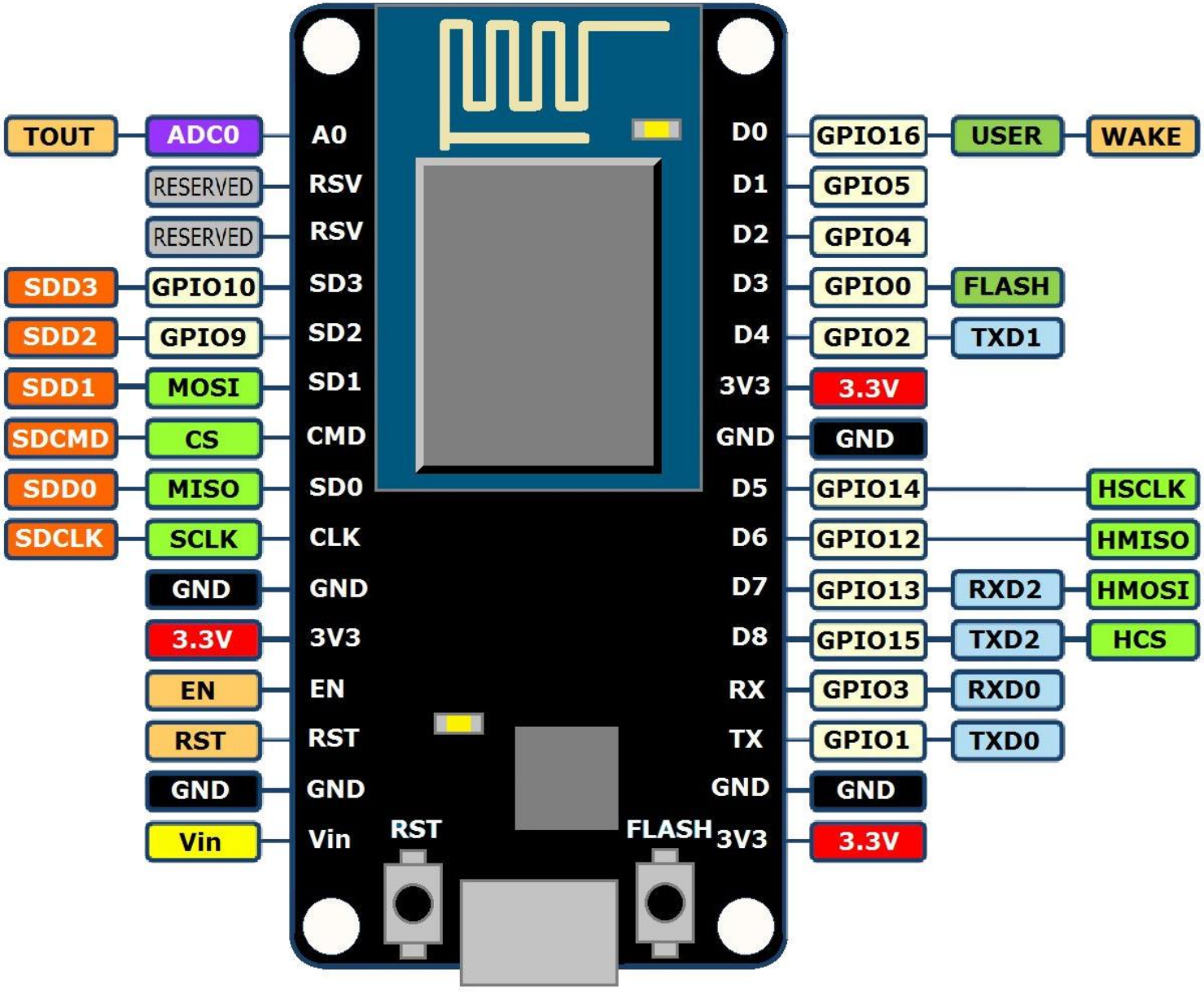
IoT Workshop

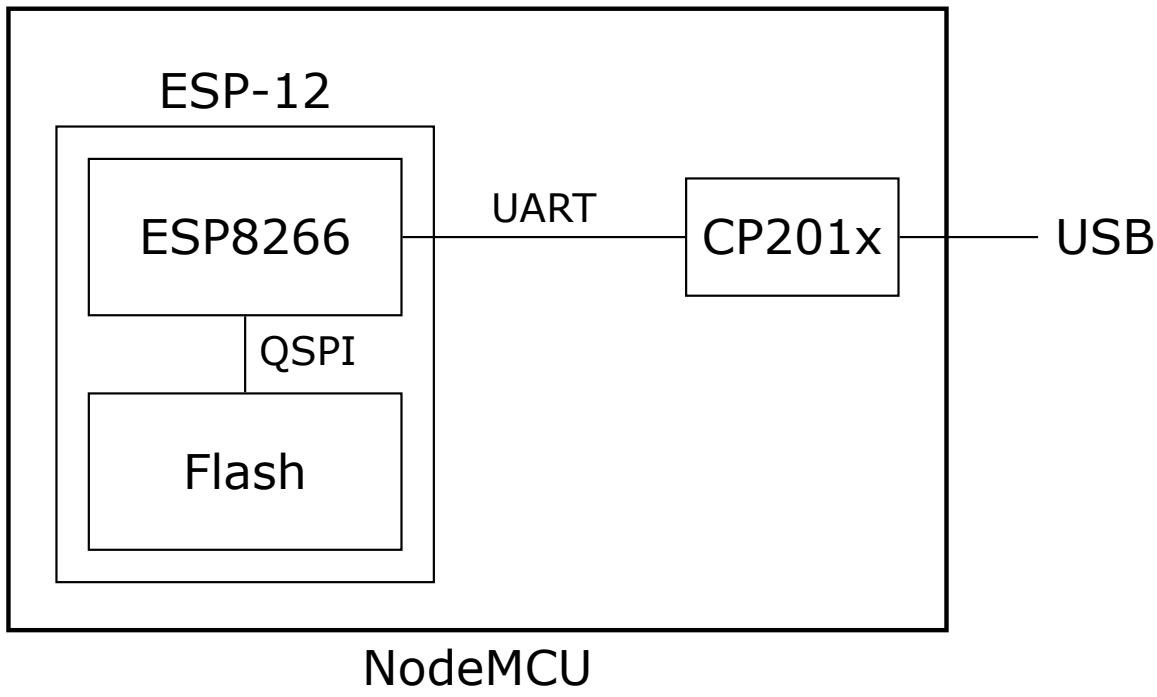
Trygve Laugstøl <trygvis@trygvis.io>

NodeMCU

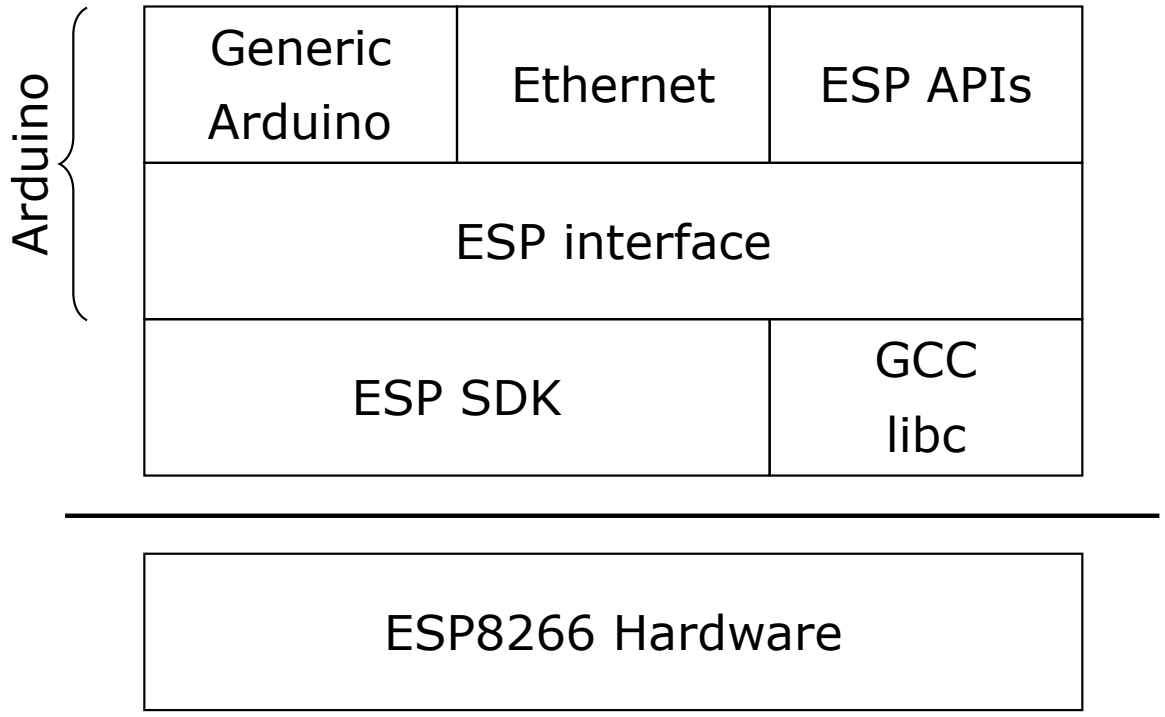
NodeMCU hardware



NodeMCU hardware



ESP8266 software layers



ESP8266 + Arduino

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

Arduino IDE


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

```
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 - Bluetooth 4/5 chips

Chip	CPU	Freq	RAM	Flash	Price
nRF52810	Cortex-M4	64 MHz	24k	192k	$1.88
nRF52832	Cortex-M4F		32k	256k	$2.54
			64k	512k	$2.59
nRF52840	Cortex-M4F		256k	1024k	$3.85

- nRF52810: High performance, entry-level Bluetooth 4/ANT/2.4GHz SoC
- nRF52832: High performance Bluetooth 4/ANT/2.4GHz SoC
- nRF52840: Advanced multi-protocol System-on-Chip Supporting: Bluetooth 5, ANT/ANT+, 802.15.4 and 2.4GHz proprietary

IoT Devices - LoRA

Modules

Module	Data Rate	Price
RN2483A-I/RM104		$12.05 @ 250
CMWX1ZZABZ-078	SX1276	$10.74 @ 1000
RF-LORA-868-SO	SX1272	$16.55 @ 1000

Chips

Chip	Price
SX1281	$3.23
SX1272	$4.25
SX1276	$4.25
SX1279	$4.74

IoT Devices - NB-IoT

Module	Price
uBlox SARA-N210	~$10 @ 100
Sierra Wireless HL7800_1103933	$15.72

IoT Devices - Wi-Fi

Chip	CPU	Freq	ROM	RAM	Price
ESP8266	Tensilica L106	160 MHz	N/A	~50 kB	< $1

ESP32 - dual cpu, Wi-Fi, Bluetooth 4 ESP32-D0WDQ6 2x Xtensa @ 160MHz $ 4.53 @ 10

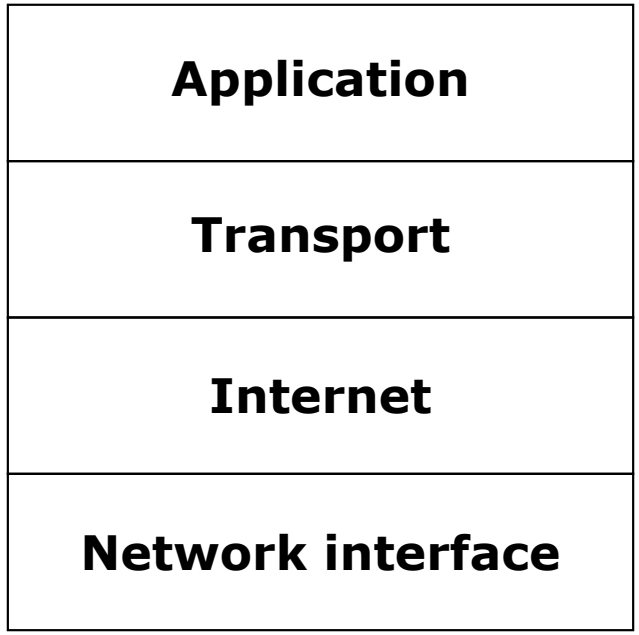
ESP8266 details - Power usage

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

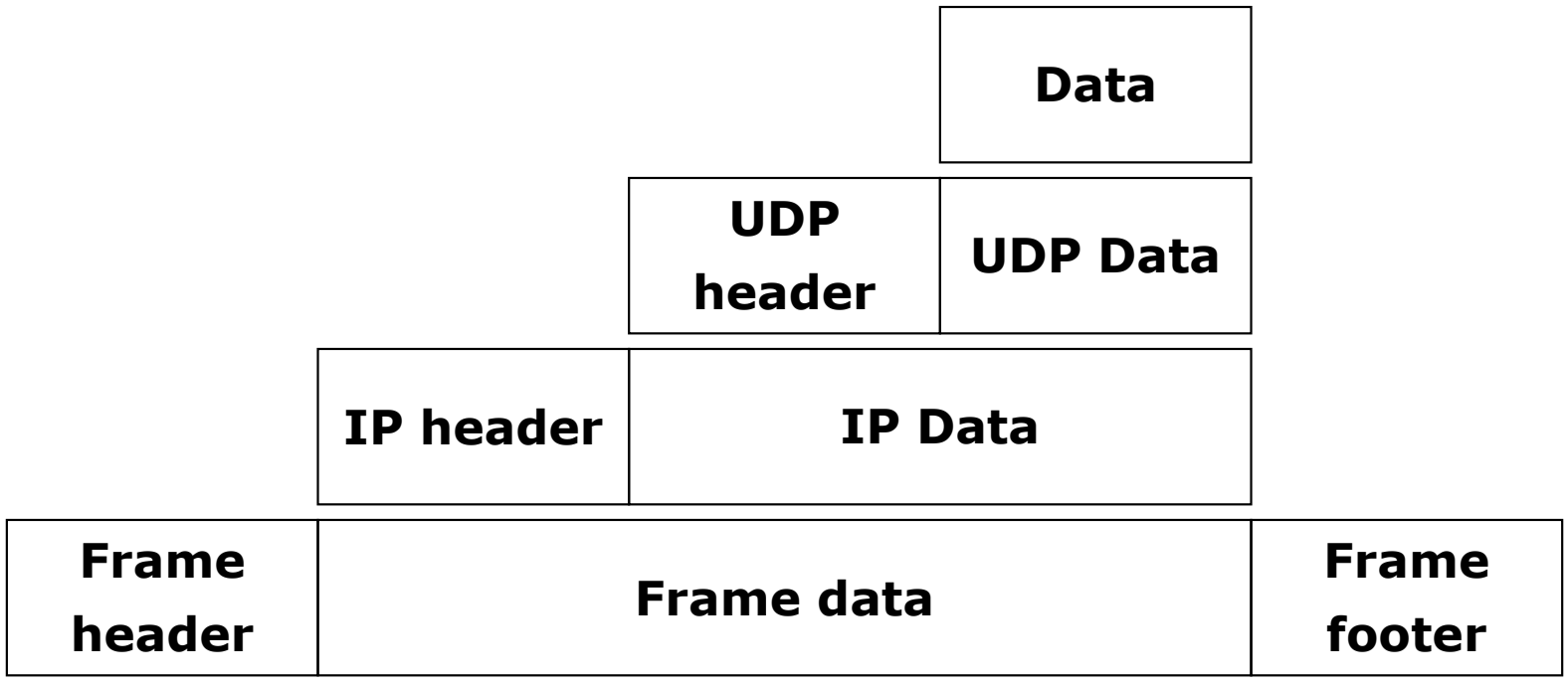
Going back to basics

What is the internet again?

TCP/IP Layers



Packet encapsulation



Network interface

- Ethernet
  - 10BASE5, 10BASE2, 10BASE-T / 100BASE-TX / 1000BASE-TX
- Wi-Fi
  - 802.11a/b/g/n
- RS-232

Internet

- IP
- ICMP

Transport

- TCP
- UDP
- SCTP
- QUIC

Layer 7: Application Layer

- HTTP
- DNS
- MQTT
- CoAP
- (everything else..)

Details: IP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				TOS								Packet length															
4	32	Identification																X	D	M	Fragement offset											
8	64	TTL								Protocol								Header checksum															
12	96	Source ip address																															
16	128	Destination ip address																															
20	160	Options (variable length)																															

Details: UDP Header

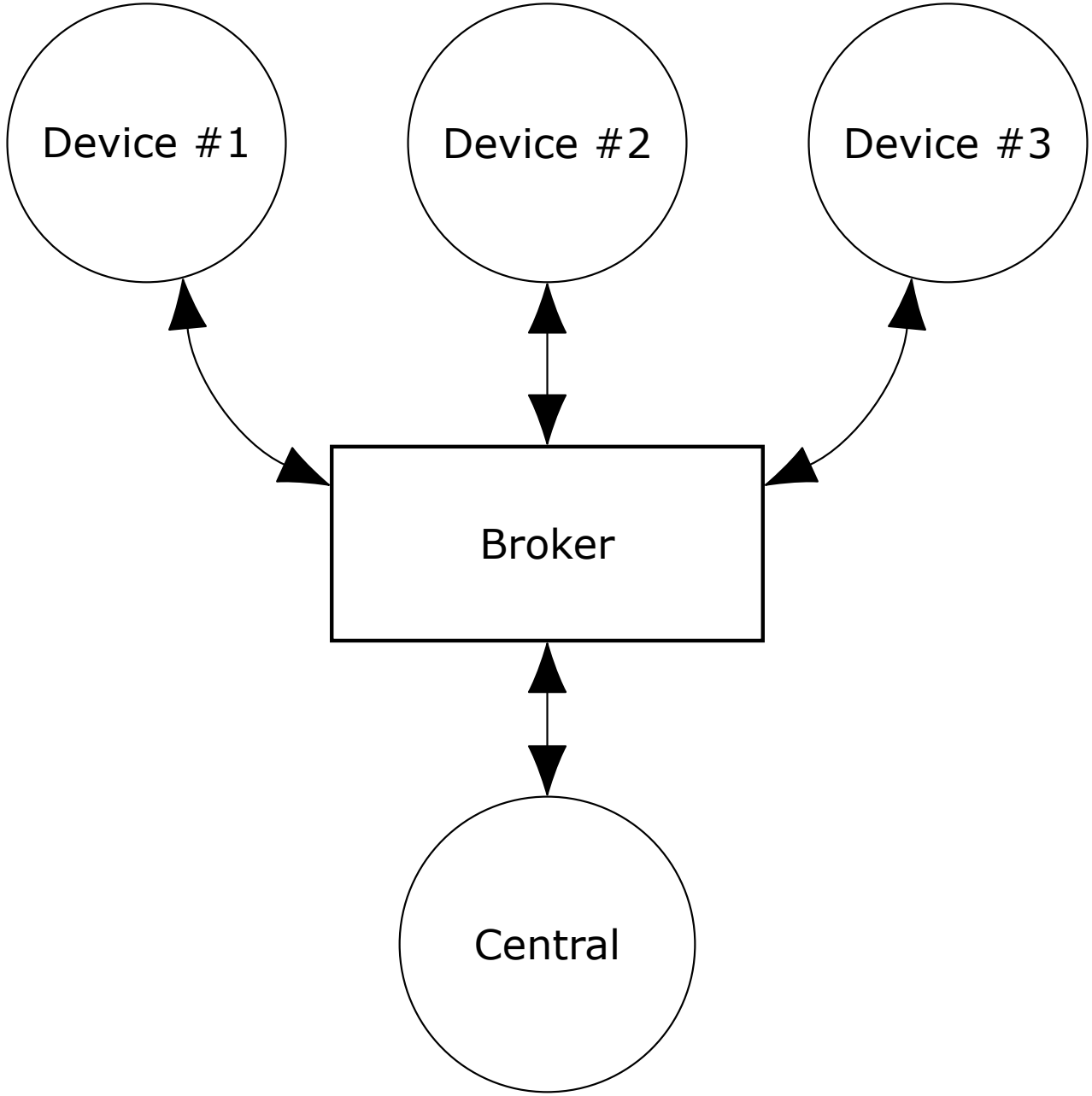
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	source port																destination port															
4	32	length																checksum															

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

MQTT - The protocol - MQTT Packet

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

MQTT Connect

- `CONNECT`
  - `clientId`
  - `username`
  - `password`
  - `keepAlive`
- Keep alive
  - `PINGREQ`
  - `PINGRESP`

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 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

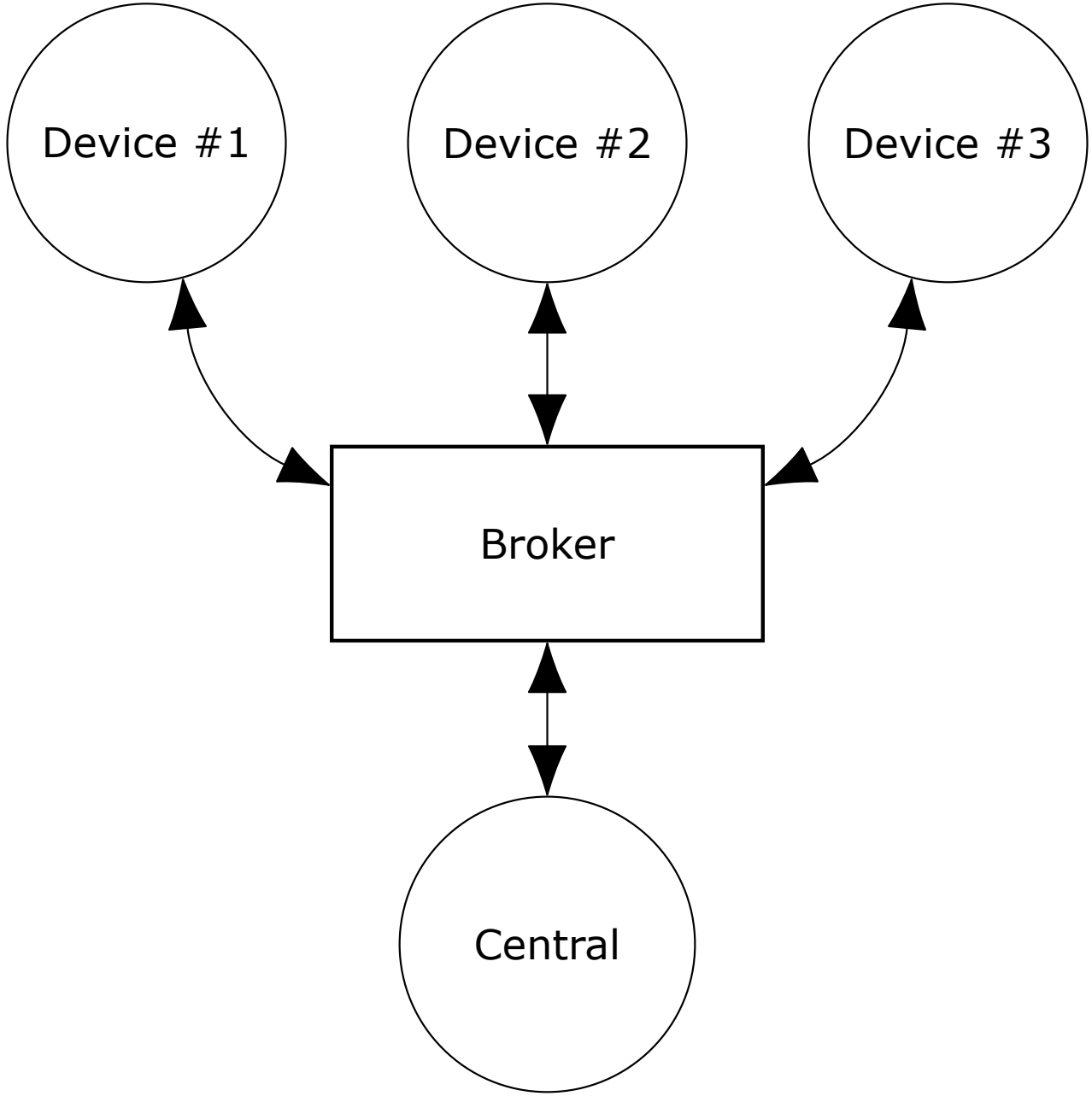
- `mqtt`

MQTT topic architecture

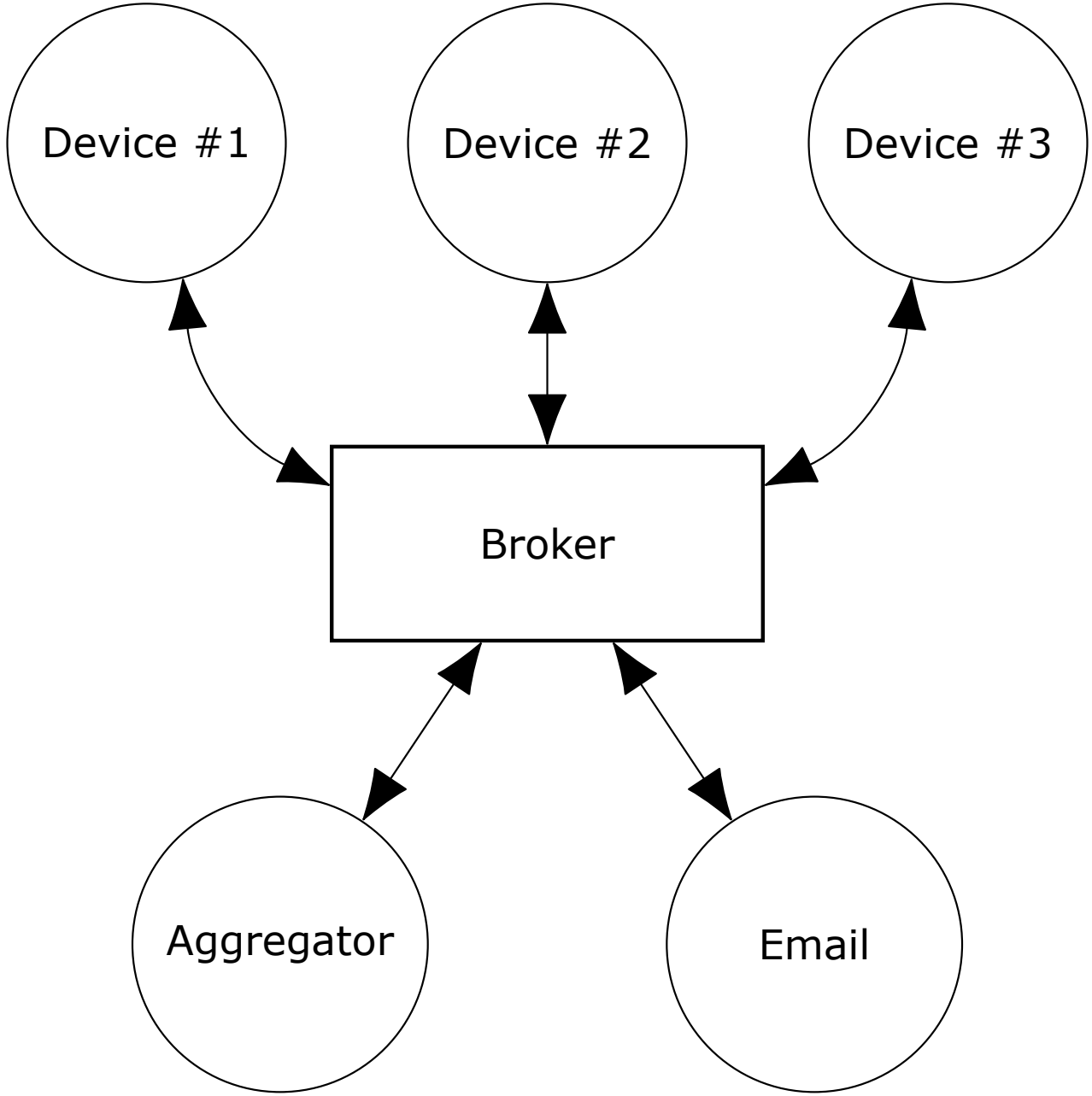
The central application is split:

- 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`