IoT Workshop

Trygve Laugstøl <trygvis@trygvis.io>

NodeMCU

NodeMCU hardware

TOUT	ADC0	A0		D0	GPIO16	USER	WAKE
	RESERVED	RSV		D1	GPIO5		
	RESERVED	RSV		D2	GPIO4		
SDD3	GPIO10	SD3		D3	GPIO0	FLASH	
SDD2	GPIO9	SD2		D4	GPIO2	TXD1	
SDD1	MOSI	SD1		3V3	3.3V		
SDCMD	CS	CMD		GND	GND		
SDD0	MISO	SD0		D5	GPIO14		HSCLK
SDCLK	SCLK	CLK		D6	GPIO12		HMISO
	GND	GND		D7	GPIO13	RXD2	HMOSI
	3.3V	3V3		D8	GPIO15	TXD2	HCS
	EN	EN		RX	GPIO3	RXD0	
	RST	RST		TX	GPIO1	TXD0	
	GND	GND		GND	GND		
	Vin	Vin		3V3	3.3V		
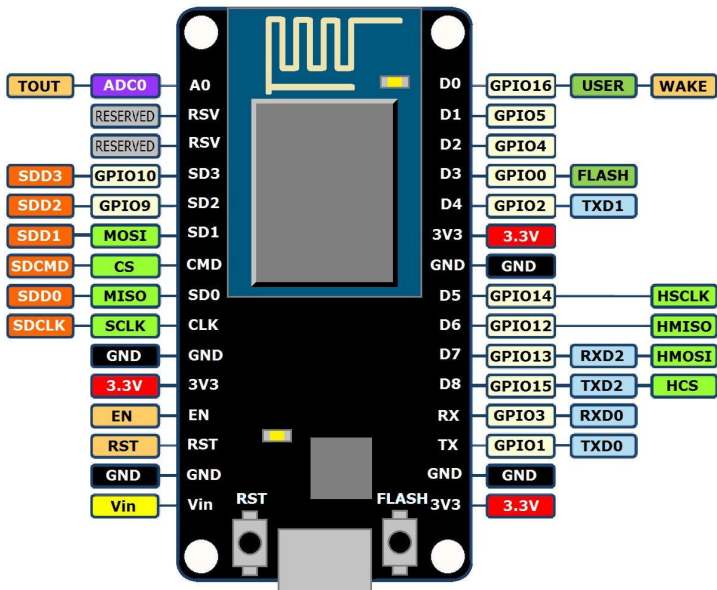
NodeMCU hardware
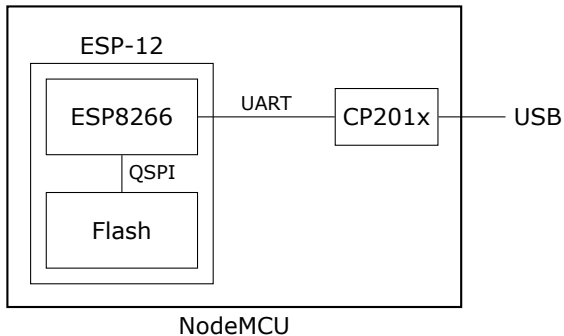
ESP8266 software layers

Generic Arduino	Ethernet	ESP APIs
ESP interface		
ESP SDK		GCC libc

Arduino brackets the top three sections (Generic Arduino / Ethernet / ESP APIs and ESP interface).
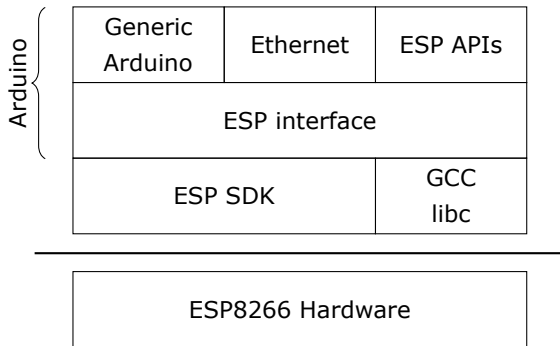
ESP8266 Hardware

ESP8266 + Arduino

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

Arduino IDE

Fil Rediger Skisse Verktøy Hjelp

sketch_apr25a

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8
9  }
```

Module), 80 MHz, 4M (1M SPIFFS), v2 Higher Bandwidth, Disabled, None, Only Sketch, 921600 on /dev/ttyUSB0

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

```cpp
class {
    void restart();
    uint32_t getFreeHeap();
    uint32_t getChipId();


    ...
} ESP;

// Usage
ESP.restart();
```

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]

---

[1] Wikipedia "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

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

Lecture: MQTT

MQTT

▶ *Message Queuing Telemetry Transport*
▶ Wikipedia: MQTT

Device and application architecture with MQTT
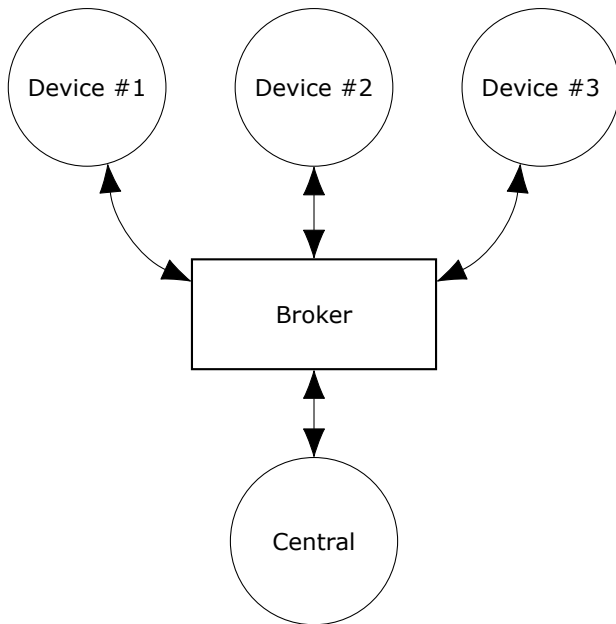
MQTT - Implementations

- 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*
- *Broker* (aka Server)
    - Handles network connections
    - Keeps subscriptions
    - Manages client
        - Disconnects
        - *(last) will*
    - Persistence of retained messages

▶ Size oriented
▶ Flags indicate type of remaining bytes
  ▶ Packet type
  ▶ Topic name
  ▶ Payload

MQTT Connect

- CONNECT
  - clientId
  - username
  - password
  - keepAlive

- Keep alive
  - PINGREQ
  - PINGRESP

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

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

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 Topic

The temperature sensor:

- ▶ Publishes on:
  - ▶ `myapp/$device-id/temperature`
  - ▶ `myapp/$device-id/humidity`
  - ▶ `myapp/$device-id/altert`
- ▶ Subscribes to:
  - ▶ `myapp/$device-id/command`

The central application:

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

MQTT on Arduino

PubSubClient is our MQTT client implementation.

```
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() {
    while (!mqtt.connect(client_id));

    mqtt.subscribe(topic_pattern);
}
```
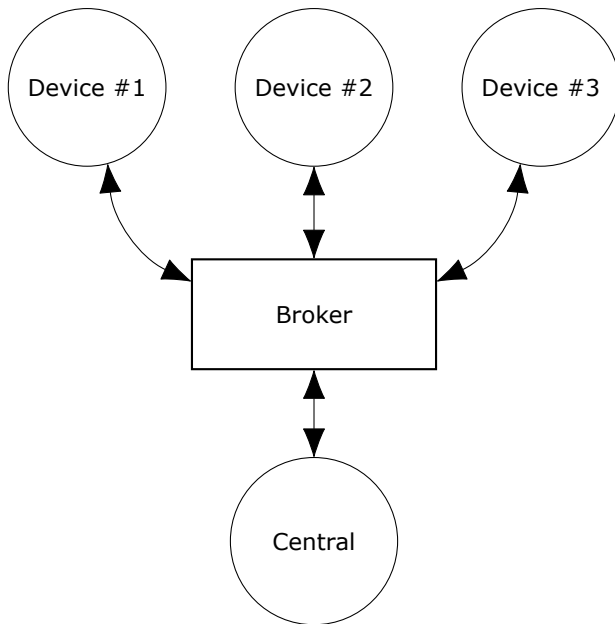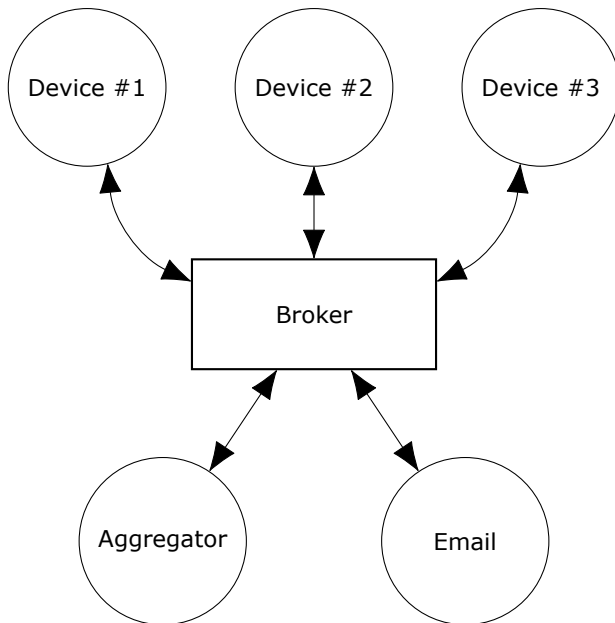
Assignment

- mqtt

MQTT topic architecture

The central application is split:

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

MQTT topic architecture

MQTT topic architecture

MQTT - Patterns

▶ Combining MQTT and HTTP
▶ Using web sockets transport

Assignment

- mqtt2

Assignment

- `mqtt3`