

Explorando el internet de las cosas

Por Haroldo López
Devs+502

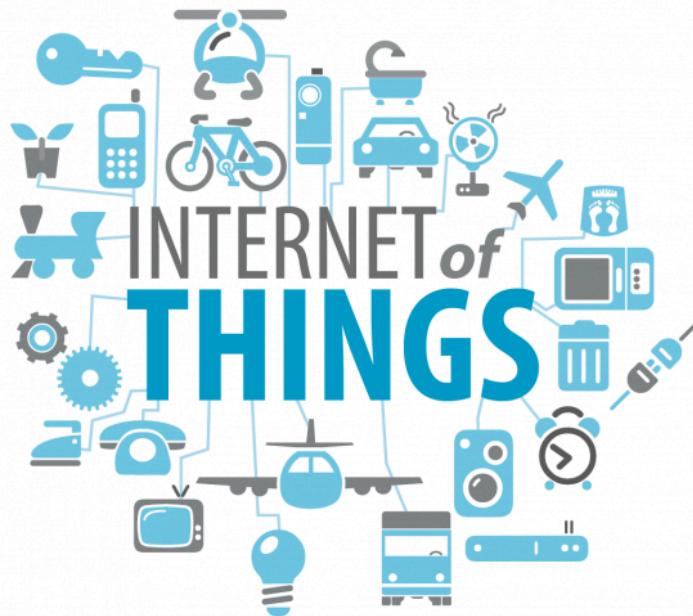
Temas a desarrollar

- ¿Qué es el internet de las cosas o IoT?
- Principios de redes de computadoras
- Android Things el sistema operativo de Google para IoT
- La raspberry Pi y otros hardwares para IoT
- Arduino y sus alternativas
- Comunicando dispositivos con NodeMcu

1

¿Qué es el internet de las cosas?

Conectando el mundo

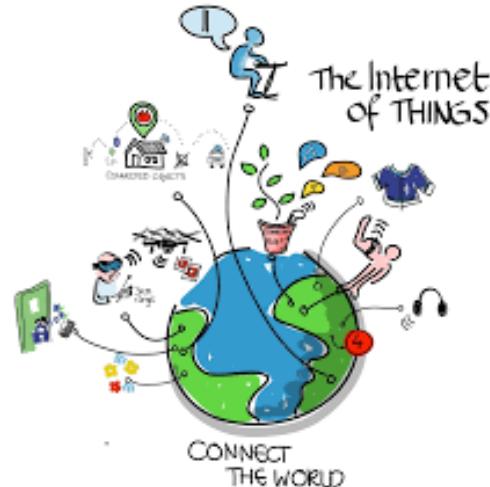


El internet de las cosas es la interconexión digital de todo tipo de objetos cotidianos a través de una red.

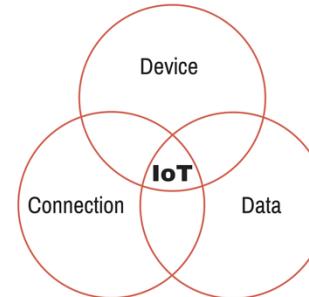
¿Cómo nació IoT?

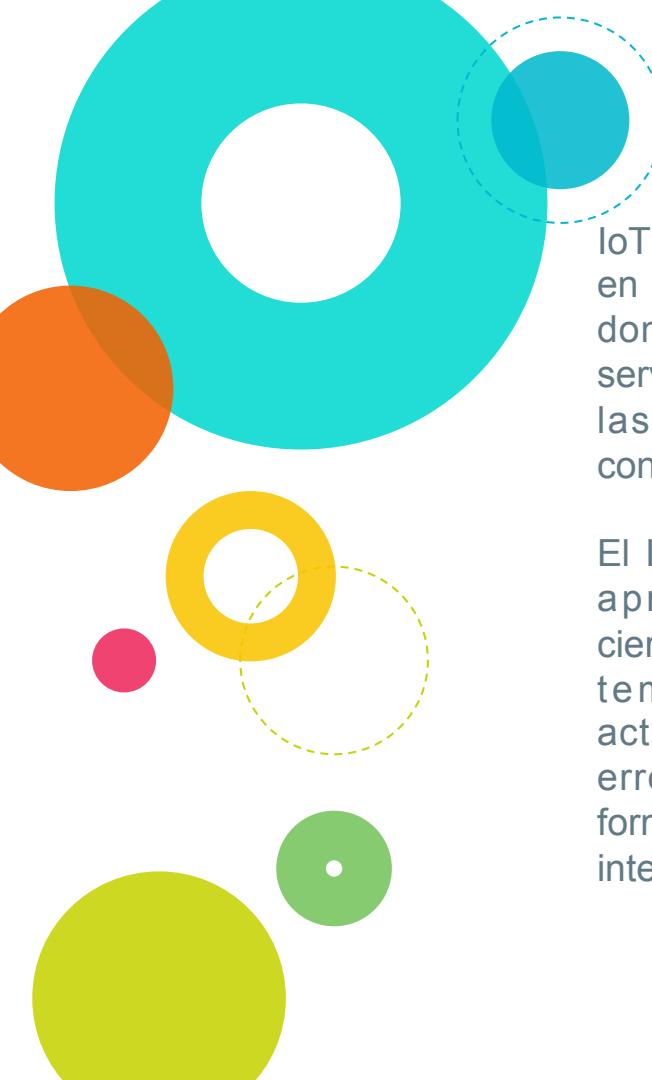
Este fue el título de una conferencia propuesta por Kevin Ashton en el año 1999.

Esta conferencia se realizó para exponer investigaciones en el campo de los sensores y la identificación por radiofrecuencia en red (RFID) del Instituto Tecnológico de Massachusetts.



1. Asignarle un identificador de red a cada dispositivo.
2. Los dispositivos serán capaces de enviar y recibir información de todo tipo.
3. La información es procesada en la nube donde se interpretarán las grandes cantidades de datos generadas.
4. La información procesada se compartirá tiempo real para un uso específico.





IoT tiene grandes avances en el área de los hogares, donde electrodomésticos, servicios o gadgets como las bombillas ya están conectados a Internet.

El Internet de las Cosas se aprovecha para medir ciertos parámetros como la temperatura, energía, actividad, luz, humedad, errores, entre otros, de forma automática y sin la interacción del humano.



Temperatura



Humedad



Presión



Velocidad del aire

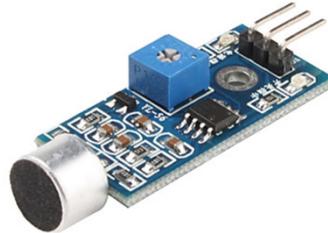
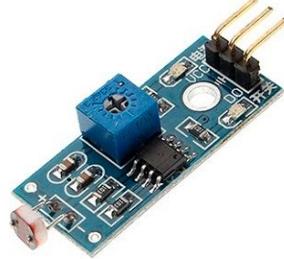
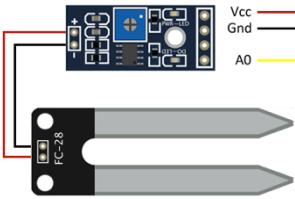


Fotometría-radiometría



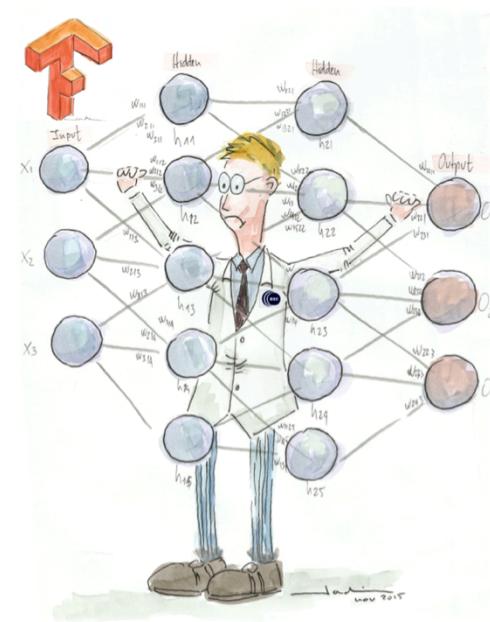
Acústica

Algunos sensores



Machine Learning

- Todo surge por el impulso tecnológico al que suele referir como Big Data.
- Una de las claves de la inteligencia artificial avanzada está en el aprendizaje.
- El Machine Learning se refiere a la disciplina del Aprendizaje Automático. Para experimentar con estos servicios tenemos plataformas como IBM Watson Developer Cloud, Amazon Machine Learning, Azure Machine Learning, TensorFlow o BigML.



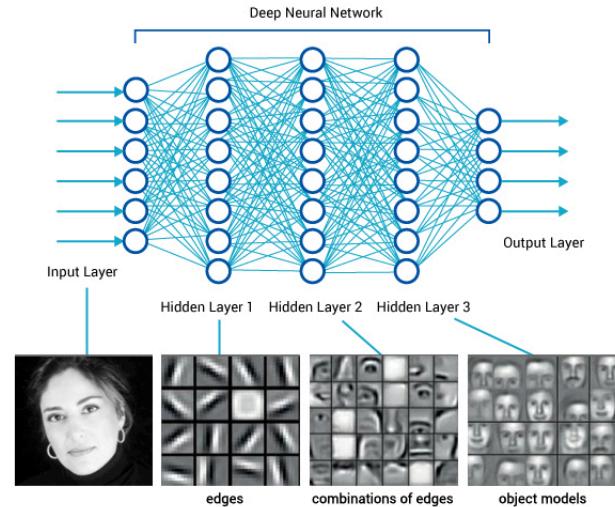
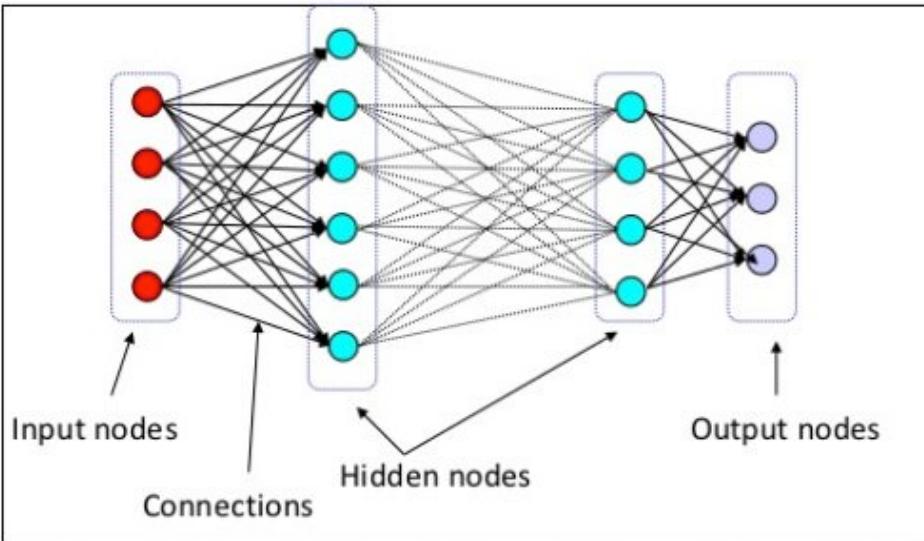
Deep Learning: la aproximación a la percepción humana

En el enfoque Deep Learning se usan estructuras lógicas que se asemejan en mayor medida a la organización del sistema nervioso de los mamíferos, teniendo capas de unidades de proceso (neuronas artificiales) que se especializan en detectar determinadas características existentes en los objetos percibidos.

La visión artificial es una de las áreas donde el Deep Learning proporciona una mejora considerable en comparación con algoritmos más tradicionales.

Los modelos computacionales de Deep Learning imitan estas características arquitecturales del sistema nervioso, permitiendo que dentro del sistema global haya redes de unidades de proceso que se especialicen en la detección de determinadas características ocultas en los datos.

Deep Learning: la aproximación a la percepción humana



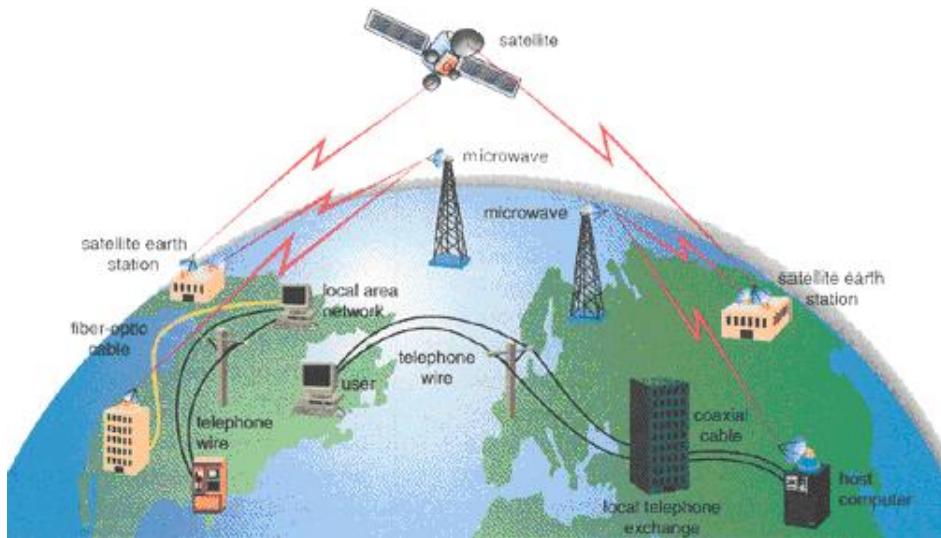
2

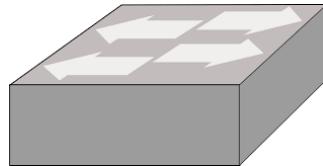
Principios de redes de computadoras

Todo tendrá una dirección de red

¿Qué es una red?

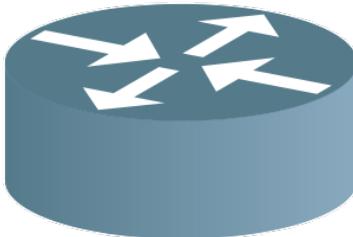
De una manera sencilla simplemente una red es un camino o carretera destinado a interconectar usuarios, dispositivos y aplicaciones.





Todos los dispositivos se conectan a un switch el cual posibilita la comunicación entre dispositivos pertenecientes a una misma LAN, utilizando para la identificación de cada dispositivo una dirección embebida conocida como **direcciones MAC**.

Para conectar muchas redes es necesario la utilización de un dispositivo llamado router capaz de encontrar una ruta y que sirva a los demás dispositivos como una puerta de enlace. El router se vale de las direcciones proporcionadas por el protocolo de internet o **direcciones IP** (internet protocol)



Sistema Operativo

	Funcionamiento	Dispositivo	Encapsulamiento	Troubleshooting (Host)
07 Aplicación	Es la capa mas familiar al usuario final Provee de acceso a la red a las aplicaciones			El problema esta en la aplicación o usuario.
06 Presentación	Convierte la información en un formato "Genérico" Presenta servicios de Encriptación / Desencriptación		Datos	- No se puede establecer sesión ?
05 Sesión	Comienza y finaliza sesiones Mantiene las sesiones separadas lógicamente			
04 Transporte	Elije el tipo de transmisión Confiable (TCP) / No Confiable (UDP) Números de Puerto		Segmentos	- Puertos en conflicto/filtrados/cerrados ?
03 Red	Provee direccionamiento "Lógico" (Direcciones IP) Encuentra la mejor ruta hacia un destino		Paquetes	- IP, máscara de sub red y gateway configurados correctamente ?
02 Enlace	Provee direccionamiento "Físico" (Direcciones MAC) Asegura que los datos no tengan errores		Tramas	- La NIC esta funcionando ? probar ping 127.0.0.1
01 Física	Transmisión de datos en bits (Electricidad o Luz)	011100111	Bits	- Los cables están bien conectados ? - Los led de la NIC están prendidos ?

3

Android Things El sistema operativo de IoT en Google

Android Things

- Es la plataforma para el Internet de las Cosas de Google diseñada para que los desarrolladores puedan sacar un mayor provecho de los productos de casa.
- Android Things es una actualización y un cambio de marca para Brillo, un sistema operativo basado en Android para dispositivos inteligentes y productos de Internet de las Cosas.
- Puede funcionar con productos como altavoces, cámaras de seguridad y routers.
- A diferencia de Brillo, el desarrollo de Android Things se puede lograr con las mismas herramientas para desarrolladores de serie Android.

Things Support Library

Peripheral I/O API

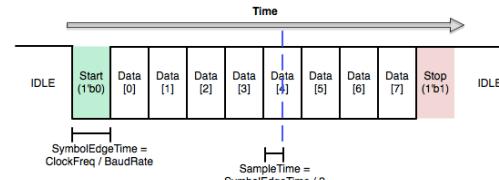
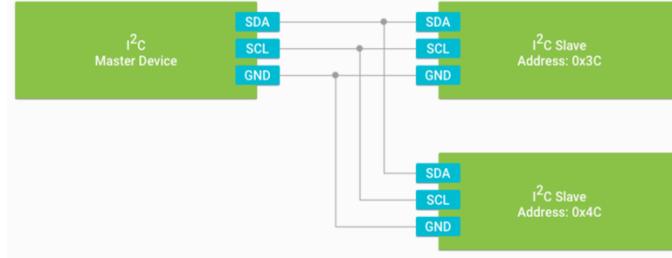
Es la que se encarga de manejar las entradas y salidas de los periféricos. Permite que las aplicaciones se comuniquen con sensores y actuadores a través de protocolos e interfaces estándar de la industria. Como lo son:

GPIO, PWM, I2C, SPI, UART.

User Driver API

Los controladores de usuario amplían los actuales servicios de framework de Android y permiten a las aplicaciones injectar eventos de hardware en el framework al que otras aplicaciones pueden acceder mediante las API estándar de Android.

Raspberry Pi 3 GPIO Header	
Pin#	NAME
01	3.3v DC Power
03	GPIO_2 (SDA1, I ² C)
05	GPIO_3 (SCL1, I ² C)
07	GPIO_4 (GPIO_GCLK)
09	Ground
11	GPIO_17 (GPIO_GEN0)
13	GPIO_27 (GPIO_GEN0)
15	GPIO_22 (GPIO_GEN3)
17	3.3v DC Power
19	GPIO_10 (SPI_MOSI)
21	GPIO_9 (SPI_MISO)
23	GPIO_11 (SPI_CLK)
25	Ground
27	ID_SD / PC_ID EEPROM
29	GPIO_5
31	GPIO_6
33	GPIO_13
35	GPIO_19
37	GPIO_26
39	Ground
40	GPIO_21



Peripheral Input/Output

GPIO

General Purpose Input/Output (GPIO) pins son usados para la comunicación digital con componentes.

Para obtener los nombres de los pines utilizando el PeripheralManagerService llamando a la función `getGpioList()`. En una Raspberry Pi retornará la siguiente lista:

[BCM12, BCM13, BCM16, BCM17, BCM18, BCM19, BCM20, BCM21, BCM22, BCM23, BCM24, BCM25, BCM26, BCM27, BCM4, BCM5, BCM6]

Una vez que se tenga el nombre del pin al que va a leer o escribir, puede obtener una referencia de objeto Gpio a ese pin llamando a `openGpio(String pin_name)` desde su PeripheralManagerService.

```
try {
    mGpio = service.openGpio(PIN_NAME);
} catch (IOException e){
}
```

Peripheral Input/Output

Los GPIO pueden ser entradas o salidas. Es por esto que se necesita configurar la dirección del pin **DIRECTION_IN** y establecer el tipo de trigger.

```
mGpio.setDirection(Gpio.DIRECTION_IN);  
mGpio.setEdgeTriggerType(Gpio.EDGE_BOTH);
```

Los tipos de trigger son **EDGE_NONE**, **EDGE_RISING**, **EDGE_FALLING**, y **EDGE_BOTH**. Ahora que su GPIO está escuchando los triggers, tendrá que crear un GpioCallback.

```
private GpioCallback mCallback = new GpioCallback() {  
    @Override  
    public boolean onGpioEdge(Gpio gpio) {  
        try {  
            Log.d("Tuts+", "GPIO value: " + gpio.getValue());  
        } catch( IOException e ) {  
  
        }  
        return super.onGpioEdge(gpio);  
    }  
  
    @Override  
    public void onGpioError(Gpio gpio, int error) {  
        super.onGpioError(gpio, error);  
    }  
};
```

Debe registrar el callback con:

mGpio.registerGpioCallback(mCallback);

Peripheral Input/Output

Si su pin GPIO está escribiendo información, deberá configurar la dirección como DIRECTION_OUT_INITIALLY_LOW o DIRECTION_OUT_INITIALLY_HIGH, dependiendo de si desea que el componente se inicie como encendido o apagado. No se requieren tipos de disparo para un pin de salida:

```
mGpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW)
```

Para escribir un valor debe utilizar la función:

```
mGpio.setValue(true);
```

Para dejar de utilizar los pines debe utilizar:

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

PWM

Para en listar los PWM disponibles se debe crear un PeripheralManagerService y llamar a la función `getPwmList()`.

Luego debe abrir el PWM que desee utilizar de la siguiente manera:

```
try {
    mPwm = service.openPwm(PIN_NAME);
} catch( IOException e ) {
}
```

Y asignarle el duty cycle, la frecuencia y la habilitación de cada uno

```
mPwm.setPwmFrequencyHz(120);
mPwm.setPwmDutyCycle(25);
mPwm.setEnabled(true);
```

Si se desea cerrar la conexión debe utilizar

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

I²C

El bus Inter-Integrated Circuit (I²C) permite comunicarse con varios dispositivos a través de una conexión física y le permite enviar datos complejos con sólo unos pocos pines de un dispositivo a otro integrado.

- SCL: shared clock signal
- SDA: Shared Data Line
- GND: Tierra

```
PeripheralManagerService manager = new PeripheralManagerService();
List<String> deviceList = manager.getI2cBusList();

if( !deviceList.isEmpty() ) {
    Log.d( "Tuts+", deviceList.toString() );
}
```

Puede encontrar la dirección de software y otros detalles de conexión de bajo nivel en el datasheet del componente periférico

```
private I2cDevice mDevice;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try {
        PeripheralManagerService manager = new PeripheralManagerService();
        mDevice = manager.openI2cDevice(I2C_DEVICE_NAME, I2C_ADDRESS);
    } catch (IOException e) {}
}
```



I²C

Cuando se comunica con un dispositivo a través de I²C, puede utilizar el protocolo System Management Bus (SMB). Haciendo uso de los siguientes métodos:

- ReadRegByte () y writeRegByte (): lee o escribe un solo byte de un registro específico.
- ReadRegWord () y writeRegWord (): Leer o escribir bytes de dos registros secuenciales en un periférico.
- GND: Tierra

```
public void singleByte(I2cDevice device, int address) throws IOException {
    // Read one register from slave
    byte value = device.readRegByte(address);

    // Write the value back to slave
    device.writeRegByte(address, value);
}

public byte[] multipleBytes(I2cDevice device, int startAddress) throws IOException {
    // Read three consecutive register values
    byte[] data = new byte[3];
    device.readRegBuffer(startAddress, data, data.length);
    return data;
}
```

Para cerrar la conexión deberá utilizar el método:

```
@Override
protected void onDestroy() {
    super.onDestroy();

    if (mDevice != null) {
        try {
            mDevice.close();
            mDevice = null;
        } catch (IOException e) {}
    }
}
```

SPI

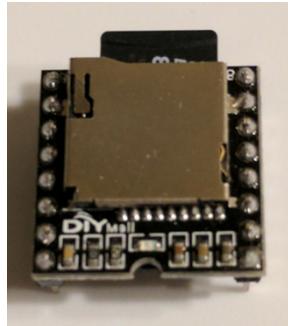
Serial Peripheral interface (SPI) funcionan de forma similar a las conexiones I2C, excepto que admiten la comunicación full-duplex. Las conexiones de esta comunicación son:

- ✓ Master Out Slave In (MOSI)
- ✓ Master In Slave Out (MISO)
- ✓ Clock signal (CLK)
- ✓ Shared ground (GND)

Se permite enlistar las conexiones spi utilizando PeripheralManagerService y llamando a getSpiBusList (), obteniendo como resultado en la raspberry Pi:
[SPI0.0, SPI0.1]

```
private SpiDevice mDevice;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try {
        PeripheralManagerService manager = new PeripheralManagerService();
        mDevice = manager.openSpiDevice(SPI_DEVICE_NAME);
    } catch (IOException e) {
    }
}
```



SPI

- SPI Mode: El modo SPI tiene dos componentes principales: si la señal de reloj es alta o baja mientras no se están transfiriendo datos y qué borde de un pulso se utiliza para transferir datos.
- Frecuencia: Representa la señal de reloj compartida en Hz, y muy probablemente será un valor determinado por las capacidades de sus dispositivos esclavos.
- Bit justification: Se utiliza para establecer la significancia de los datos. Por defecto es MSB.
- Bits per word (BPW): Configura cuántos bits se enviarán a un dispositivo esclavo antes de alternar la señal de selección de chip. De forma predeterminada, se enviarán ocho bits por palabra.

```
// Low clock, leading edge transfer
device.setMode(SpiDevice.MODE0);

device.setFrequency(16000000);
device.setBitsPerWord(8);
device.setBitJustification(false);
```

```
@Override
protected void onDestroy() {
    super.onDestroy();

    if (mDevice != null) {
        try {
            mDevice.close();
            mDevice = null;
        } catch (IOException e) {}
    }
}
```

```
//Half-duplex mode
public void sendCommand(SpiDevice device, byte[] buffer) throws IOException {
    // Shift data out to slave
    device.write(buffer, buffer.length);

    // Read the response
    byte[] response = new byte[32];
    device.read(response, response.length);
}

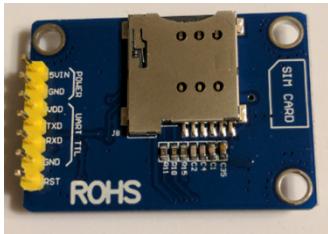
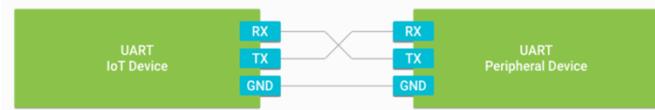
// Full-duplex mode
public void sendCommand(SpiDevice device, byte[] buffer) throws IOException {
    byte[] response = new byte[buffer.length];
    device.transfer(buffer, response, buffer.length);
}
```

UART

Los periféricos más complejos, como las pantallas LCD, los lectores de tarjetas SD y los módulos GPS, a menudo utilizan comunicación serie a través de puertos UART al comunicarse con un dispositivo maestro.

Esta comunicación tiene tres conexiones:

- RX: Recibir datos.
- TX: Transmitir datos.
- GND: Tierra.



Para listar los dispositivos debe utilizar:

```
PeripheralManagerService manager = new PeripheralManagerService();
List<String> deviceList = manager.getUartDeviceList();

if (!deviceList.isEmpty()) {
    Log.d("Tuts+", deviceList.toString());
}
```

Obtendrá como resultado, si es utiliza una raspberry Pi:
[UART0]

UART

Para abrir la conexión deberemos utilizar:

```
PeripheralManagerService manager = new PeripheralManagerService();
mDevice = manager.openUartDevice(UART_DEVICE_NAME);
```

Las configuraciones de la comunicación UART son:

```
uart.setBaudrate(9600);
uart.setDataSize(6);
uart.setParity(UartDevice.PARITY_NONE);
uart.setStopBits(2);
```

Configurar tipo de flujo de datos:

```
uartDevice.setHardwareFlowControl(UartDevice.HW_FLOW_CONTROL_AUTO_RTSCTS);
//or
uartDevice.setHardwareFlowControl(UartDevice.HW_FLOW_CONTROL_NONE);
```

Preparar el buffer para la comunciación:

```
uartDevice.write(buffer, buffer.length);
```

```
private UartDeviceCallback mUartCallback = new UartDeviceCallback() {
    @Override
    public boolean onUartDeviceDataAvailable(UartDevice uart) {
        byte[] buffer = new byte[MAX_BUFFER_SIZE];
        try {
            uartDevice.read(buffer, buffer.length)
            //do something with the data in the buffer byte array
        } catch (IOException e) {}

        //Returning true keeps the callback active. If you return false,
        //the callback will automatically unregister.
        return true;
    }

    @Override
    public void onUartDeviceError(UartDevice uart, int error) {}
};

@Override
protected void onStart() {
    super.onStart();
    mDevice.registerUartDeviceCallback(mUartCallback);
}

@Override
protected void onStop() {
    super.onStop();
    mDevice.unregisterUartDeviceCallback(mUartCallback);
}
```

User-space drivers

Permiten a los desarrolladores injectar nuevo hardware en el framework de Android, permitiéndoles interactuar con las ya establecidas API de Android.

Hay tres clasificaciones principales de controlador:

- controladores de GPS.
- Controladores de dispositivos de entrada humana (HID).
 - Eventos de botones.
 - KeyEvent.
 - Controller joysticks.
- Controladores de sensores. Android Things tiene un robusto framework de sensores.
 - Niveles de pH de agua.
 - Velocidad de viento.
 - Detección de movimiento.
 - Temperatura y presión.



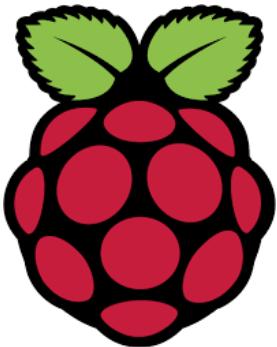
4

Hardware permitido por
Android Things
Android Things te permite crear
productos profesionales

System-on-Modules (SoMs)

El SoMs integran el SoC, RAM, Flash Storage, WiFi, Bluetooth y otros componentes en una sola placa.

La construcción de dispositivos está ahora al alcance de todos, incluso si nunca ha diseñado sistemas embebidos. Google proporciona una solución accesible a hardware y una plataforma de desarrollo de software fácil de usar, basada en Android Studio y el SDK de Android.



Para mas información: <https://developer.android.com/things/hardware/developer-kits.html>

Plataformas permitidas

Platform	Intel® Edison	Intel® Joule	NXP Pico i.MX6UL	NXP Argon i.MX6UL	Raspberry Pi 3
	 Learn More Where to buy Get Started	 Learn More Where to buy Get Started	 Learn More Where to buy Get Started	 Learn More Where to buy Get Started	 Learn More Where to buy Get Started
CPU & Memory	<ul style="list-style-type: none">Intel® Atom™500MHz dual-core x861GB RAM	<ul style="list-style-type: none">Intel® Atom™1.5GHz/1.7GHz quad-core x863GB/4GB RAM	<ul style="list-style-type: none">NXP i.MX6Ultralite500MHz ARM Cortex A7512MB RAM	<ul style="list-style-type: none">NXP i.MX6Ultralite500MHz ARM Cortex A7512MB RAM	<ul style="list-style-type: none">Broadcom BCM28371.2GHz quad-core ARM Cortex A531GB RAM
Storage	4GB eMMC	8GB/16GB eMMC	4GB eMMC	4GB eMMC	MicroSD card slot
Display	No	HDMI	No	No	HDMI
Camera	No	CSI-2	No	No	CSI-2
Audio	USB 2.0	USB 2.0	3.5mm Analog	3.5mm Analog	USB 2.0
Networking	Wi-Fi 802.11n Bluetooth® 4.0	Wi-Fi 802.11ac Bluetooth® 4.2	10/100 Ethernet Wi-Fi 802.11n Bluetooth® 4.1	10/100 Ethernet Wi-Fi 802.11n Bluetooth® 4.1	3.5mm Analog Output 10/100/1000 Ethernet Wi-Fi 802.11n Bluetooth® 4.1
USB	1x USB 2.0 OTG	2x USB 2.0 Host 1x USB 3.0 OTG	1x USB 2.0 Host 1x USB 2.0 OTG	1x USB 2.0 Host 1x USB 2.0 OTG	4x USB 2.0 Host

Raspberry Pi

Raspberry Pi es una computadora de placa reducida de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Algunos sistemas operativos oficiales para raspberry pi:

- ✓ RASPBIAN Debian Jessie
- ✓ OSMC
- ✓ LibreELEC
- ✓ RISC OS



3.3V PWR	1	5V PWR	2
I2C1 SDA	3	5V PWR	4
I2C1 SCL	5	GND	6
GPIO 4	7	UART0 TX	8
GND	9	UART0 RX	10
GPIO 17	11	GPIO 18	12
GPIO 27	13	GND	14
GPIO 22	15	GPIO 23	16
3.3V PWR	17	GPIO 24	18
SPI0 MOSI	19	GND	20
SPI0 MISO	21	GPIO 25	22
SPI0 SCLK	23	SPI0 CS0	24
GND	25	SPI0 CS1	26
Reserved	27	Reserved	28
GPIO 5	29	GND	30
GPIO 6	31	GPIO 12	32
GPIO 13	33	GND	34
GPIO 19	35	GPIO 16	36
GPIO 26	37	GPIO 20	38
GND	39	GPIO 21	40

Android Debug Bridge (ADB)

Android Debug Bridge (ADB) es una herramienta de líneas de comandos versátil que te permite comunicarte con una instancia de un emulador o un dispositivo Android conectado.

Esta herramienta proporciona diferentes acciones en dispositivos, como la instalación y la depuración de apps, y proporciona acceso a un shell Unix que puedes usar para ejecutar varios comandos en un emulador o un dispositivo conectado.

Es un programa cliente-servidor que incluye tres componentes:

- Un cliente, que envía comandos. El cliente se ejecuta en tu máquina de desarrollo. Puedes invocar un cliente desde un terminal de línea de comandos emitiendo un comando de ADB.
- Un daemon, que ejecuta comandos en un dispositivo. El daemon se ejecuta como un proceso en segundo plano en cada instancia del emulador o dispositivo.
- Un servidor, que administra la comunicación entre el cliente y el daemon. El servidor se ejecuta como un proceso en segundo plano en tu máquina de desarrollo.

Si tiene instalado Android Studio puedes encontrar la herramienta adb en [android_sdk/platform-tools/](#).

Android Things y Android Studio

Utilice los comandos en una consola linux donde tenga instalado Android Studio, el cual integra adb.

```
$ adb connect <ip-address>  
connected to <ip-address>:5555
```

```
$ adb connect Android.local
```

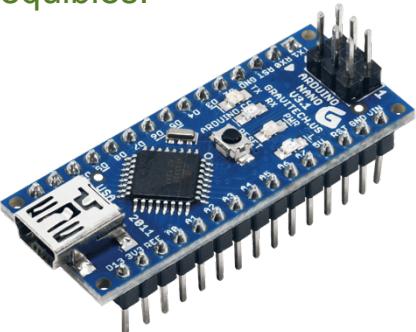
5

Arduino y NodeMcu
Arduino es hardware Libre

Arduino

Es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas computadora de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa.

Este está orientado para crear “proyectos caseros” y asequibles.



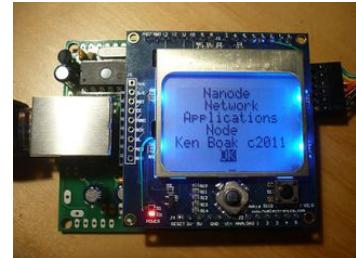
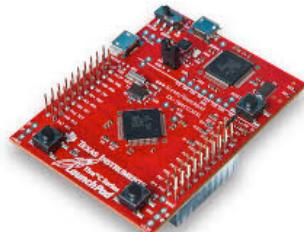
```
sketch_mar09a | Arduino 1.6.8
File Edit Sketch Tools Help
sketch_mar09a
void setup() {
  // put your setup code here, to run once:
}

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

3 Arduino Due (Programming Port) on COM1

Alternativas para Arduino

1. **WaspMote**: Es una plataforma modular opensource para construir redes de sensores inalámbricas de muy bajo consumo.
2. **TM4C123G LaunchPad (Tiva)**: Es una plataforma de evaluación de bajo costo para los microcontroladores ARM Cortex-M4F de Texas Instruments.
3. **Nanode**: Es una evolución de Arduino, pero que es capaz de conectarse a internet a través de un navegador o una API.
Nanode permite a cualquier desarrollador utilizarlo como su servidor privado de proyectos web, como conexión con otros dispositivos o como captador de datos ambientales a través de la colocación de sensores.



NodeMcu ESP8266

Comunicando dispositivos con NodeMcu

NodeMcu es una placa de desarrollo basado en ESP8266, integrando GPIO, PWM, IIC, 1-Wire y ADC todo en una misma placa.



Muchas gracias



¿Alguna pregunta?

Cualquier consulta a: haroldlopeziron@gmail.com