPACE) {  
        // ...  
        return true;  
    }  
    return super.onKeyUp(keyCode, event);  
}
```



TIPOS DE CONTROLADORES

- **Dispositivos de Interfaz Humana (HID):**
información proporcionada por el usuario. teclados, almohadillas táctiles y mando.
permiso: `MANAGE_INPUT_DRIVERS`. [Más info](#)
- **Ubicación:**
Ubicación física del dispositivo.
permiso: `MANAGE_GNSS_DRIVERS` [Más info](#)
- **Sensores:**
miden condiciones del entorno físico. permiso:
`MANAGE_SENSOR_DRIVERS`. [Más info](#)
- **LoWPAN:** redes inalámbricas de área personal de baja potencia, módulo de radio externo al dispositivo para crear redes en malla.
permiso: `MANAGE_LOWPAN_INTERFACE`. [Más info](#)



CONTROLADORES DE INTERFAZ DE USUARIO (I)

```
public class ButtonDriverService extends Service {  
    private static final String DRIVER_NAME = "EscapeButton";  
    private static final int KEY_CODE = KeyEvent.KEYCODE_ESCAPE;  
    private InputDriver driver;  
  
    @Override public void onCreate() {  
        super.onCreate();  
        driver = new InputDriver.Builder()  
            .setName(DRIVER_NAME)  
            .setSupportedKeys(new int[] {KEY_CODE})  
            .build();  
        UserDriverManager manager = UserDriverManager.getInstance();  
        manager.registerInputDriver(driver);  
    }  
  
    @Override public IBinder onBind(Intent intent) {  
        return null;  
    }  
}
```



CONTROLADORES DE INTERFAZ DE USUARIO (II)

○ Lanza el evento:

```
private void lanzarEvento(boolean pressed) {  
    InputDriverEvent event = new InputDriverEvent();  
    event.setKeyPressed(KEY_CODE, pressed);  
    driver.emit(event);  
}
```

○ Desactiva el registro

```
@Override public void onDestroy() {  
    super.onDestroy();  
    UserDriverManager.getInstance().unregisterInputDriver(driver);  
}
```



CONTROLADORES HID - MOVIMIENTO

○ Inicializa:

```
driver = new InputDriver.Builder()  
    .setName(DRIVER_NAME)  
    .setAxisConfiguration(MotionEvent.AXIS X, 0, 255, 0, 0)  
    .setAxisConfiguration(MotionEvent.AXIS Y, 0, 255, 0, 0)  
    .build();
```

○ Lanza el evento:

```
private void lanzarEvento(int x, int y, boolean pressed) {  
    InputDriverEvent event = new InputDriverEvent();  
    event.setPosition(MotionEvent.AXIS X, x);  
    event.setPosition(MotionEvent.AXIS Y, y);  
    event.setContact(pressed);  
    driver.emit(event);  
}
```



CONTROLADORES DE UBICACIÓN

- Pide el permiso:

```
<uses-permission android:name=  
    "com.google.android.things.permission.MANAGE_GNSS_DRIVERS" />
```

- Inicializa:

```
driver = new GnssDriver();
```

- Lanza el evento:

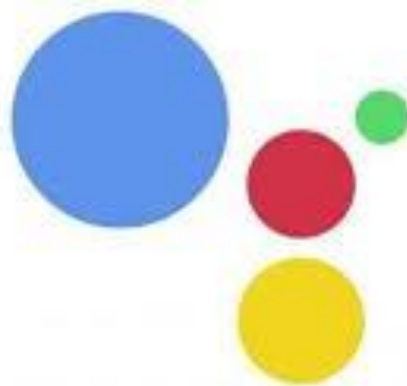
```
private void lanzarEvento(DatosDePosicion datos) {  
    Location location = obtenLocalización(datos);  
    driver.reportLocation(location);  
}
```



9 - INTEGRAR GOOGLE ASSISTANT SDK



UNIVERSIDAD
POLITECNICA
DE VALENCIA

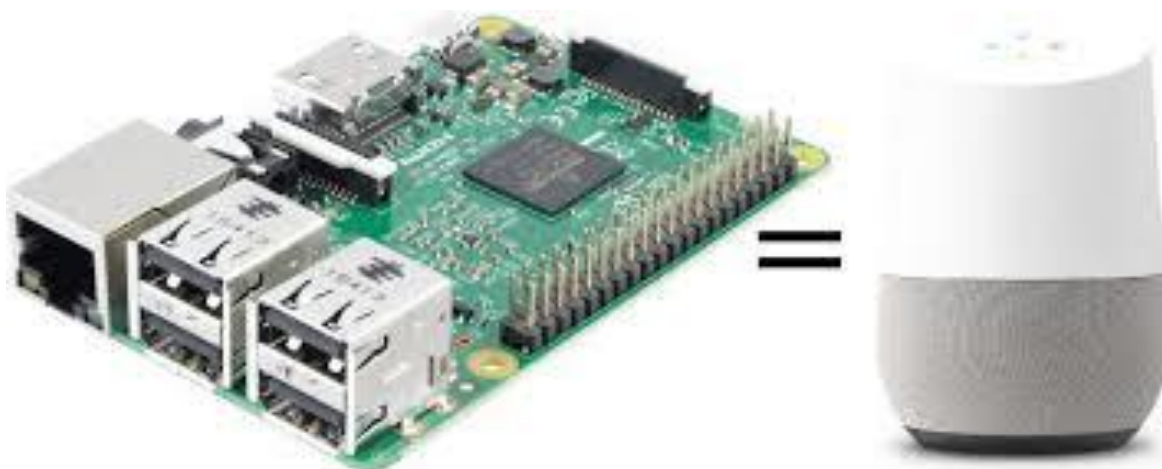


Google Assistant SDK



GOOGLE ASSISTANT

- Google ha apostado muy fuerte por el control de dispositivos IoT por medio de la voz.
- Es la forma natural de comunicarnos.
- En un entorno como el hogar puede ser el lugar más adecuado.
- *Google Assistant SDK for devices* nos va a permitir que cualquier dispositivo empotrado tenga la misma funcionalidad que Google Home.



OPCIONES DEL SDK:

○ Google Assistant Library

- biblioteca en Python, compatible con arquitecturas linux-armv7l y linux-x86_64.
- API de alto nivel basada en eventos
- Características lista para usar:
 - Activación manos libres: con Hey Google o Ok Google
 - Captura y reproducción de audio
 - Gestión del estado de conversación
 - Gestión de temporizador y alarma

○ Servicio Asistente de Google

- Más flexible y con más amplio soporte.
- API de bajo nivel que manipula directamente audio.
- basado en gRPC, cualquier plataforma
- No admite las características anteriores.



gRPC



UNIVERSIDAD
POLITECNICA
DE VALENCIA

- Sistema de llamadas a procedimiento remoto (RPC)
- código abierto
- desarrollado por Google.
- alternativa a REST
- Usa HTTP/2 en lugar de HTTP 1.1
- Usa protobuf en lugar de JSON



ACCIONES EN GOOGLE

- Agregar funcionalidad única a nuestro dispositivo:

Nombre	Dispositivos	Descripción	Frases
<u>OnOff</u>	cualquiera	conectar / desconectar el dispositivo	Turn on. Turn off.
<u>StartStop</u>	cualquiera	arranque / parada general	Start the washing machine.
<u>Modes</u>	cualquiera	dispositivos que pueden trabajar en diferentes modos.	What mode is the dryer in? Set the dryer to delicate.
<u>Toggles</u>	cualquiera	cambia botones físicos, asociados a una función.	Is my dryer sterilization on? Turn on sterilization for the dryer.



ACCIONES EN GOOGLE

Nombre	Dispositivos	Descripción	Frases
<u>Brightness</u>	<u>Light</u>	nivel de brillo de 0 a 100	Adjust my light to 65% brightness.
<u>FanSpeed</u>	<u>Air conditioning unit</u> , <u>Air purifier</u> , <u>Fan</u>	velocidad del ventilador	What speed are the fans in the living room?
<u>TemperatureSetting</u>	<u>Air conditioning</u> , <u>Thermostat</u>	temperatura de la habitación	Initialize Thermostat setting.
<u>RunCycle</u>	cualquiera	dispositivos vasados en ciclos de trabajo	What is the washing machine doing?
<u>Locator</u>	<u>Vacuum</u> , otros móviles	para preguntar ubicación del dispositivo.	Where are my keys?
...			

- Estas acciones han sido definidas para en, de, fr, ja y es.



DIÁLOGO BÁSICO CON EL ASISTENTE (PASOS)

1. Implementa cliente gRPC de transmisión de audio.
2. Espera a que el usuario active solicitud (botón).
3. Envía un AssistRequest de tipo **config** con los campos:
 - audio_in_config: formato del audio que se va a enviar. (ver AudioInConfig)
 - audio_out_config: formato deseado para el audio recibido (AudioOutConfig).
 - device_config: dispositivo registrado en el Asistente (ver DeviceConfig).
 - dialog_state_in: estado del dialogo, ej. idioma, localización (DialogStateIn).
4. Empieza a grabar.
5. Envía mensajes AssistRequest con audio de consulta



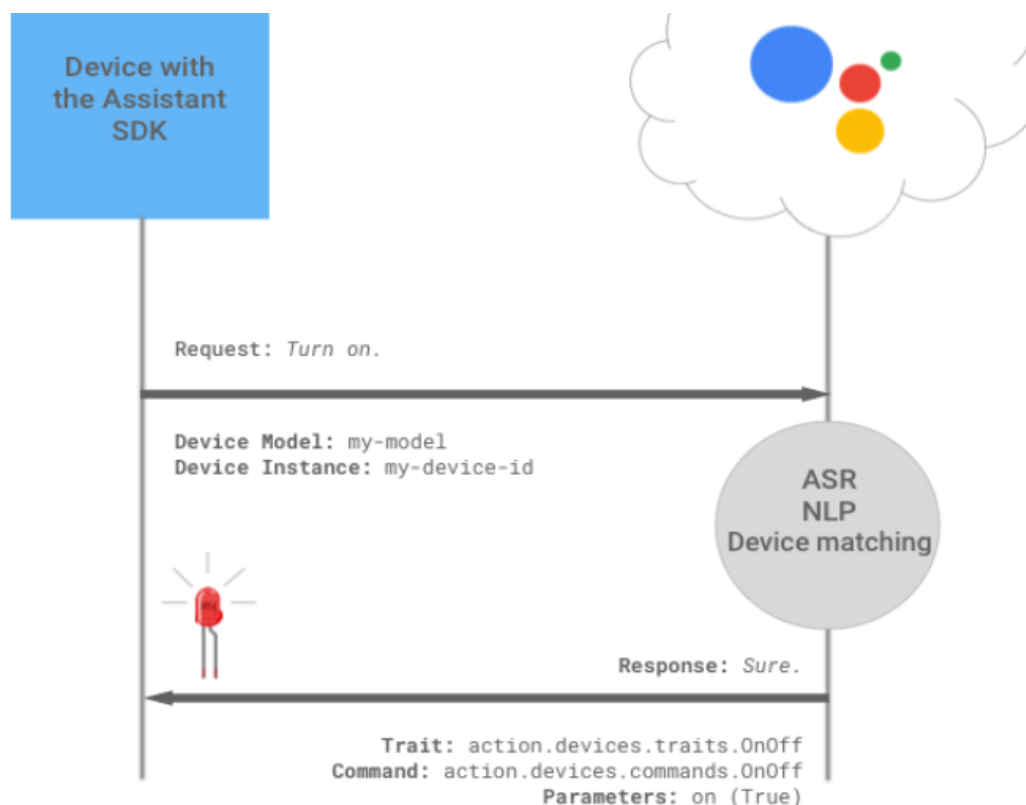
DIÁLOGO BÁSICO CON EL ASISTENTE (II)

6. Recoge los mensajes AssistResponse entrantes.
7. Extrae los metadatos del mensaje. Por ejemplo, conversation_state, si se quiere cambiar el volumen o texto suplementario para visualizar en una pantalla (ver DialogStateOut).
8. Detén la grabación cuando reciba un AssistResponse con un event_type igual a END_OF_UTTERANCE.
9. Reproduce el audio de la respuesta del Asistente.
10. Toma el conversation_state que extrajiste antes y cópialo en el DialogStateIn del mensaje en siguiente AssistRequest.



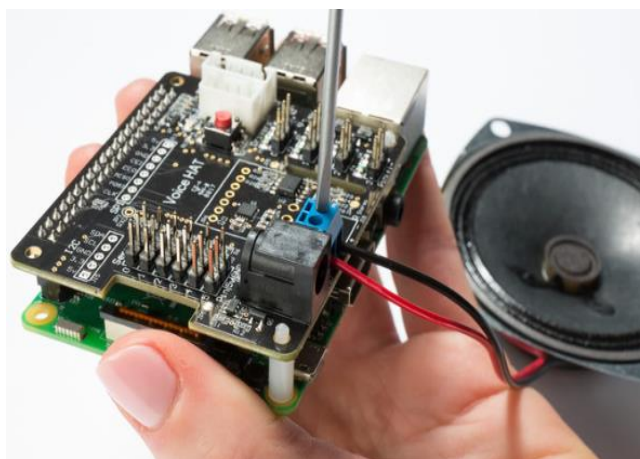
DIÁLOGO BÁSICO CON EL ASISTENTE (III)

- Si acciones específicas para tu dispositivo:
 - En mensaje entrante, extrae el `device_action` ([DeviceAction](#)).
 - Analiza en JSON las diferentes características ([traits](#))



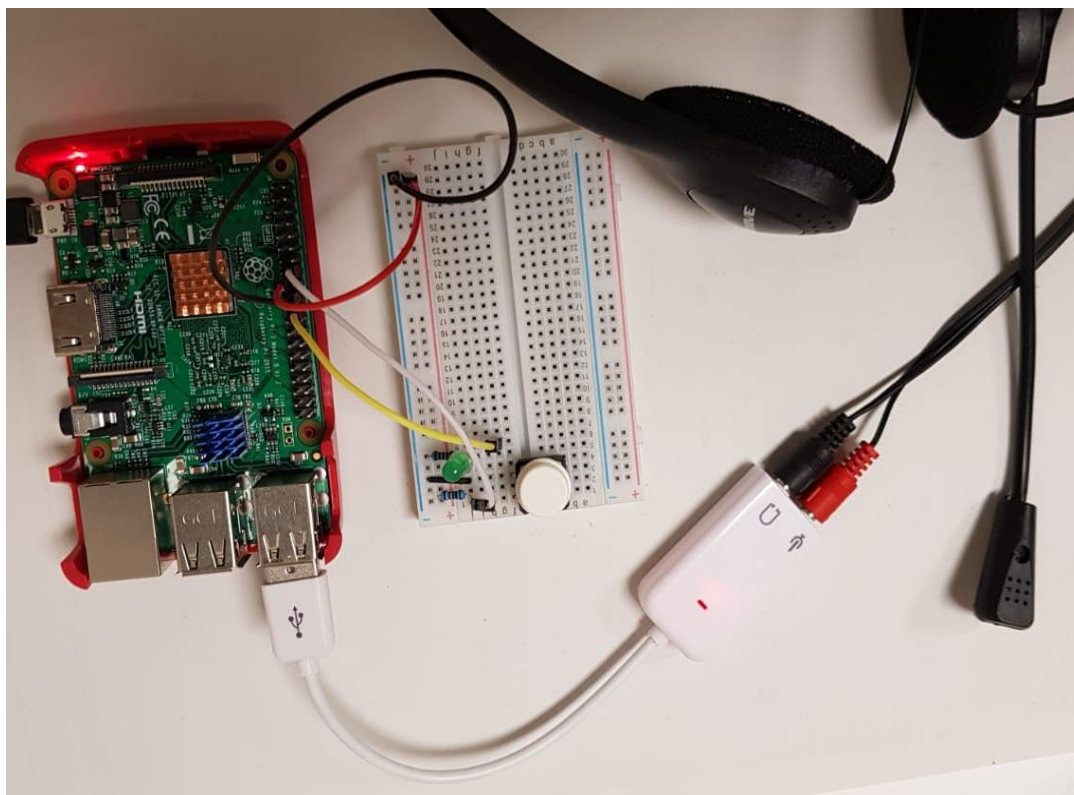
TUTORIAL BASADO EN UN CODELAB

- <https://codelabs.developers.google.com/codelabs/androidthings-assistant>
- Otra alternativa es usar el Voice Kit.
 - Raspberry Pi Zero W, altavoz, pulsador, caja y VoiceHat.
 - 30€



MONTAJE

- Material necesario:

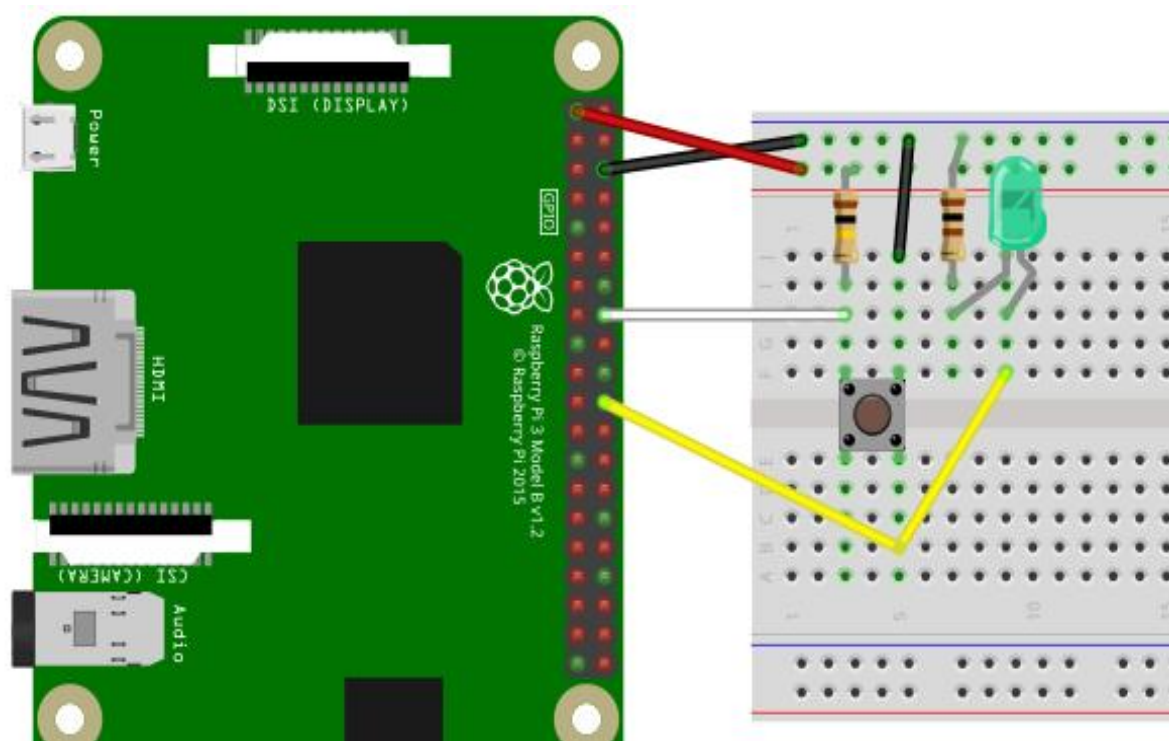


- Otra alternativa es usar micrófono por Bluetooth:



MONTAJE

○ Cableado:



CONFIGURAR CREDENCIALES

- Controles de actividad de tu cuenta Google
<https://myaccount.google.com/activitycontrols>



Actividad en la Web y en Aplicaciones



Información del dispositivo



Actividad de Voz y Audio



CONSOLA DE ACCIONES

- <https://console.actions.google.com>
- Crea un nuevo proyecto y registra un modelo

Register model

1 — 2 — 3

Create model Download credentials file Specify traits

Product name ?

Producto

Manufacturer name ?

UPV

Device type ?

Light

Device Model id ?

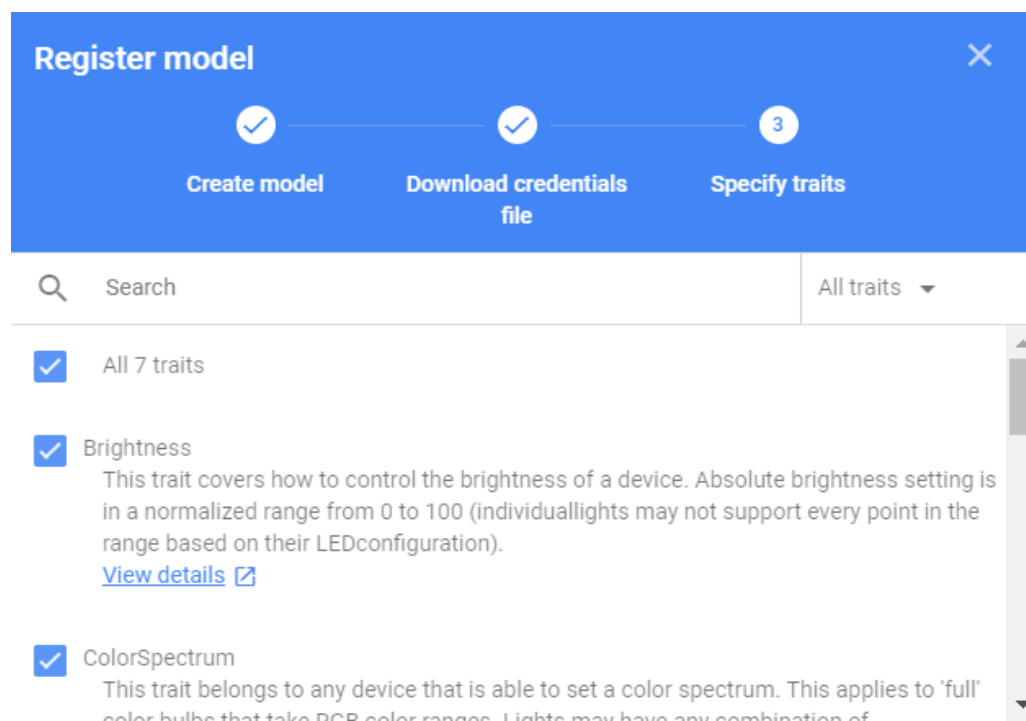
asistente-39d8f-producto-hfkhdv

- paso 2: descarga *credentials.json* (en raíz proyecto)




CONSOLA DE ACCIONES

- paso 3: Selecciona características (traits) para dispositivo tipo Ligth.



The screenshot shows a 'Register model' dialog box with a blue header and a close button (X). Below the header is a progress bar with three steps: 'Create model' (checked), 'Download credentials file' (checked), and 'Specify traits' (active, indicated by a '3' in a circle). Below the progress bar is a search bar with a magnifying glass icon and the text 'Search'. To the right of the search bar is a dropdown menu labeled 'All traits'. Below the search bar is a list of traits, each with a checked checkbox and a description:

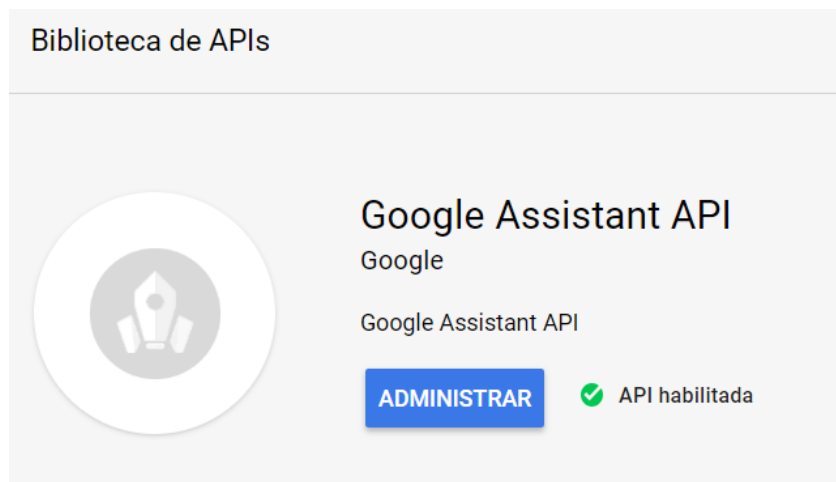
- ☒ All 7 traits
- ☒ Brightness
This trait covers how to control the brightness of a device. Absolute brightness setting is in a normalized range from 0 to 100 (individual lights may not support every point in the range based on their LED configuration).
[View details](#) 
- ☒ ColorSpectrum
This trait belongs to any device that is able to set a color spectrum. This applies to 'full' color bulbs that take RGB color ranges. Lights may have any combination of

- Selecciona idiomas: es, en



CONSOLA GOOGLE APIS

- Habilita Google Assistant API en consola Google APIs



- Desde esta consola puedes consultar las cuotas:

Nombre de cuota	Límite
(Unlabeled quota) por día	500
(Unlabeled quota) por minuto por usuario	60



HERRAMIENTA PYTHON3 PARA CREDENCIALES

- Instala python3.
- Instala librerías:

```
pip install --upgrade pip setuptools wheel  
pip install --upgrade google-auth-oauthlib[tool]
```

- Desde raíz proyecto ejecuta:

```
google-oauthlib-tool --client-secrets credentials.json  
--credentials shared/src/main/res/raw/credentials.json  
--scope https://www.googleapis.com/auth/assistant-sdk-  
prototype --save
```

- Se abrirá un navegador para autorizar
- Se crea un nuevo credentials.json en res/raw/









PRIMERA VERSIÓN DEL ASISTENTE

○ Instala desde GitHub

<https://github.com/googlecodelabs/androidthings-googleassistant.git>

○ Cinco módulos

- Común
- Asistente básico
- Control de volumen
- Acciones predefinidas
- Nuevas acciones

- >  **shared**
- >  **step1-start-here**
- >  **step2-volume-control**
- >  **step3-builtin-device-actions**
- >  **step4-custom-device-actions**
- >  **Gradle Scripts**



MODULO COMUN

- MyDevice.class

```
public class MyDevice {  
    public static final String MODEL_ID = "asistente-39d8f-prod-d91";  
    public static final String INSTANCE_ID = "id_unica_dispositivo";  
    public static final String LANGUAGE_CODE = "es-ES"; // "en-US"  
}
```

- BroadDefaults.class - define los puertos GPIO
- Credentials.class - verifica credenciales



STEP1 - COMUNICACIÓN SIMPLE CON ASISTENTE

- Se pueden seguir los 10 pasos de dialogo con el asistente
- Arranque y parada de forma manual

```
@Override public void onButtonEvent(Button button, boolean pressed) {  
    ...  
    mLed.setValue(pressed);  
    if (pressed) mAssistantHandler.post(mStartAssistantRequest);  
    else         mAssistantHandler.post(mStopAssistantRequest);  
}
```

- Podrás hacer consultas sobre el tiempo, pedir un chiste,...



STEP2 – CONTROL DE VOLUMEN

- “pon el volumen a 2” o “sube el volumen”
- Método llamado tras cada respuesta

```
@Override public void onNext(AssistResponse value) {
    ...
    if (value.getDialogStateOut() != null) {
        int volume = value.getDialogStateOut().getVolumePercentage();
        if (volume > 0) {
            mVolumePercentage = volume;
            Log.i(TAG, "assistant volume changed: " +
                mVolumePercentage);
            mAudioTrack.setVolume(AudioTrack.getMaxVolume() *
                mVolumePercentage / 100.0f);
        }
        mConversationState= vaue.getDialogStateOut()
                                .getConversationState();
    }
}
```

- El asistente cambia el volumen
 - En algunas tarjetas no tiene efecto



STEP3 – ACCIONES PREDEFINIDAS

- Podemos asociar a nuestro dispositivo una serie de características (traits)
- Podrán ser modificados por medio de acciones
- ¿Posible error? : modificar características en modelo no funciona. Crea un nuevo modelo.
- Tras decir “turn on” / “encender”

```
{  
  "requestId": "ff36a3cc-ec34-11e6-b1a0-64510650abcf",  
  "inputs": [{  
    "intent": "action.devices.EXECUTE",  
    "payload": {  
      "commands": [{  
        "devices": [{ "id": "123" }],  
        "execution": [{  
          "command": "action.devices.commands.OnOff",  
          "params": { "on": true }  
        }]  
      }  
    }  
  }  
}]
```



STEP4 - DEFINIR ACCIONES PERSONALIZADAS

```
{
  "manifest": {
    "displayName": "Blinky light",
    "invocationName": "Blinky light",
    "category": "PRODUCTIVITY" },
  "actions": [
    {
      "name": "com.example.actions.BlinkLight",
      "availability": {
        "deviceClasses": [ {"assistantSdkDevice": {} } ] },
      "intent": {
        "name": "com.example.intents.BlinkLight",
        "parameters": [
          { "name": "number",
            "type": "SchemaOrg_Number" },
          { "name": "speed",
            "type": "Speed" } ],
        "trigger": {
          "queryPatterns": [
            "parpadea ($Speed:speed)? $SchemaOrg_Number:number veces",
            "parpadea $SchemaOrg_Number:number veces ($Speed:speed)?" ] }
      }
    }
  ]
}
```

...



STEP4 - DEFINIR ACCIONES PERSONALIZADAS

...

```
"fulfillment": {
  "staticFulfillment": {
    "templatedResponse": {
      "items": [
        {"simpleResponse": {
          "textToSpeech": "Parpadeando $speed.raw $number veces"
        } },
        {"deviceExecution": {
          "command": "com.example.commands.BlinkLight",
          "params": {
            "speed": "$speed",
            "number": "$number"
          }
        }
      ]
    }
  }
}, ...
```



STEP4 - DEFINIR ACCIONES PERSONALIZADAS

...

```
"types": [  
  { "name": "$Speed",  
    "entities": [  
      { "key": "lentamente",  
        "synonyms": [ "lento" ] },  
      { "key": "normal",  
        "synonyms": [ "regular" ] },  
      { "key": "rápidamente",  
        "synonyms": [ "rápido" ] }  
    ]  
  }  
]
```



ANDROID THINGS



UNIVERSIDAD
POLITECNICA
DE VALENCIA

- ¿Alguna pregunta?

