Introduction to IoT

Internet of Things

SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg

Have a Questions?

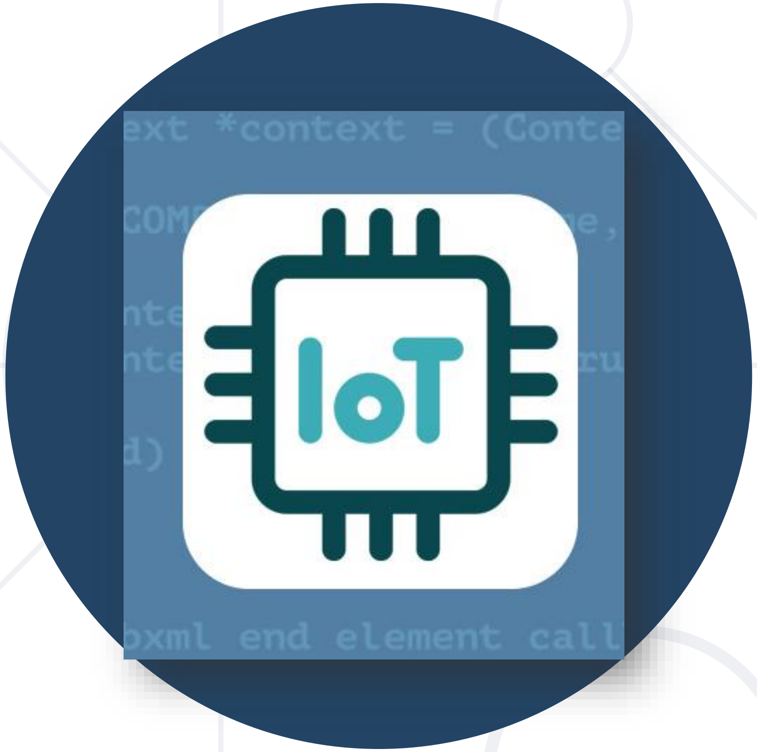
Software University

sli.do

#IoThings

Table of Contents

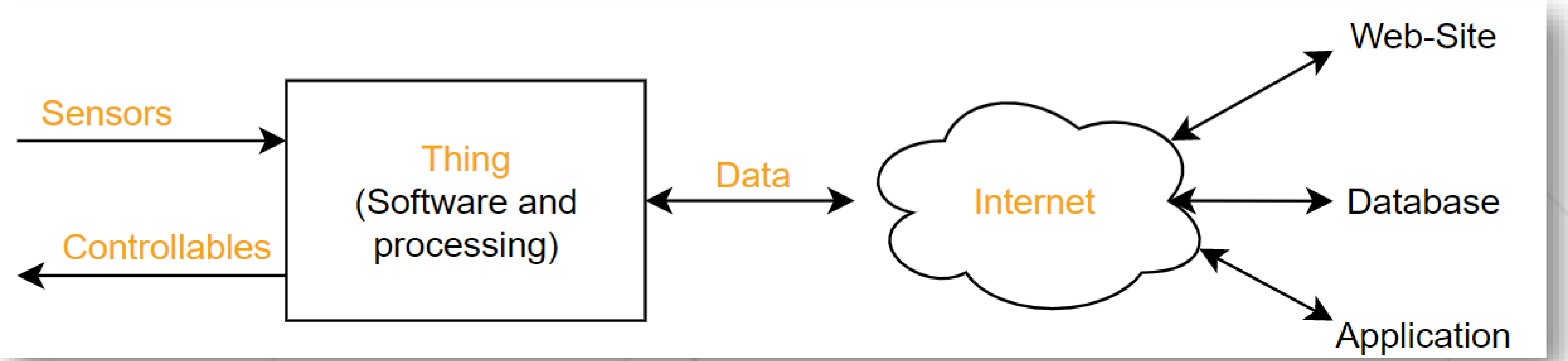
1. What is **IoT**?
2. **Steps** of **Designing IoT Product**
3. **MQTT** vs **Web-Services**



What is IoT?

What is IoT?

- **Definition** - The Internet of Things (IoT) refers to a network of physical objects ("**things**") embedded with **sensors**, and external **controllables** for the purpose of connecting and exchanging **data** with other systems over the **Internet**

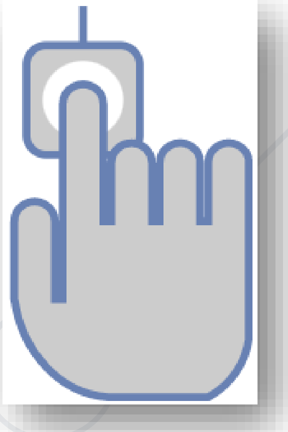


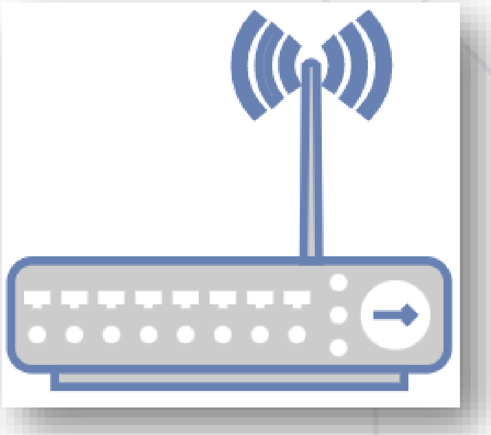
What is IoT?

- **Key Points**:
  - **Connectivity**: IoT devices can connect to the internet and each other, creating a network of interlinked devices
  - **Interaction**: These devices can collect, transmit, and act on data, enabling them to interact with the physical world in real time
  - **Automation**: IoT facilitates the automation of daily tasks, improving efficiency and reducing human intervention

What is IoT?

- **Impact**:
  - The IoT is dramatically transforming various sectors by enabling **smart homes** and **cities**, enhancing **healthcare** and revolutionizing industrial operations with automated processes
- This technology makes environments more **intelligent**, **measurable**, and capable of **sophisticated** interactions





Key Components of IoT

- **Sensors**:
  - Collect data from their environment
  - This could be as simple as a temperature reading or as complex as a full video feed

Key Components of IoT

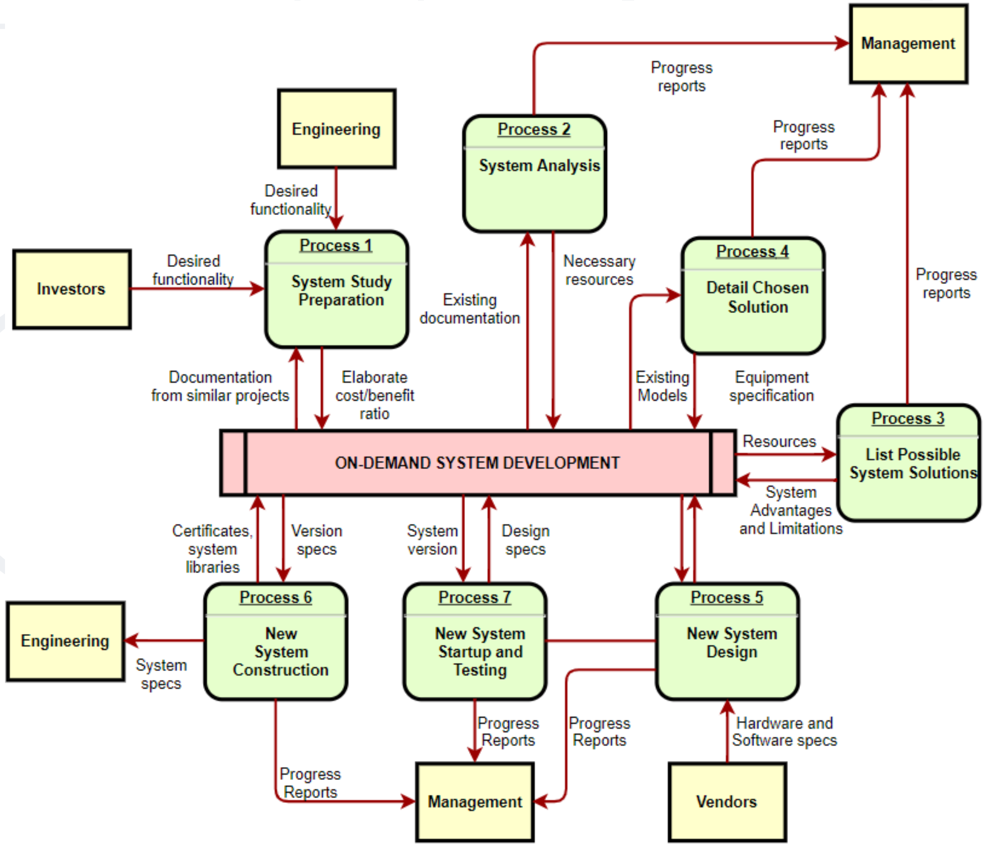
- **Controllables**:
  - Manipulate the environment by regulating or turning on / off peripherals

Key Components of IoT

- **Connectivity**:
  - Devices need to connect to an IoT platform to send and receive data
  - This can be achieved via various forms of network protocols and connectivity options like Wi-Fi, Bluetooth, and cellular networks

Key Components of IoT

- **Data Processing**:
  - Once the data is collected, it needs to be processed either on the device or in the cloud
  - This step involves analysis and decision-making processes



Management

Progress reports

Engineering

Process 2
System Analysis

Progress reports

Desired functionality

Investors

Desired functionality

Process 1
System Study Preparation

Existing documentation

Necessary resources

Process 4
Detail Chosen Solution

Progress reports

Documentation from similar projects

Elaborate cost/benefit ratio

Existing Models

Equipment specification

Process 3
List Possible System Solutions

ON-DEMAND SYSTEM DEVELOPMENT

Resources

System Advantages and Limitations

Certificates, system libraries

Version specs

System version

Design specs

Engineering

System specs

Process 6
New System Construction

Process 7
New System Startup and Testing

Process 5
New System Design

Progress Reports

Progress Reports

Hardware and Software specs

Progress Reports

Management

Vendors

Key Components of IoT

- **User Interface**:
  - The information needs to be available to the end-users in a usable way, which can involve alerts, dashboards, reports, or even automatically triggering actions based on the data

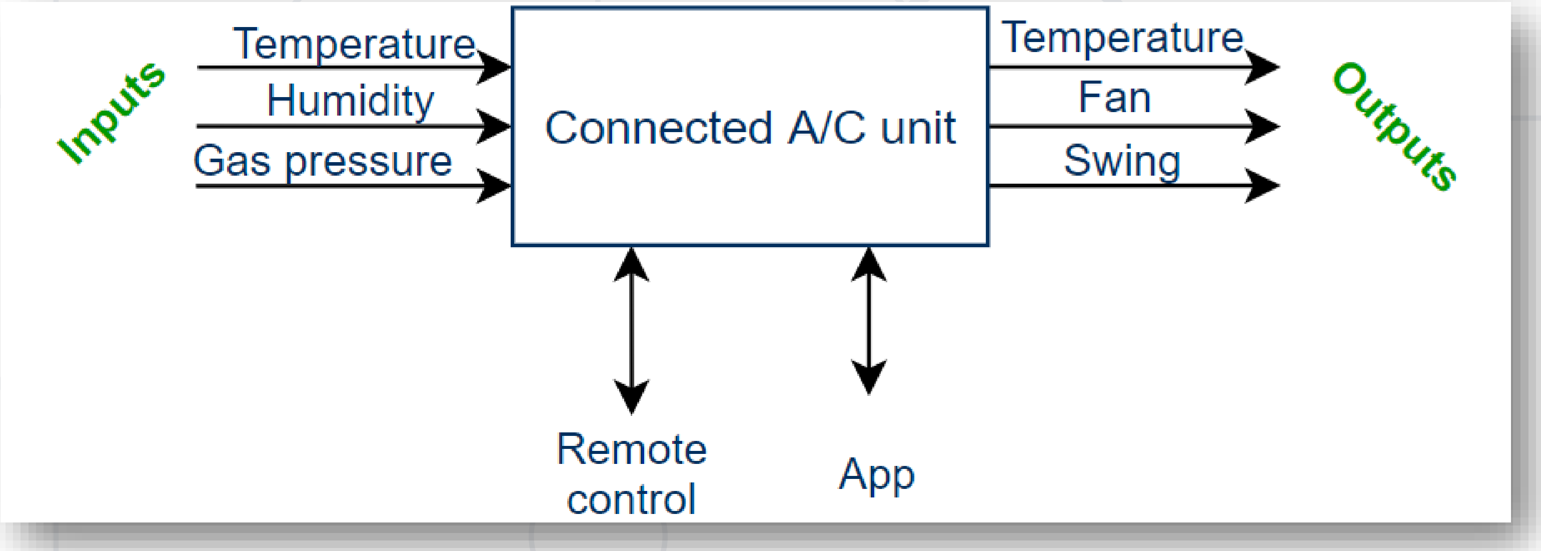
Steps of Designing IoT Product

Designing an IoT System

- Build a **boundary diagram**
- Identify **key parameters** of the system
- Define how the **data flows**
- Pick **connectivity** architecture
- Consider user **installation process**
- Define device **states**
- Start the **implementation**

Designing an IoT System

- Build a **boundary diagram**
  - Identify system **inputs**
  - Identify system **outputs**



Designing an IoT System

- Identify **key** parameters of the system
  - **What** needs to be specified to start looking for a concreate technical solution?

Key Parameters of an IoT System

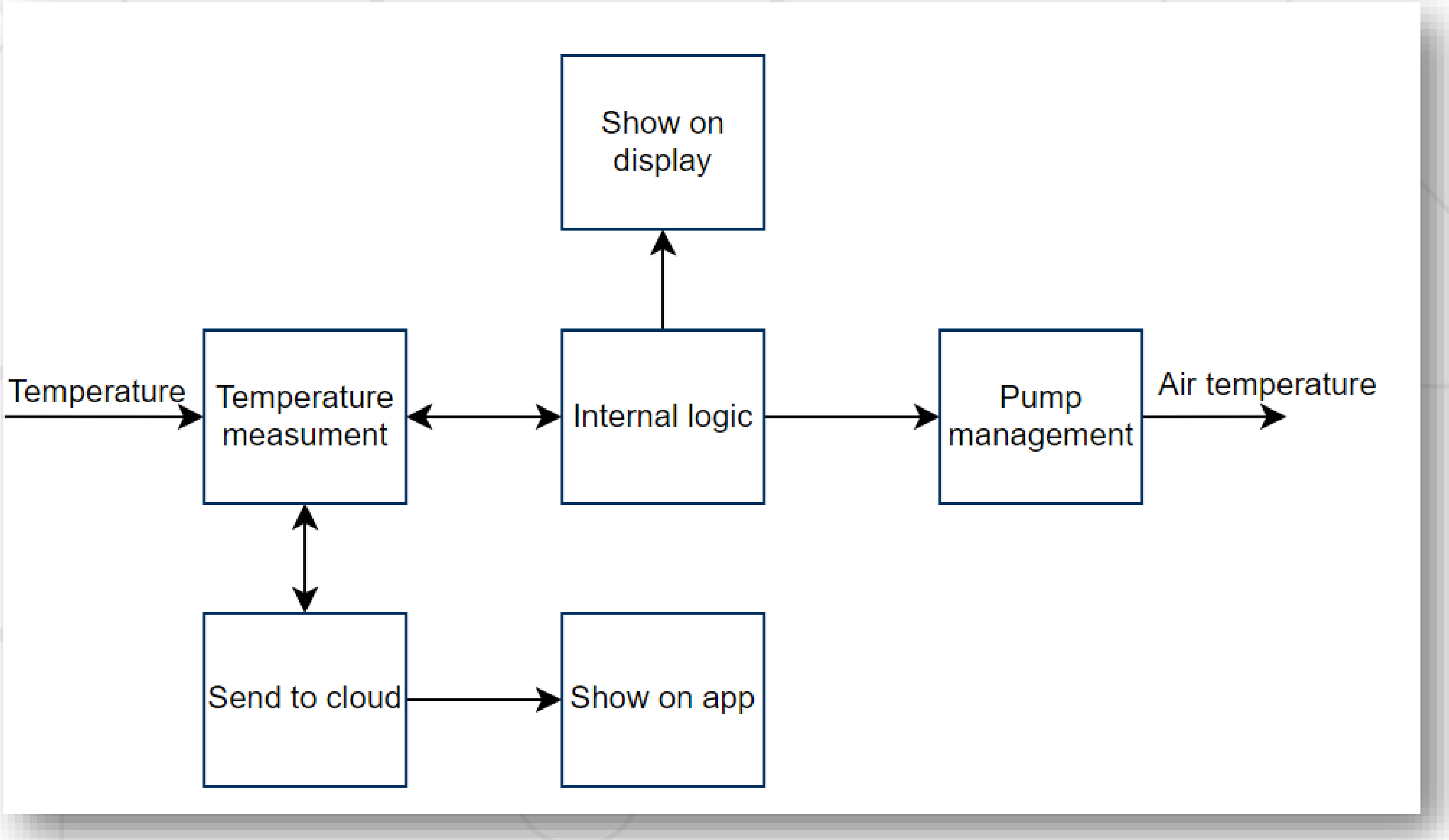
- **Data quality** – inputs and outputs value ranges, precision and consistency
- **Reaction time** – how quick inputs are measured, how quick outputs are affected, how quick the device interprets commands
- **Cost of operation** – Fixed and dynamical costs e.g. server time, data retention cost, electricity cost
- **User experience** – how will the user control the device; sync within several endpoints

Key Parameters of an IoT System

- **Power budget** – device power, battery life
- **Security** – how easy it is to prevent external breaches? What will happen if data breach happens?
- **Reliability** – How to recover from internal fail? How serious a fail would be?
- **Scalability** – How many devices do you want connected? 10? 100? 10k? 100M?

Designing an IoT System

- Define how the **data flows** – list what is affected by each parameter and design the flow



Designing an IoT System

- Pick **connectivity architecture** – connection over **Internet** is needed
  - Many options to consider, which shine in different situations
  - For prototypes, almost all possible architectures are feasible, but for mass production, this is the **steppingstone decision**, which can determine to make it or break it
  - Mostly used **MQTT** and **Web-Services**

MQTT vs Web-Services

MQTT IoT Architecture

- MQTT (**Message Queuing Telemetry Transport**) is a **lightweight** messaging protocol designed for small code footprints and **minimal network bandwidth** usage
- It is based on a **publish** / **subscribe model**, making it highly effective for remote communication in IoT networks
- Developed in 1999 for monitoring oil pipelines over **satellite networks**

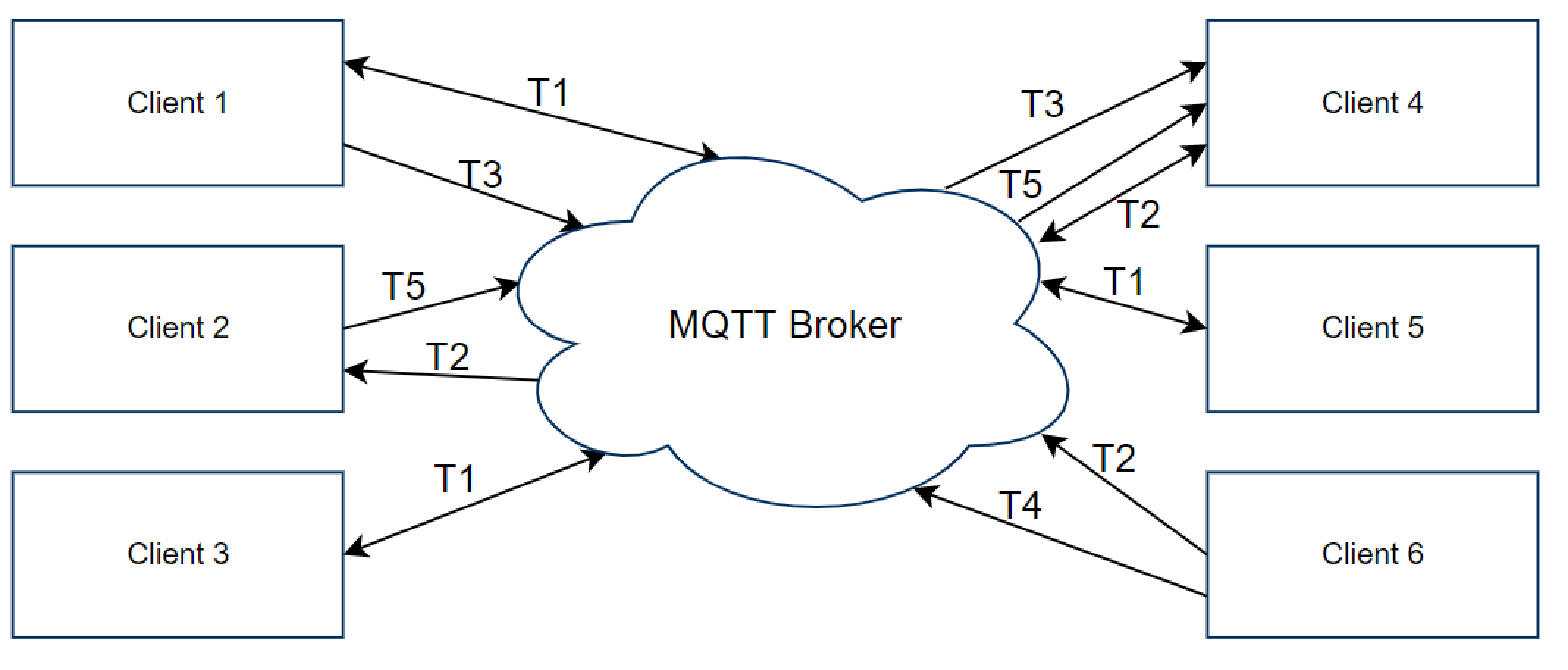
MQTT IoT Architecture

- **Broker** - The **central server** that manages message distribution
  - It receives **all messages from** the clients, filters them, decides who is interested in them, and publishes the message to subscribed clients
- **Client** - Any **device** (like a sensor, smartphone, or computer) that connects to the broker. It can publish messages to the broker and subscribe to topics to receive messages from the broker

MQTT IoT Architecture

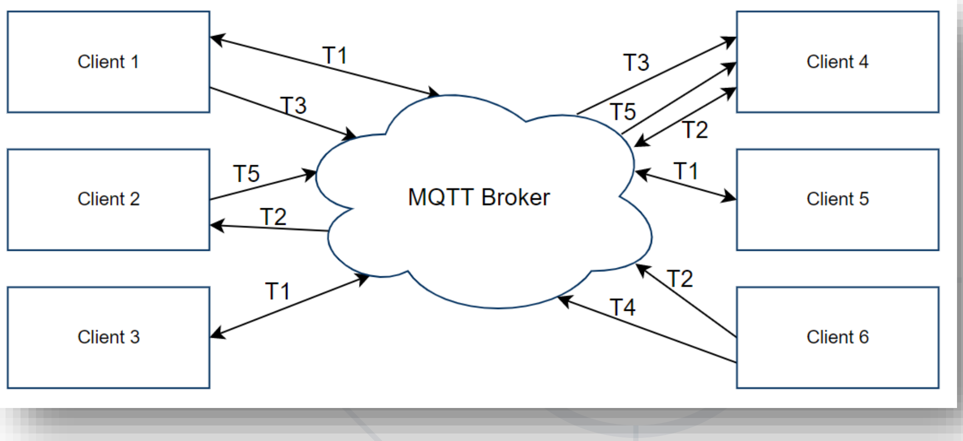
- **Message** - The **data** sent between clients, which can include **commands, information**, or **status updates**
- **Topic** - A string that the broker uses **to filter messages** for each connected client
  - Topics allow a high granularity in message handling, designed hierarchically to facilitate precise filtering

MQTT IoT Architecture



MQTT IoT Architecture

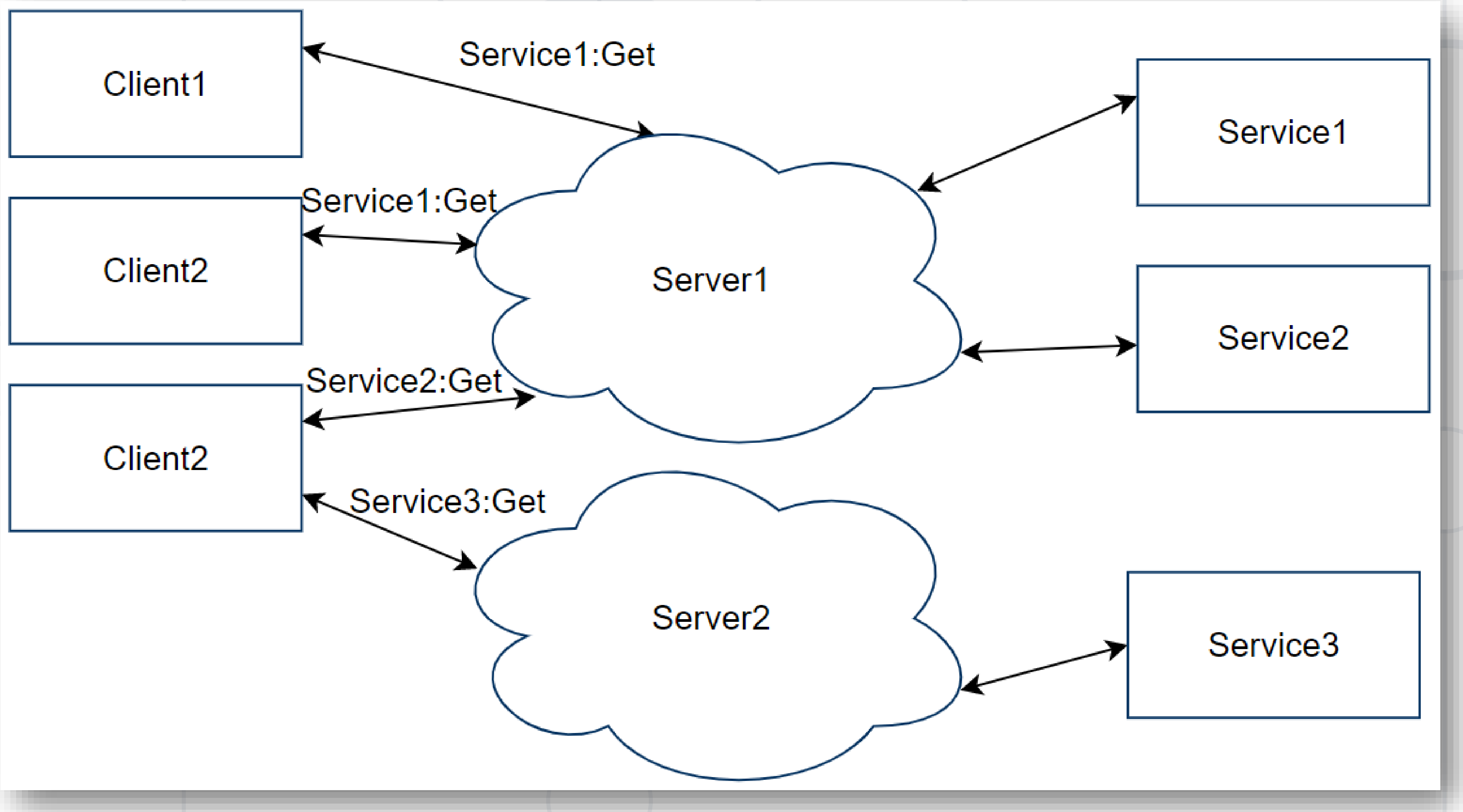
- Client 1 gets data from **T1**
- Client 1 gets data from **T1** and **T3**
- Client 2 gets data from **T2** … and so on
- Clients can be subscribed to unlimited topics
- **Topics** can be used to send and to receive data by the clients
- **Broker** (normally) does not save messages



Web-Services as IoT Architecture

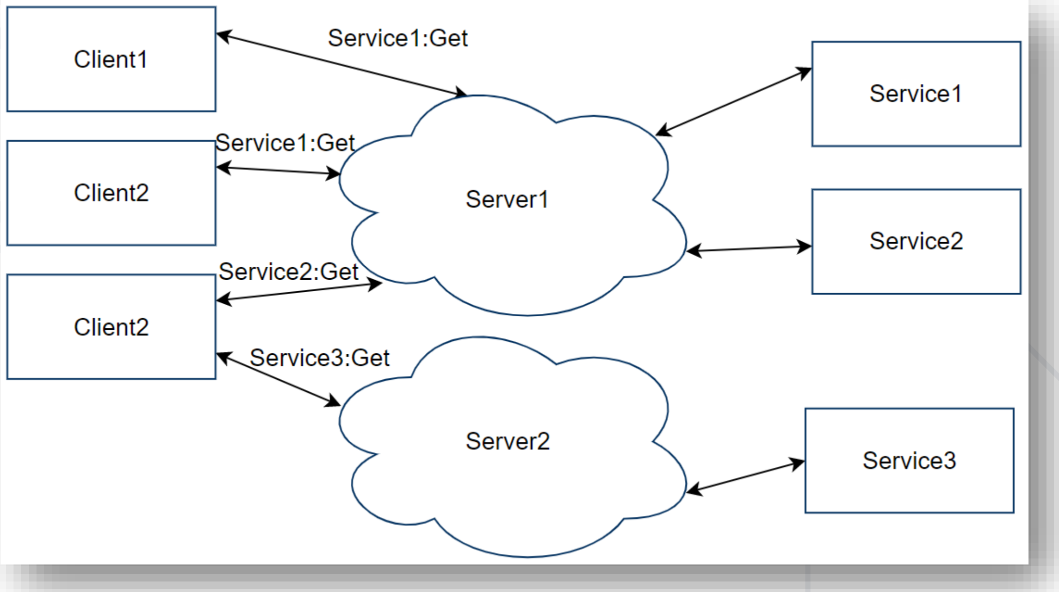
- Enable communication and data exchange across different systems and platforms using **standard web protocols**
- Mainly involves **RESTful** services and **SOAP** (Simple Object Access Protocol) for messaging in a platform-independent way
- Easily integrates with **existing** web technologies, facilitating the management and scalability of IoT applications

Web-Services as IoT Architecture



Web-Services as IoT Architecture

- Multiple servers in the environment
- Each server can have **multiple services**
- Each **client** can connect to **multiple servers**
- Clients know their **result of the request**
- Potential for **Load balancers**



Architecture Comparison vs Key Parameters

	MQTT	Web Services
Data Quality	Optimized for high-frequency, small-size data	Handles large and complex data well
Reaction Time	Very fast, suitable for real-time applications	Generally slower due to HTTP overhead
Cost of Operation	Lower due to minimal data transmission and overhead	Higher due to more data being sent and processed
User Experience	Smooth and efficient, especially in dynamic environments	Can be less responsive, especially wit h complex requests

Architecture Comparison vs Key Parameters

	MQTT	Web Services
Power Budget	Low power usage ideal for battery-operated devices	Higher power usage due to larger data requirements
Security	Provides fundamental levels but requires additional security measures	Often built with comprehensive security features like HTTPS
Reliability	High with Quality-of-Service options for message delivery guarantee	High, provided there is robust network infrastructure
Scalability	Excellent, can handle thousands of connections with low resource usage	Good, but may require more resources as scale increases

Designing an IoT System

- Consider user installation process
  - How can the user connect **his** device to **his** phone?
  - How can the user connect **his** device to **his** network?

QR-codes

App process

Web page

Designing an IoT System

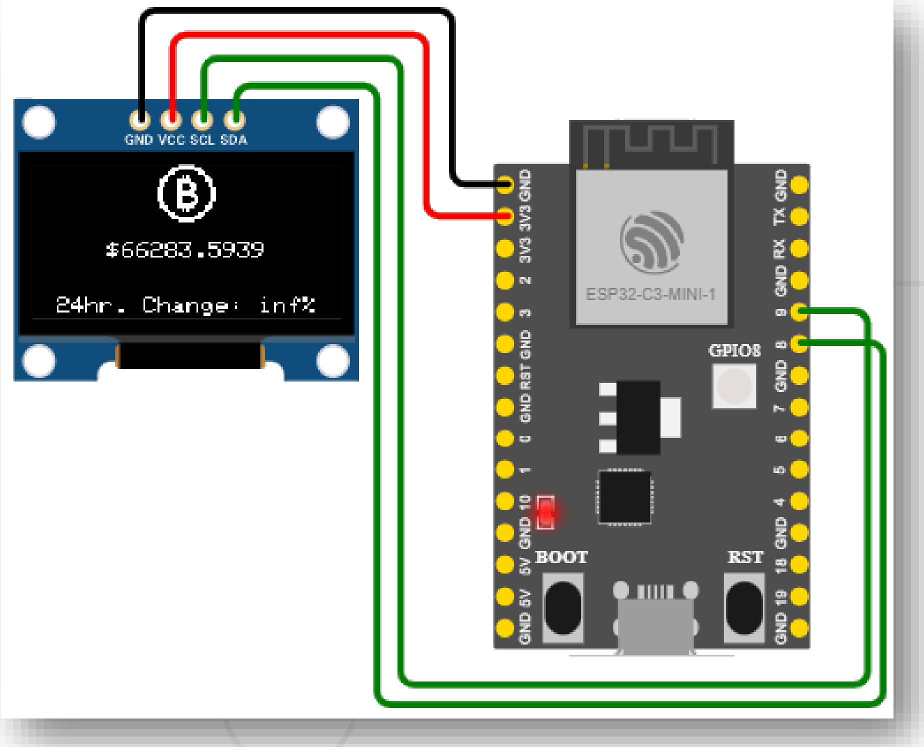
- Define device states – before the implementation think of all the **special cases the IoT device shall cover**
  - What shall the device do if it is just turned on?
  - What shall the device do if connection is interrupted?
  - What shall the device do if you change your router password?
  - Would you update **OTA** (over the air)? What shall the device do during OTA?
  - What would you do with the devices if you change the servers / endpoints?

Designing an IoT System

- Start the implementation
  - Build a physical device with several iterations
    - Wait for **PCB** designers
    - Wait for **production**
    - Something will **burn** during initial prototyping
  - Simulate as much as possible **without physical** prototype

Implement a Thing without a device

- **WokWi** – **online Electronics simulator**
- You can use it to simulate Arduino, ESP32, STM32, and many other popular boards, parts and sensors


GND VCC SCL SDA
$66283.5939
24hr. Change: inf%
ESP32-C3-MINI-1
GPIO8
BOOT
RST
GND 3V3 3V3 2 3 GND RST GND 0 1 10 GND 5V 5V GND
GND TX RX GND 9 8 GND 7 6 5 4 GND 18 19 GND

Why WokWi?

- Start right **now**
- No waiting for components or downloading large software
- Your browser has everything you need to start coding your next **IoT project** in seconds
- Mistakes are **okay**
- You can't destroy the **virtual hardware**
- Trust us, we tried
- So don't worry about frying your **precious components**
- And unlike real hardware, you can always **undo**
- Easy to **get help** and **feedback**

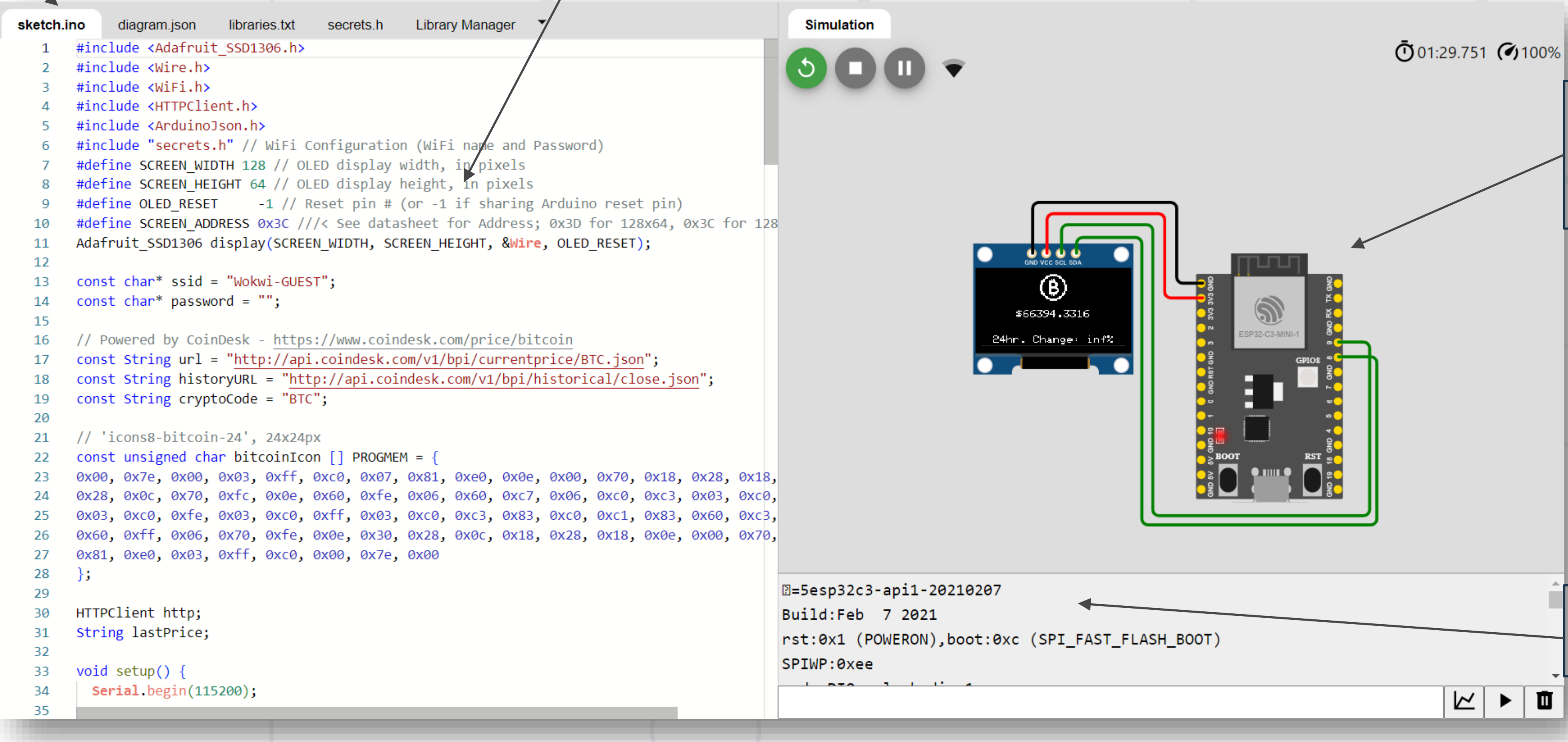
Why WokWi?

- Sharing a link to your **Wokwi** project is all you need
- Gain confidence in your code
- Separate **hardware** and **software** issues
- Unlimited hardware
- No need to scavenge parts from old projects
- Use as many parts as you need, without worrying about project price and stock
- **Maker-friendly** community
- A place for you to share your projects, ask for **help**, and get inspiration
- Wokwi **Discord Community**

Wokwi - Web

All your files

Text editor



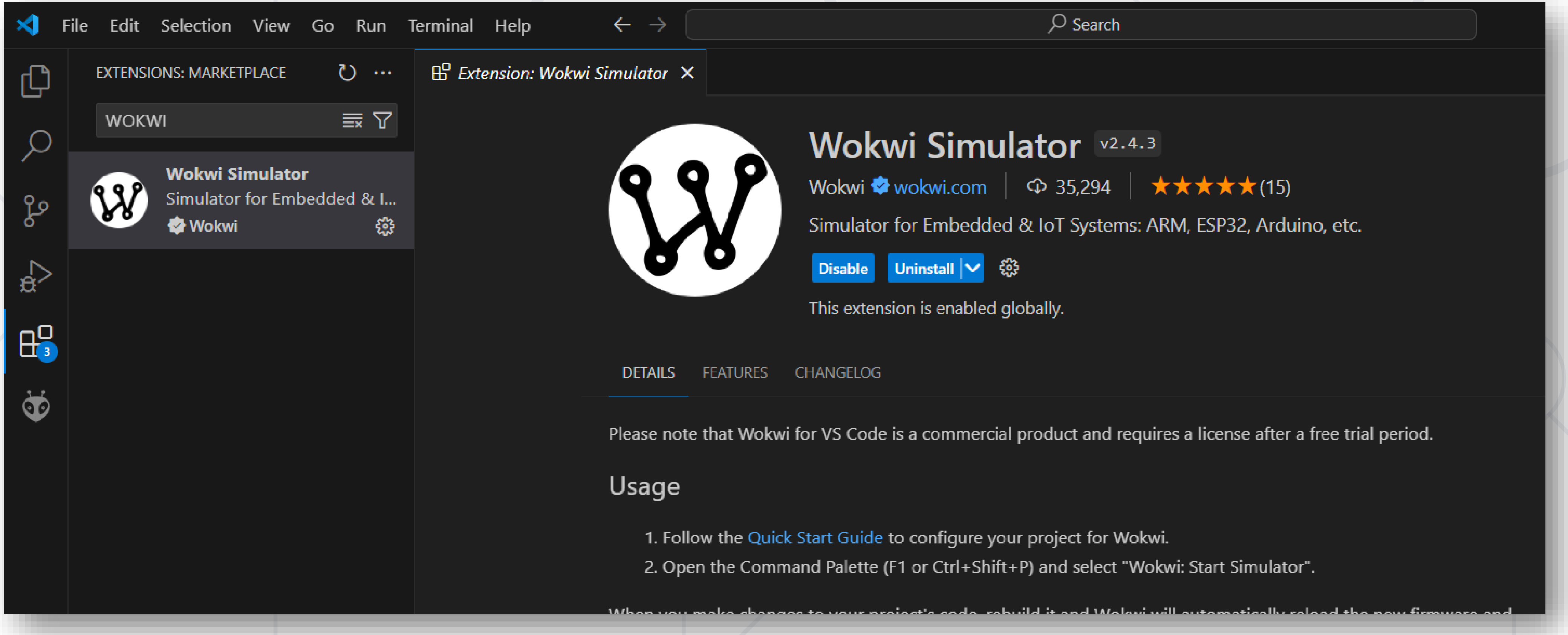
```
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include "secrets.h" // WiFi Configuration (WiFi name and Password)
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

const char* ssid = "Wokwi-GUEST";
const char* password = "";

// Powered by CoinDesk - https://www.coindesk.com/price/bitcoin
const String url = "http://api.coindesk.com/v1/bpi/currentprice/BTC.json";
const String historyURL = "http://api.coindesk.com/v1/bpi/historical/close.json";
const String cryptoCode = "BTC";

// 'icons8-bitcoin-24', 24x24px
const unsigned char bitcoinIcon [] PROGMEM = {
0x00, 0x7e, 0x00, 0x03, 0xff, 0xc0, 0x07, 0x81, 0xe0, 0x0e, 0x00, 0x70, 0x18, 0x28, 0x18,
0x28, 0x0c, 0x70, 0xfc, 0x0e, 0x60, 0xfe, 0x06, 0x60, 0xc7, 0x06, 0xc0, 0xc3, 0x03, 0xc0,
0x03, 0xc0, 0xfe, 0x03, 0xc0, 0xff, 0x03, 0xc0, 0xc3, 0x83, 0xc0, 0xc1, 0x83, 0x60, 0xc3,
0x60, 0xff, 0x06, 0x70, 0xfe, 0x0e, 0x30, 0x28, 0x0c, 0x18, 0x28, 0x18, 0x0e, 0x00, 0x70,
0x81, 0xe0, 0x03, 0xff, 0xc0, 0x00, 0x7e, 0x00
};

HTTPClient http;
String lastPrice;

void setup() {
  Serial.begin(115200);

```

```
⍰=5esp32c3-api1-20210207
Build:Feb  7 2021
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
```

Virtual Breadboard

Console log

WokWi – VS Code



Simulator for Embedded & IoT Systems: ARM, ESP32, Arduino, etc.

This extension is enabled globally.

Please note that Wokwi for VS Code is a commercial product and requires a license after a free trial period.

Usage

1. Follow the Quick Start Guide to configure your project for Wokwi.
2. Open the Command Palette (F1 or Ctrl+Shift+P) and select "Wokwi: Start Simulator".

When you make changes to your project's code, rebuild it and Wokwi will automatically reload the new firmware and

During this Course

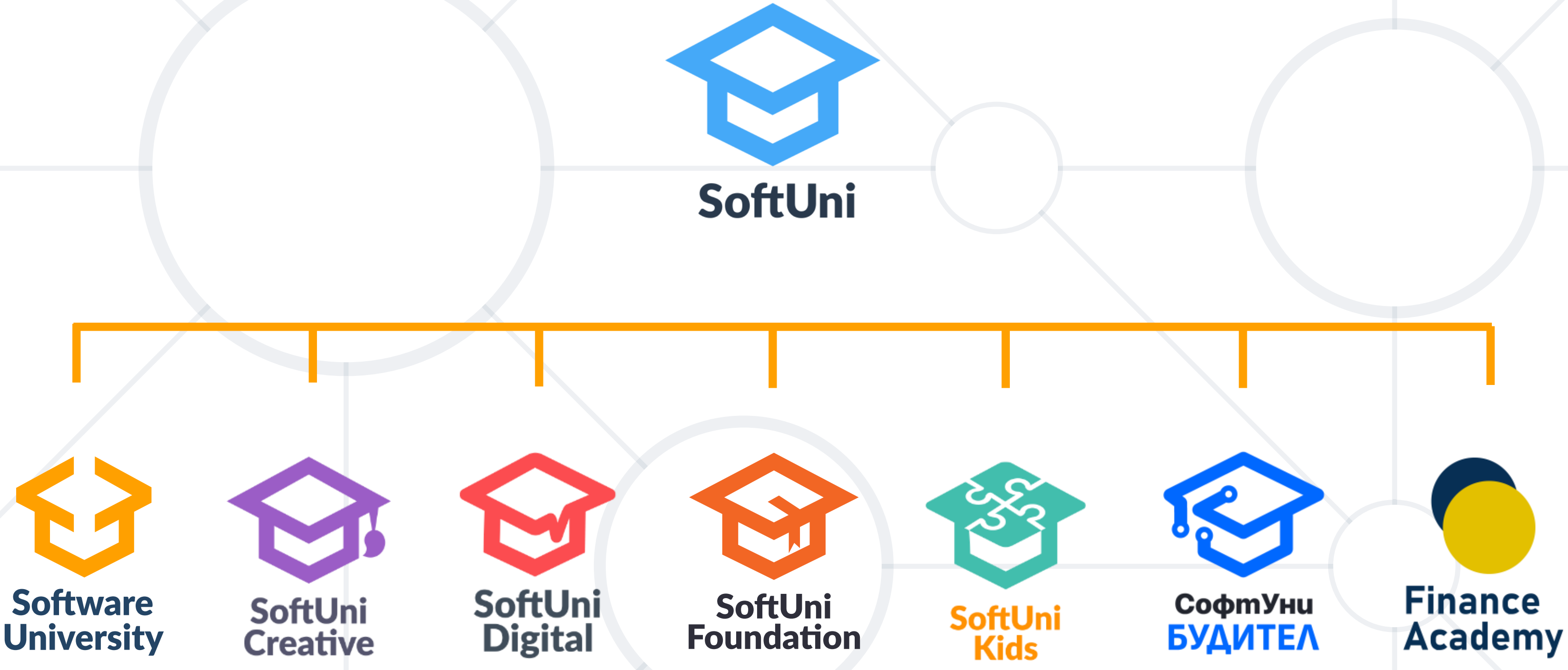
- We are going to use **ESP-32** as main controller
- Any code, that is working in **WokWi** will work in real device – will be shown in later lectures
- Rainmaker for the **end-to-end** communication

Summary

- What is IoT?
- Steps of designing IoT product
- MQTT vs Web-Services as an IoT architecture

Software University

Questions?


SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

SmartIT

createX

VIVACOM

INDEAVR
Serving the high achievers

Coca-Cola HBC
Bulgaria

Postbank
Решения за твоето утре

AMBITIONED

SOFTWARE
GROUP

SUPER
HOSTING
.BG

DRAFT
KINGS

PHAR
VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg