# IoT Crashcourse for Beginners
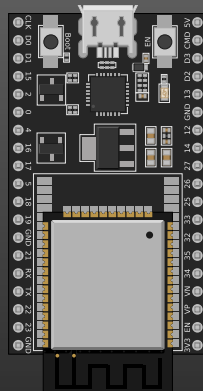
## Med ESP32

Jacob Bechmann Pedersen

September 26, 2023

# Contents

**Figure 1:** ESP32 DevkitC v4, the board we're working with

# Who am I?



Jacob Bechmann Pedersen

- Speaker/Facilitator on Embedded Electronics programming and Arduino workshops
- Embedded electronics engineer at DTU Electro, Automation and Control
  - Robots, embedded Linux, autonomous systems
- Embedded software developer at Oticon
  - Applications for hearing aid OS, unit- and device testing
- Teacher at MakerCamp
  - "Inventors" team - 12-16 y/o
- Volunteer in Coding Pirates 2016-2018
- Electronic Design Engineer (AU, 2019)
- Started with Arduino in 2014

# Purpose

- To understand the basic principles of IoT
  - Topologies
  - Protocols i. e.:
    - HTTP
    - Websockets
    - MQTT
- To program simple implementations
  - On ESP32
  - With the Arduino platform
  - In VSCode

# Resources

Useful links:

- https://github.com/iakop/IoT-Crashcourse
  - Presentation and code for this workshop
- https://code.visualstudio.com/
  - Download for Visual Studio Code
- https://platformio.org/
  - Download for PlatformIO
- https://www.arduino.cc/en/reference
  - Reference on keywords in Arduino
- http://mqtt-explorer.com/
  - MQTT client to explore topics on a broker
- https://nodered.org/
  - Editor based tool for flowbased IoT programming

# Setup of VSCode and PlatformIO

# Setup VSCode



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.
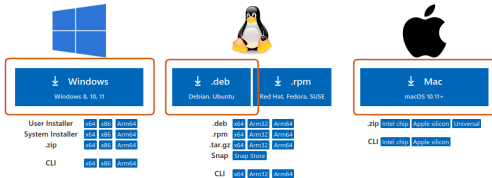
**Figure 2:** The `Visual Studio Code` download page has versions for many different architectures, typically the default button will download the right installer

- Download Visual Studio Code from the link:
  - `https://code.visualstudio.com/Download`
- Click the big button for you OS
- Run the installer, this should go without a hitch
- **IF** that doesn't work, you can try:
  - Windows:
    - If you don't have admin rights, you can download the `User Installer` (typically x64-version)
  - Linux:
    - If you don't use Ubuntu try checking your package managers repositories, or try the `CLI` installer
  - Mac OS X:
    - Try the `Universal .zip`, or maybe the App Store? 👑

# Setup PlatformIO

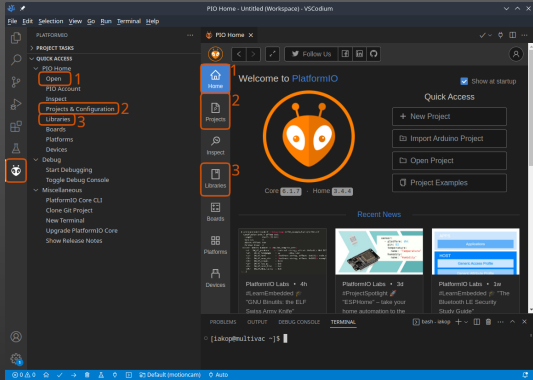- When VSCode is installed and started, go to the Extensions tab
  - Search for PlatformIO
  - Pick the extension pictured, and click install
  - VSCode will automatically install and configure PlatformIO
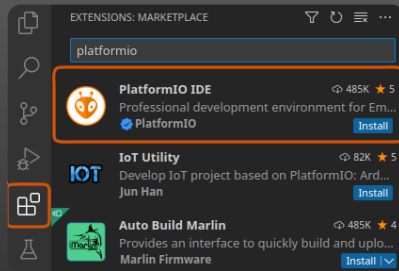


**Figure 3:** Installing PlatformIO in VSCode

# Setup PlatformIO



**Figure 4:** PlatformIO standard view, featuring Home, Projects, Libraries, and Quick Access

- The PlatformIO extension is opened by clicking the PlatformIO tab 🐜
- The most important menu items of the extension:
  1. **Open / Home**
     - Main page of PlatformIO, featuring quick access and tabs for most functions
  2. **Projects & Configuration / Projects**
     - Project management page, for creating and managing projects
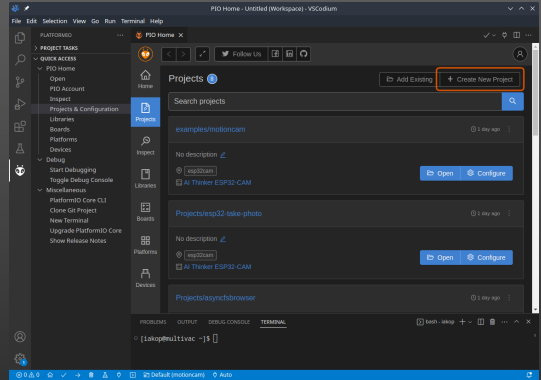  3. **Libraries**
     - For searching and adding Libraries to PlatformIO projects

**Setup of ESP32 project in PlatformIO**

# Setup of ESP32 project in PlatformIO



- Click the tab `Projects` to enter the projects view
- To create a new project, click the button `Create New Project`

**Figure 5:** `Projects` tab in PlatformIO. For creating and managing projects within the GUI
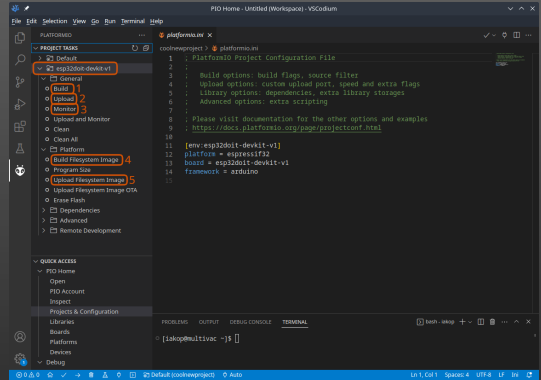
# Setup of ESP32 project in PlatformIO



**Figure 6:** The `Project Wizard` dialog in PlatformIO, with settings for name, board and framework

- A `Project Wizard` dialog will be opened
- It contains 3 fields, to be filled as follows:
  - Name
    - A fitting name for the project, e.g. "`coolnewproject`"
  - Board
    - `DOIT ESP32 DEVKIT V1`
  - Framework
    - `Arduino`
- Finish by clicking `Finish`

# Setup of ESP32 project in PlatformIO

- PlatformIO will generate the project and set up the toolchain
  - This requires an internet connection
- When done, load `platformio.ini`
  - This file contains the project settings, and can be edited by hand
- On the side of the window there are `Project Tasks`:
  1. **Build**
     - Build an image for the device to be flashed
  2. **Upload**
     - Uploads the image through an automatically detected USB/UART connection
  3. **Monitor**
     - Monitors the UART connection to the hardware (Baud rate can be set in `platformio.ini`)
  4. **Build Filesystem Image**
     - Builds file system image for the hardware (based on the contents of the `data` folder of the project)
     - `data` folder needs to be created manually
     - File system can be specified in `platformio.ini`
  5. **Upload Filesystem Image**
     - Uploads the built image to the hardware
     - `IMPORTANT`: Monitor can not be active during upload



**Figure 7:** Opened project in PlatformIO, shows `platformio.ini` and the `Project Tasks` for the project
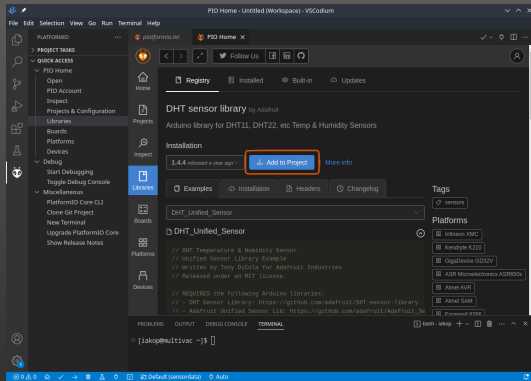
# Add libraries to an ESP32 project
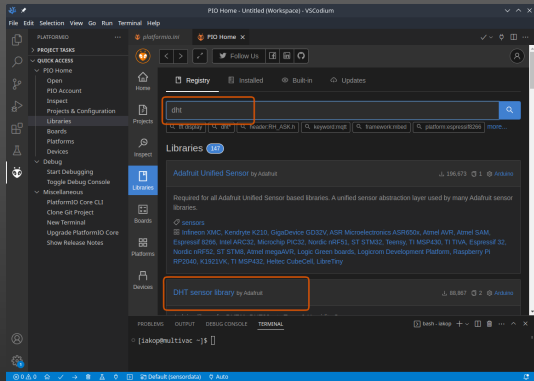


**Figure 8:** The `Libraries` tab in PlatformIO. For searching and adding libraries to projects

- To add external libraries to a project, use the `Libraries` tab for finding contributed libraries
  - Can be found under `Registry`
  - Installed libraries can be viewed underr `Installed`
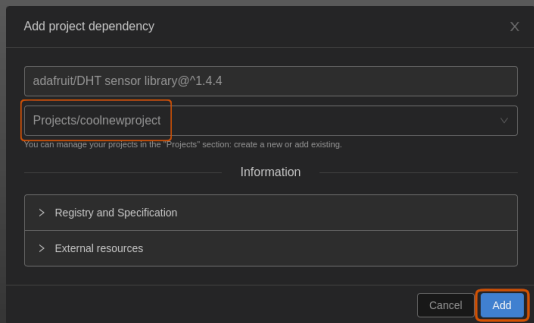- Click a relevant library

# Add libraries to an ESP32 project

- The following can be found within the library:
  - Examples
  - Headers
  - etc.

- Click Add to Project to add the library to a project



**Figure 9:** DHT sensor library in PlatformIO. Can be added to projects that support the Arduino framework

# Add libraries to an ESP32 project



**Figure 10:** The `Add project dependency` dialog in PlatformIO. To pick which library to add the library to

- The `Add project dependency` dialog will open
- Under `Select a project`, pick the project to add the library to
- Click `Add`
- PlatformIO will automatically add a `lib_deps` dependency within `platformio.ini`, and set up the library

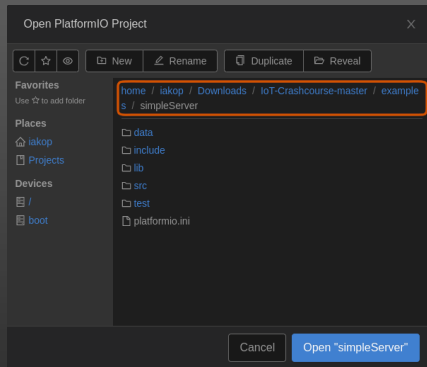# Opening external ESP32 Projekt in PlatformIO
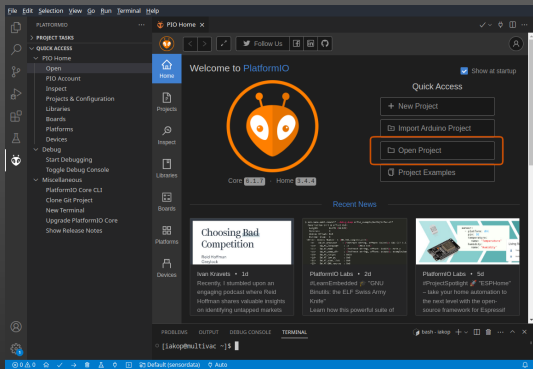


**Figure 11:** Home tab in PlatformIO, button to load project from disk is highlighted

- The projects for this workshop use specific libraries and settings
- To get them quickly set up, the projects can be downloadet and imported from the Github repo
- Download the entire workshops materials here:
  - https://github.com/iakop/ IoT-Crashcourse/archive/ refs/heads/master.zip
- Extract them somewhere easy to locate
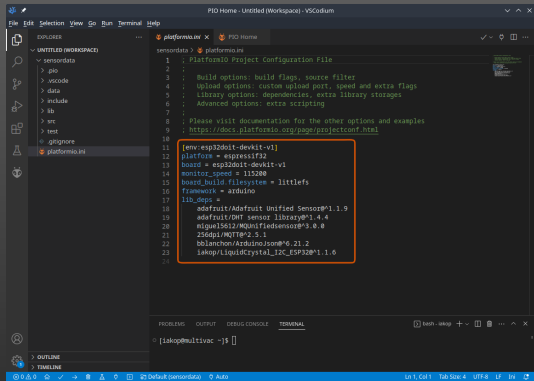- Under the Home tab, click Open Project

# Opening external ESP32 Projekt in PlatformIO

- In the `Open PlatformIO Project` dialog, open the examples folder for the workshop

- If the `Open` button, for example, shows `Open "simpleServer"` the dialog is in the correct folder

- Click the `Open` button



**Figure 12:** `Open PlatformIO Project` dialog in PlatformIO. To open a project the folder needs to be extracted and located on the disk, for example, in: `Downloads/IoT-Crashcourse-master/examples/simpleServer`

# Opening external ESP32 Projekt in PlatformIO



**Figure 13:** Example of a `platformio.ini` for a project. Pay attention to `monitor-speed`, `filesystem` and `lib_deps` that are pre-defined

- When the project is loaded, open the `platformio.ini`
- Specifies libraries and components that the project depends on
- Tools and settings will be set up autimatically by PlatformIO
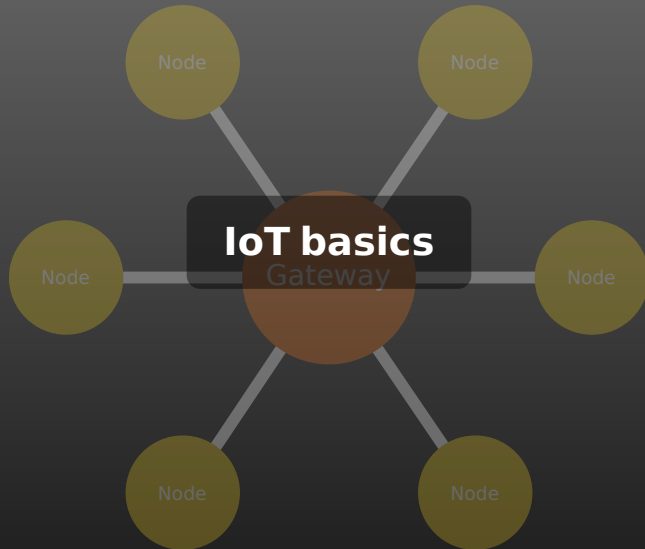
**IoT basics**

# IoT basics

- IoT (Internet of Things), is a common name for networked devices
- These devices typically consist of:
  - A microprocessor or -computer
  - Sensors
  - Actuators
  - Wired or wireless connectivity



**Figure 14:** Nedis SmartLife torn down to show the insides. Contains a TYWE3S WiFi module and an HLW8012 power sensor
**Kilde:** https://callaa.github.io/2021/01/26/liberating-nedis-smartplug.html

# IoT basics



**Figure 15:** Star-topology, where every device communicates through a central gateway to the rest of the internet



**Figure 16:** Tree-topology, Where the devices are connected in branches, where they heirarchically relay information to the gateway



**Figure 17:** Mesh-topology, where devices communicate internally, relaying information through eachother to the gateway

- Communication between devices can be done in several ways
- Some typical IoT toplogies:
  - Star
  - Tree
  - Mesh

# IoT basics

- There are also several protocols for devices to communicate
- In this workshop we focus on:
  - HTTP
    - The ubiquitous Hypertext Transfer Protocol, for transferring web content, e.g. between servers and browsers
  - WebSocket
    - A full duplex (two-way communication) protocol for quick, simultaneous communication between client and server - low overhead
  - MQTT
    - (Originally acronym for MQ (Message Queue) Telemetry Transport) Publish-subscribe based protocol between devices and a central broker - low overhead



**Figure 18:** HTTP logo
**Kilde:** https://en.wikipedia.org/wiki/File:HTTP_logo.svg
**Licens:** Public Domain



**Figure 19:** WebSocket logo
**Kilde:** https://logodix.com/logos/1825947
**Licens:** Non-Commercial



**Figure 20:** MQTT logo
**Kilde:** https://en.wikipedia.org/wiki/File:Mqtt-hor.svg
**Licens:** Public Domain

**Build a simple ESP32 webserver**

# Simple Server

- For this example we need a breadboard setup
  - An ESP32 board
  - An LED
  - A 220Ω resistor
- HTML and the Arduino program will be presented and explained on the board
- Source code can be found on:
  - https://github.com/iakop/IoT-Crashcourse/tree/master/examples/simpleServer
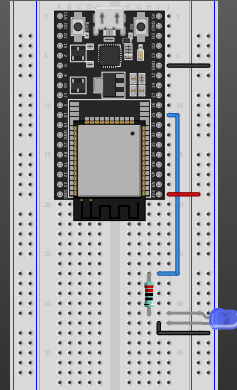


**Figure 21:** Breadboard setup with ESP32 and LED

# WebSockets on ESP32

# WebSocket Server

- This example adds a sensor to the setup
  - An ESP32 board
  - An LED
  - A 220Ω resistor
  - A DHT11 temperature/humidity sensor module
- We'll add Javascript and a WebSocket connection, which we'll also cover on the board
- Source code can be found on:
  - https://github.com/iakop/IoT-Crashcourse/tree/master/examples/websocketServer
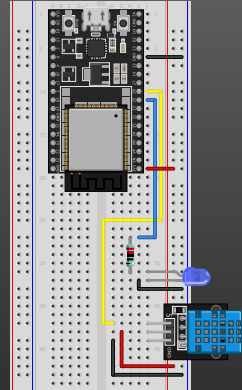


**Figure 22:** Breadboard setup with ESP32, LED and DHT11 sensor

MQTT on ESP32

# MQTT Client

- Same setup
  - An ESP32 board
  - An LED
  - A 220Ω resistor
  - A DHT11 temperature/humidity sensor module
- All server code is exchanged for client code, connecting through SSL to an MQTT broker
- Source code can be found on:
  - https://github.com/iakop/IoT-Crashcourse/tree/master/examples/mqttClient
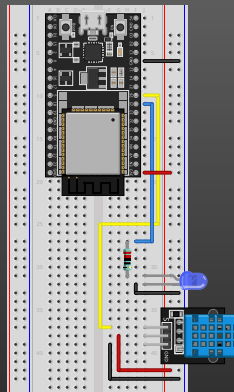


**Figure 23:** Breadboard setup with ESP32, LED and DHT11 sensor

# MQTT Client



**Figure 24:** MQTT Explorer Connection dialog window, with the settings for connecting to mqtt.bechmann.xyz

- MQTT Explorer can be used to check and explore topics on a broker:
  - http://mqtt-explorer.com/
- Settings for public server for this workshop:
  - Name: Bechmann (optional)
  - Validate certificate: off
    - Bug in MQTT Explorers cert storage prevents validating Let's Encrypt RootCA
  - Encryption (tls): on
  - Protocol: mqtt://
  - Host: mqtt.bechmann.xyz
  - Port: 8883
  - Username: blank
  - Password: blank