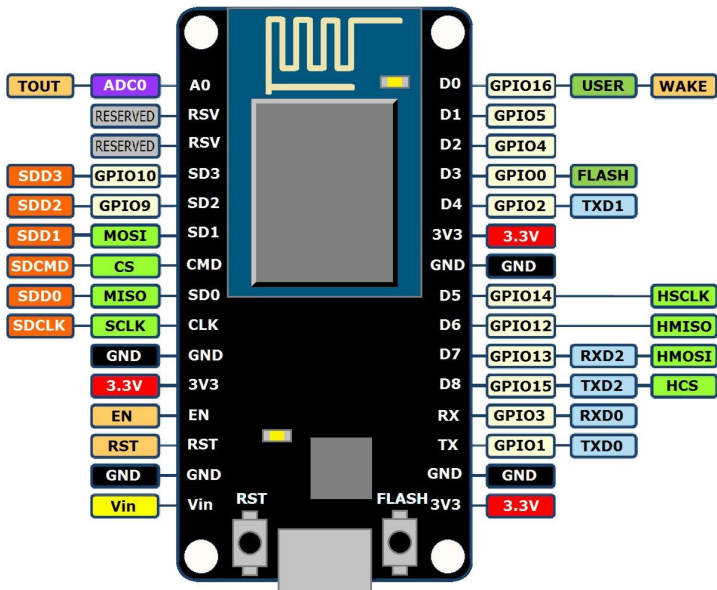


IoT Workshop

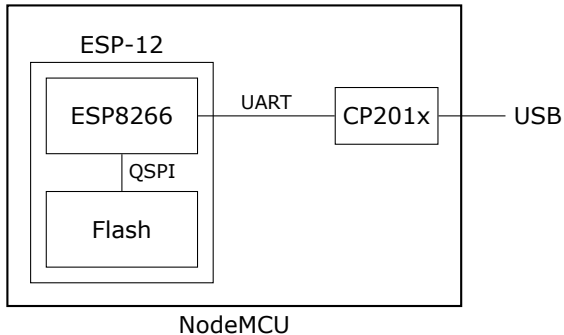
Trygve Laugstøl <trygvis@trygvis.io>

NodeMCU

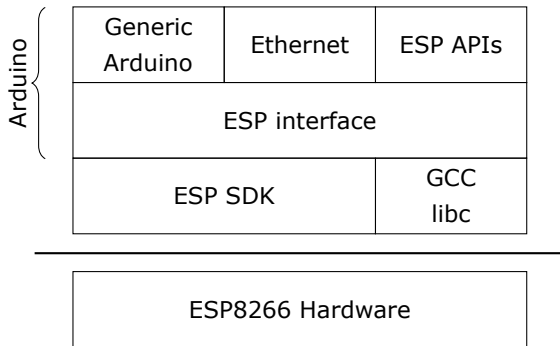
NodeMCU hardware



NodeMCU hardware



ESP8266 software layers



ESP8266 + Arduino

- ▶ Standard Arduino IDE
- ▶ ESP8266 Arduino core
 - ▶ <https://github.com/esp8266/Arduino>

Arduino IDE



Arduino code structure

```
void setup() {  
    // Called once  
}
```

```
void loop() {  
    // Called repeatedly  
}
```

Arduino file structure

foo/

foo.ino

config.h

Generic Arduino APIs

```
// Pin: D0, D1, etc.  
// Mode: OUTPUT, INPUT, INPUT_PULLUP  
void pinMode(uint8_t pin, uint8_t mode);  
  
// State: HIGH, LOW, true/false, 1/0  
void digitalWrite(uint8_t pin, uint8_t state);  
int digitalRead(uint8_t pin);  
  
unsigned long now millis();  
unsigned long now micros();
```

ESP Arduino APIs

```
class {  
    void restart();  
    uint32_t getFreeHeap();  
    uint32_t getChipId();  
  
    ...  
} ESP;
```

// Usage

```
ESP.restart();
```

ESP

```
// Top of file
#include <ESP8266WiFi.h>

// In setup()
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
```

ESP Arduino APIs

```
class {  
    String macAddress();  
  
    wl_status_t status();  
    int32_t RSSI();  
  
    IPAddress localIP();  
    IPAddress subnetMask();  
    IPAddress gatewayIP();  
    IPAddress dnsIP(uint8_t dns_no = 0);  
  
    ...  
} WiFi;  
  
// Usage:  
  
Serial.println(WiFi.localIP().toString());
```

What is IoT

What is IoT

- ▶ Not “a computer connected to the internet”
 - ▶ Then it is really just another computer connected to the internet
- ▶ Must be something else
 - ▶ It is simply devices that are resource constrained
 - ▶ Usually in more than one way
- ▶ Autonomous operation, the connection might not be permanent

IoT is just a concept

- ▶ *The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data.*¹

¹Wikipedia "Internet of Things"

What is an IoT Device?

What is an IoT Device?

- ▶ Constrained in (one or more of):
 - ▶ Memory
 - ▶ CPU
 - ▶ Network bandwidth and/or latency
 - ▶ Storage
- ▶ Has connectivity
 - ▶ Bluetooth
 - ▶ Wi-Fi
 - ▶ NB-IoT
 - ▶ LTE Cat-M
 - ▶ LoRa
 - ▶ Proprietary radio

IoT Devices - Example chips

Protocol	Chip	Specs
Bluetooth 4/5	nRF52x	32-64 MHz, Cortex-M0/M4F, 24-256k RAM, 192-1024 k Flash, \$1.88-\$3.85
WiFi	ESP8266/ESP32	80MHz-160MHz, 1-2 cores, ~80k RAM, < \$1 - \$4.53
LoRa	Semtech	\$3.23 - \$4.74

ESP8266 details - Power usage

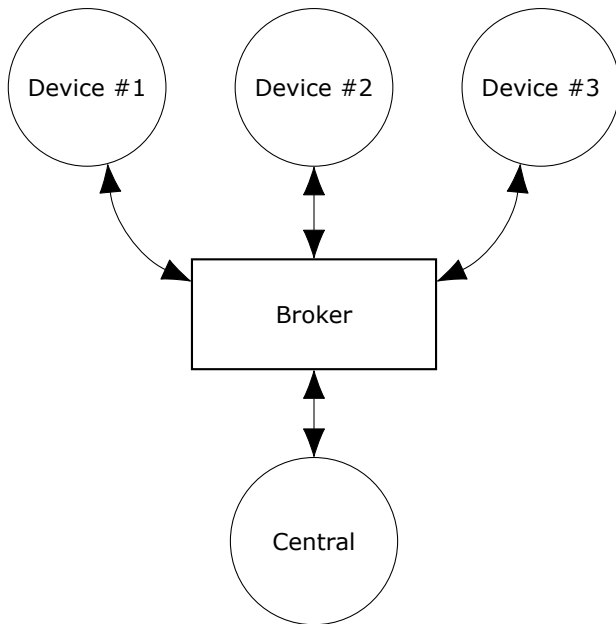
State	Current usage
Off	0.5 μ A
Deep sleep with RTC	20 μ A
Light sleep (with Wi-Fi)	1 mA
Sleep with peripherals	15 mA
TX	170 mA

Lecture: MQTT

MQTT

- ▶ *Message Queuing Telemetry Transport*
- ▶ [Wikipedia: MQTT](#)

Device and application architecture with MQTT



MQTT Topic

The temperature sensor:

- ▶ Publishes on:
 - ▶ myapp/\$device-id/temperature
 - ▶ myapp/\$device-id/humidity
 - ▶ myapp/\$device-id/alert
- ▶ Subscribes to:
 - ▶ myapp/\$device-id/command

The central application:

- ▶ Subscribes to:
 - ▶ myapp/#/temperature
 - ▶ myapp/#/humidity
- ▶ Publishes on:
 - ▶ myapp/\$device-id/command

MQTT - Implementations

- ▶ Mosquitto
- ▶ Eclipse Paho
- ▶ RabbitMQ
- ▶ ActiveMQ

MQTT Cloud Connectors

- ▶ Cloud
 - ▶ Amazon IoT
 - ▶ Google Cloud IoT
 - ▶ Microsoft Azure IoT
 - ▶ CloudMQTT (at Heroku)
- ▶ DIY
 - ▶ ThingMQ
 - ▶ HiveMQ

MQTT - The protocol

Agents have one of two roles:

- ▶ *Client*
 - ▶ Publishes *messages*
 - ▶ Subscribes / unsubscribes to *topics*
 - ▶ Keep alive
- ▶ *Broker* (aka Server)
 - ▶ Handles network connections
 - ▶ Keeps subscriptions
 - ▶ Manages client
 - ▶ Timeouts and disconnects
 - ▶ *(last) will*
 - ▶ Persistence of retained messages

MQTT - The protocol - MQTT Topic

- ▶ Topic name: `foo/bar/baz`
- ▶ Topic filter
 - ▶ `foo/bar/?`
 - ▶ `foo/#`

MQTT - The protocol - Retained message

Message is kept by the server even after disconnect

- ▶ CONNECT
- ▶ PUBLISH
 - ▶ RETAIN
 - ▶ \$app/\$device/temperature
 - ▶ 22.3
- ▶ DISCONNECT

Later on:

- ▶ SUBSCRIBE
 - ▶ \$app/#/temperature
- ▶ PUBLISH
 - ▶ \$app/\$device/temperature
 - ▶ 22.3

MQTT - The protocol - Will message

Message sent when you disconnect

Client #1:

1. CONNECT

- ▶ WILL TOPIC: \$app/\$device/online
- ▶ WILL PAYLOAD: 0

2. PUBLISH

- ▶ \$app/\$device/online
- ▶ 1

3. DISCONNECT

Broker

1. *To all subscribers* PUBLISH

- ▶ \$app/\$device/online
- ▶ 0

MQTT on Arduino

PubSubClient is our MQTT client implementation.

```
#include <PubSubClient.h>
```

```
WiFiClient wifiClient;
```

```
PubSubClient mqtt(wifiClient);
```

```
void callback(char* topic,  
              byte* payload,  
              unsigned int length);
```

```
void setup() {  
    // Configure WiFi  
    mqtt.setServer(mqtt_server, 1883);  
    mqtt.setCallback(callback);  
}
```

MQTT on Arduino

```
void loop() {  
    if (!mqtt.connected()) {  
        reconnect();  
    }  
    else {  
        mqtt.loop();  
    }  
  
    // Do work  
}  
  
void reconnect() {  
    do {  
        Serial.println("Connecting to MQTT");  
    } while (!mqtt.connect(client_id));  
    Serial.println("Connected to MQTT server");  
  
    // Subscribe to any topics you need
```

Assignment

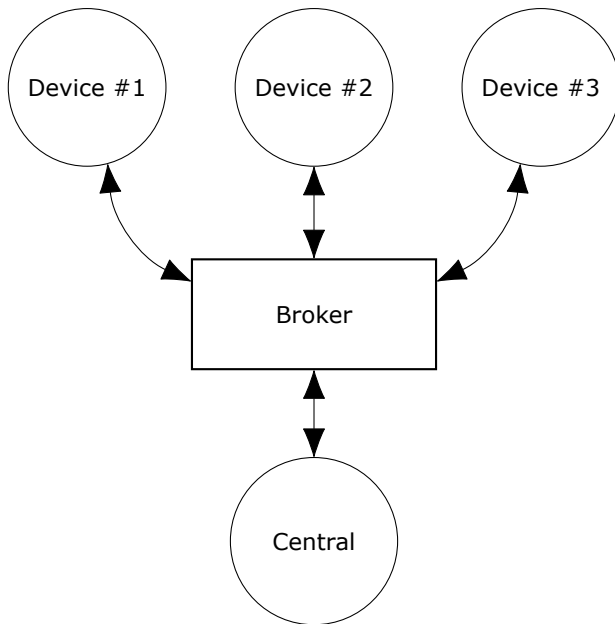
► mqtt

MQTT topic architecture

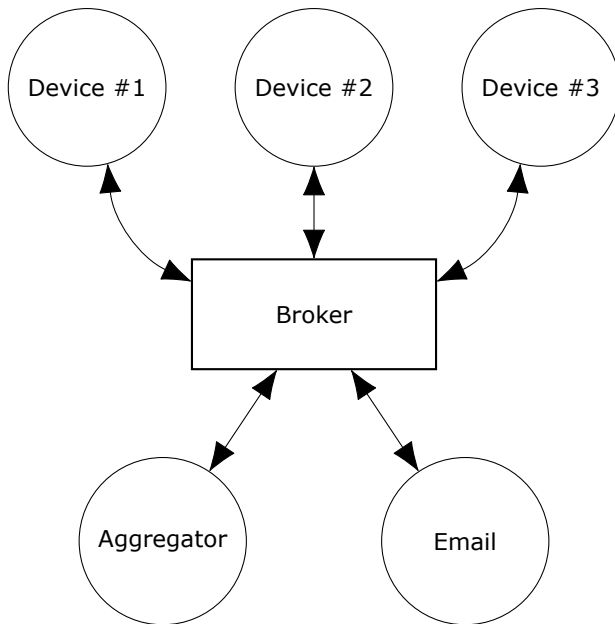
The central application is split:

- ▶ An aggregating agent:
 - ▶ myapp/#/temperature
 - ▶ myapp/#/humidity
- ▶ Emailing agent
 - ▶ myapp/\$device-id/alert
- ▶ Publishes on:
 - ▶ myapp/\$device-id/command

MQTT topic architecture



MQTT topic architecture



MQTT - Patterns

- ▶ Combining MQTT and HTTP
- ▶ Using web sockets transport

Assignment

► mqtt2

Assignment

► mqtt3