



Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores

EIE

Escuela de
Ingeniería Eléctrica

Introducción a IoT con MCU

MSc. Marco Villalta Fallas - `marco.villalta@ucr.ac.cr`

II Ciclo 2023

Conceptos

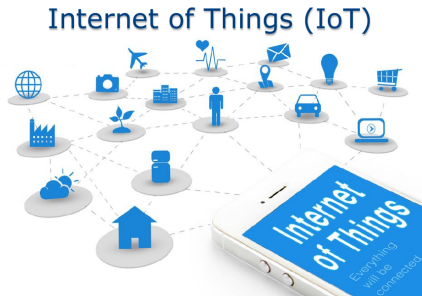
Antecedentes

- Acceso a Internet es más accesible
- IPv6
- Soporte de conectividad Wifi/Ethernet en dispositivos
- Dispositivos móviles y aplicaciones

Que es IoT?

Descripción general

- Sistema de dispositivos informáticos interrelacionados, máquinas mecánicas y digitales provistas de identificadores únicos (UID) y la capacidad de transferir datos a través de una red sin requerir interacción de persona a persona o de persona a computadora



Que es IoT?

Dispositivos

- Millones de objetos que pueden sensor, controlar, comunicar y compartir información
- Utilizan datos para realizar acciones
- Definición informal: Cualquier cosa que se puede encender/apagar y tiene acceso a la red



Impacto en la vida

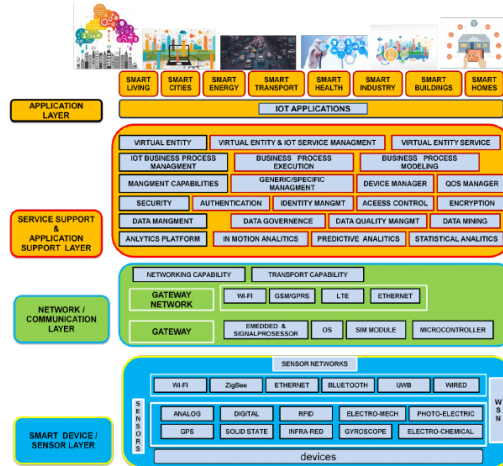
- En el trabajo
- En el hogar
- En la industria



Que compone lot?

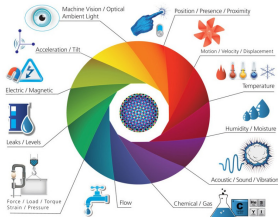
Arquitectura

- Infraestructura global
- Mezcla de tecnologías de hardware y software



Que compone lot?

Capa baja



- Integrada por objetos con sensores
- Interconectan mundo fisico con el digital
- Sensan temperatura, velocidad, humedad, presion, flujo, movimiento, electricidad, etc
- Se conectan a un sensor gateway con LAN(Ethernet,Wifi) y PAN (Zigbee, Bluetooth, UWB)
- Si sensor no ocupa conectividad a sensor gateway se conectan a servidores/aplicacionse con WAN (GSM,GPRS,LTE)

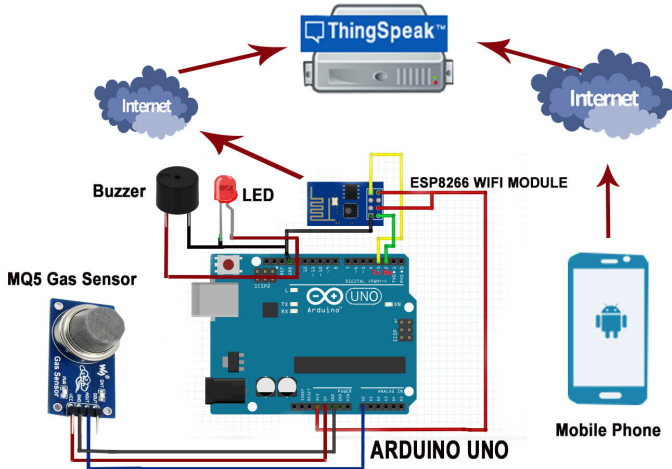
Que compone lot?

Otras capas

- Gateways y redes (Machine to machine)
- Capa administrativa de servicios (Se encarga de procesar la información: analisis, control de flujos, administración de dispositivos)
- Capa de aplicación (Ambientes inteligentes: Edificios, ciudades, industrias, agricultura, salud, energia, medio ambiente, turismo,etc.)

IoT y MCU

Perspectiva general



IoT y MCU

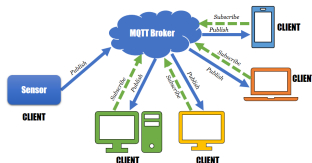
Algunos protocolos de datos

- MQTT (Message Queuing Telemetry Transport): Modelo publicador/suscriptor de mensajes extremadamente liviano. Útil para conexiones en ubicaciones remotas donde es necesario usar poca memoria y/o ancho de banda es importante.
- AMQP (Advanced Message Queuing Protocol): Protocolo de capa de aplicación estándar abierto para middleware orientado a mensajes.
- CoAP (Constrained Application Protocol): Protocolo de capa de aplicación destinado a su uso en dispositivos de Internet con recursos limitados, como los nodos WSN.
- WebSocket: Especificación (parte de HTML5) que define una conexión de socket único dúplex completo a través de la cual se pueden enviar mensajes entre el cliente y el servidor.

IoT y MCU

MQTT

- Para hacer uso de protocolos de datos con Arduino es necesario un shield o una SBC (Raspberry PI)
- Incluir bibliotecas de shield y protocolo de datos
 - Ethernet.h
 - PubSubClient.h
- En ambiente simulado Python realizará un puente de comunicaciones sustituyendo el shield de Wifi/Ethernet y funciones de bibliotecas de protocolo de datos
- MQTT broker: Servidor que puede enviar y recibir mensajes a clientes, no están diseñados para guardar datos
- Una forma de proteger los datos es utilizando un *token*.



Python: MQTT

Cliente

Se utiliza la clase cliente de *paho.mqtt*. Flujo típico incluye:

- 1 Crear instancia de cliente
- 2 Conectar a un broker con alguna función **connect*()**
- 3 Llamar a un de las funciones **loop*()** para mantener el flujo de tráfico con el broker
- 4 Utilizar **subscribe()** para suscribirse a un tema y recibir mensajes
- 5 Utilizar **publish()** para publicar mensajes a un broker/servidor desde un cliente.
- 6 Utilizar **disconnect()** para desconectarse del broker
- 7 Utilizar **on_message()** para publicar mensajes que pueden ser leídos por un cliente.

Arduino: MQTT

Cliente

Se utiliza una biblioteca, por ejemplo

`https://github.com/arduino-libraries/ArduinoMqttClient`. Es necesario un shield de wifi o ethernet.

- 1 Incluir bibliotecas necesarias
- 2 Crear un archivo de encabezado para las credenciales de Wi-Fi
- 3 Configurar el **dispositivo publicador** para crear topics y publicar a un broker.
- 4 Configurar el **dispositivo subscriptor**

Arduino: MQTT

Funciones importantes

- **WiFiClient wifiClient** Crea un cliente de Wi-Fi
- **MqttClient mqttClient(wifiClient)** Conecta el cliente Wi-Fi al cliente MQTT
- **WiFi.begin(ssid, pass)** Conectar a la red local de Wi-Fi
- **mqttClient.connect(broker, port)** Conectarse al broker
- **mqttClient.poll()** Mantiene la conexión, utilizado en función loop().
- **mqttClient.beginMessage(topic)** Crea un nuevo mensaje para publicar
- **mqttClient.print()** Imprimir mensaje
- **mqttClient.endMessage()** Publica el mensaje al broker
- **mqttClient.subscribe(topic)** Suscribirse a un tema/topic
- **mqttClient.available()** Revisar si un mensaje esta disponible en el tema/topic
- **mqttClient.read()** Leer un mensaje

Plataformas IoT

- Una plataforma IoT es un servicio integrado de tecnologías que ofrece lo necesario para poner en línea objetos físicos
- Proporcionan la infraestructura que utiliza para crear las características específicas de una solución.
- Objetivo: Proporcionar toda la funcionalidad genérica para una aplicación
- Funciones que debería realizar:
 - Adquiera datos a través de sensores
 - Analizar datos localmente
 - Conectar a la nube para transmitir datos y recibir comandos
 - Almacenar datos en la nube
 - Analice datos en la nube para crear ideas
 - Dirige las cosas para realizar tareas específicas
 - Presentar información a los usuarios
- Plataformas comerciales: AWS, Azure, Google, Arduino, Samsung, IBM, Bosch, Cisco, Intel, etc

Thingsboard I

Descripción

- Open source IoT Platform para recolectar, procesar y visualizar datos. También administración de dispositivos.
- Provee monitoreo y control utilizando API de servidor. Se puede definir relaciones entre dispositivos, activos, clientes y otros dispositivos
- Define cadenas de reglas para el procesamiento de datos.
- Habilita alarmas por eventos de telemetría, actualizaciones de atributos y acciones de usuarios.
- Acceso en `iot.eie.ucr.ac.cr` o localmente si se instala, mejor usar Chrome
- Si utiliza MQTT mensajes se publican en `v1/devices/me/telemetry`
- Información debe empaquetarse como JSON.



Thingsboard II

Comunicacion servidor a MCU

- Con MQTT mensajes que envia el MCU a Thingsboard se publican en el tema `v1/devices/me/telemetry`
- Para enviar mensajes del servidor al MCU se utilizan llamadas RPC en el tema `v1/devices/me/rpc/request/`
- Para diferenciar los dispositivos de control se utilizan los nombres de los metodos (p.e. `getValue`, `setValue`).
- Revisar ejemplos en <https://github.com/thingsboard/thingsboard/tree/master/tools/src/main/python> y <https://shiyaztech.wordpress.com/2018/08/25/remote-procedure-calls-rpc-on-thingsboard-iot-platform>

Aplicación

Hola MQTT

main

```
import paho.mqtt.client as mqtt #Importar biblioteca MQTT
import time

broker="iot.eie.ucr.ac.cr"
client = mqtt.Client("python1_id")#create new instance
client.on_connect=on_connect #bind connect call back function
client.on_disconnect=on_disconnect #bind disconnect call back function
#client.on_log=on_log #bind logging call back function
client.on_message=on_message #bind message call back function
print("Connecting to broker ",broker)
client.connect(broker) #connect to broker
client.loop_start() #Start loop
client.subscribe("house/sensor1")
client.publish("house/sensor1","my first message")
time.sleep(4)
client.loop_stop() #Stop loop
client.disconnect() # disconnect
```

Hola MQTT

callbacks

```
def on_log(client, userdata, level, buf): #Callback para logging
    print("log: "+buf)
def on_connect(client, userdata, flags, rc): #Callback cuando se conecta
    if rc==0:
        print("connected OK")
    else:
        print("Bad connection Returned code=",rc)
def on_disconnect(client, userdata, flags, rc=0): #Callback cuando se desconecta
    print("DisConnected result code "+str(rc))
def on_message(client, userdata, msg): #Callback cuando recibe mensaje
    topic=msg.topic
    m_decode=str(msg.payload.decode("utf-8","ignore"))
    print("message received",m_decode)
```

Thingsboard

Creación de un Dashboard

- 1 Crear un device
- 2 Mapear un device a un widget
- 3 Asignar un widget a un dashboard
- 4 Agregar el widget

Otras Referencias

- <https://pypi.org/project/paho-mqtt/>
 - <https://thingsboard.io/docs/samples/arduino/temperature/>
 - <https://thingsboard.io/docs/samples/esp8266/gpio/>
- Instalación thingsboard:
- <https://thingsboard.io/docs/user-guide/install/docker-windows/>
 - <https://thingsboard.io/docs/user-guide/install/docker/>
 - <https://mqtt.org/>
 - <https://test.mosquitto.org/>
 - <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/>