IoT Workshop

Trygve Laugstøl <trygvis@trygvis.io>

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
- Connected
  - Bluetooth
  - Wi-Fi
  - NB-IoT
  - LTE Cat-M
  - IR
  - UART
  - CAN

Typical IoT chips - Bluetooth 4/5

Chip	CPU	Freq	RAM	Flash	Price
nRF52810	Cortex-M4	64 M	Hz 24k	192k	$1.88
High perf	ormance,	entry	-level Bl	uetooth	4/ANT/2.4GHz SoC

nRF52832 Cortex-M4F 32k 256k $2.54 64k 512k $2.59 High performance Bluetooth 4/ANT/2.4GHz SoC

nRF52840 Cortex-M4F 256k 1024k $3.85 Advanced multi-protocol System-on-Chip Supporting: Bluetooth 5, ANT/ANT+, 802.15.4 and 2.4GHz proprietary

Typical IoT chips - 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

ESP8266 details - Arduino

https://github.com/esp8266/Arduino

Going back to basics

What is the internet again?

OSI model

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

- Wikipedia: OSI model
- Wikipedia: OSI model#Examples

Layer 1: Physical Layer

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

Layer 2: Data Link Layer

- Ethernet
- WiFi
- Bluetooth
- Token Ring

Layer 3: Network Layer

- IP
- ICMP
- IPX

Layer 4: Transport Layer

- TCP
- UDP

Layer 5: Session Layer

- “sockets”
- NetBIOS

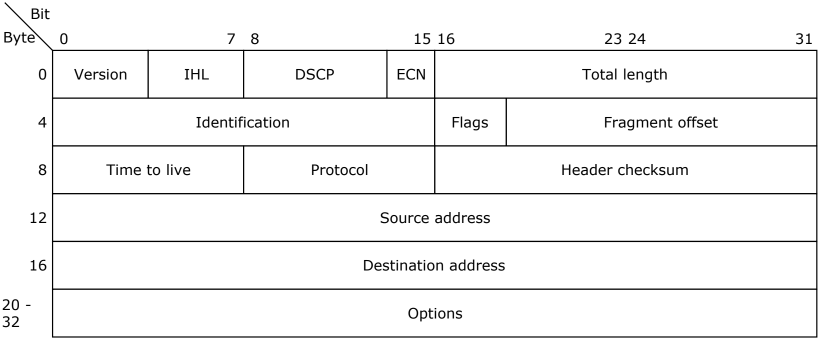
Layer 6: Presentation Layer

- SSL

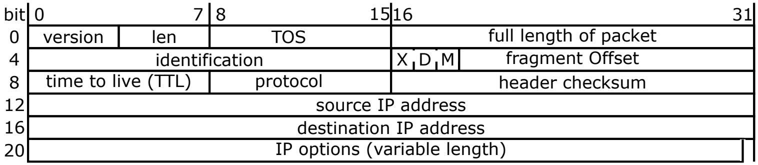
Layer 7: Application Layer

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

Details: IP

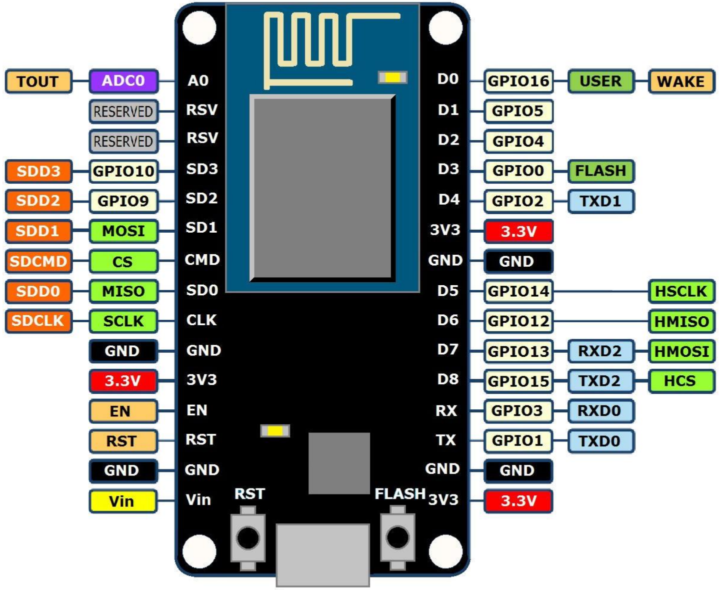

Bit
Byte
0
7
8
15
16
23
24
31
0
Version
IHL
DSCP
ECN
Total length
4
Identification
Flags
Fragment offset
8
Time to live
Protocol
Header checksum
12
Source address
16
Destination address
20 - 32
Options

Details: IP

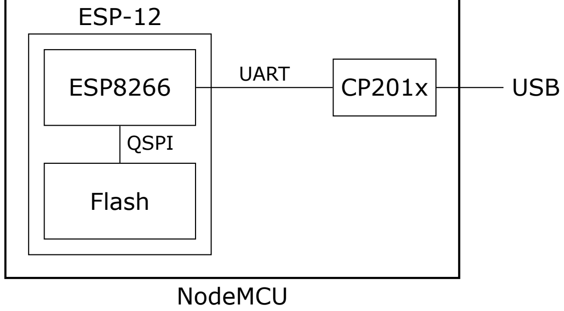

bit
0
7
8
15
16
31
0
version
len
TOS
full length of packet
4
identification
X
D
M
fragment Offset
8
time to live (TTL)
protocol
header checksum
12
source IP address
16
destination IP address
20
IP options (variable length)

Lecture: ESP8266

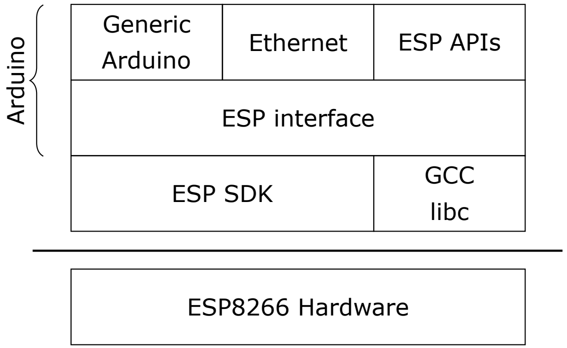
NodeMCU hardware



NodeMCU hardware



ESP8266 software layers



Lecture: MQTT

MQTT

- *Message Queuing Telemetry Transport*
- Wikipedia: MQTT

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 Topic

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

MQTT - The protocol - 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 - The protocol - MQTT Packet

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

MQTT - The protocol - MQTT Topic - more

Enten må den holdes rett etter “## MQTT - The protocol - MQTT Topic” ellers kanskje flyttes etter “patterns”.

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 - 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 - Patterns

Må utvides

Explain:

- Push vs pull, central applications can push to clients
- mostly mqtt, some http
- Client id - sparker ut gamle koblinger
- Keep alive / ping meldinger
- Alternative transporter - websockets(!)

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

Assignments

Assignment 1: Blink a led

Assignment 2: Connect to Wi-Fi

Assignment 3: Connect to MQTT broker

Assignment 4: Network play time

- Measure round trip time/latency. Measure UDP, TCP. Measure when the packet size is greater than the MTU
- Notice variations in RTT