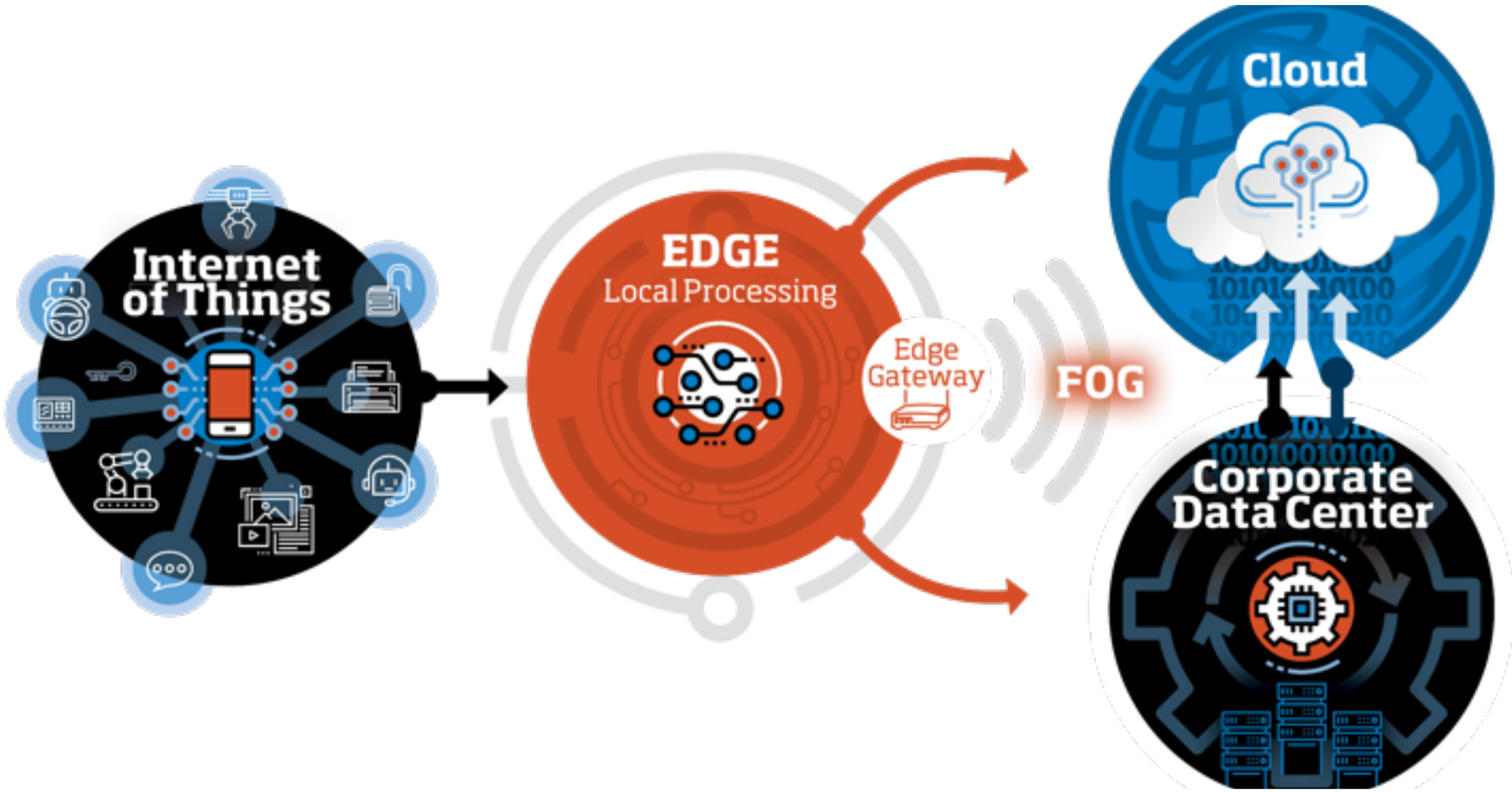
IoT Lab

Michael Gerndt, Mohak Chadha, Isaac Nunez
Technische Universität München

IoT Components



Goals

- Familiarization with IoT Cloud concepts
- Implement a concrete use case
- Deploy machine learning technologies

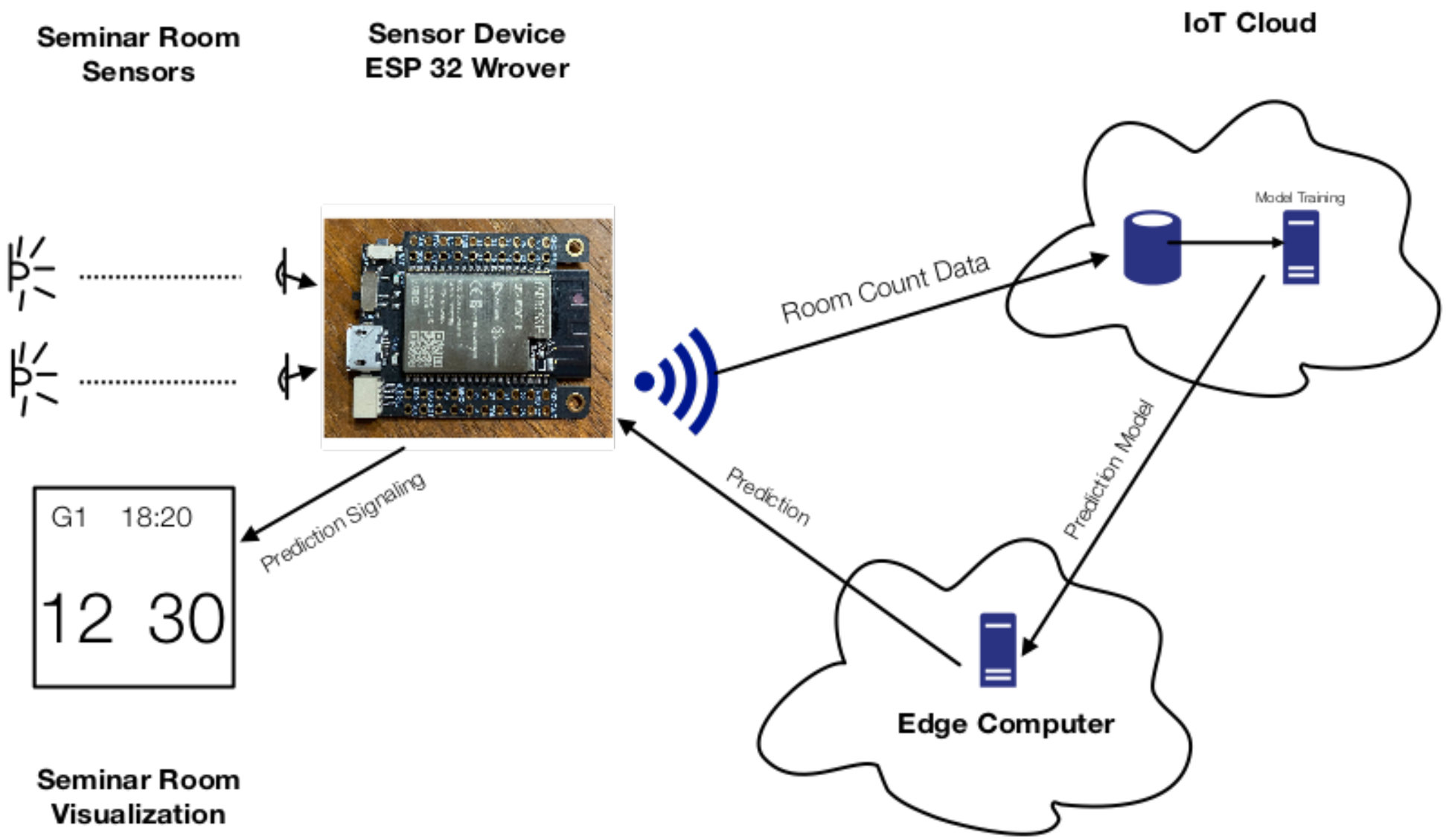
Use Case

Seminar Room Monitoring

Features

Room Occupancy Prediction

Final Architecture



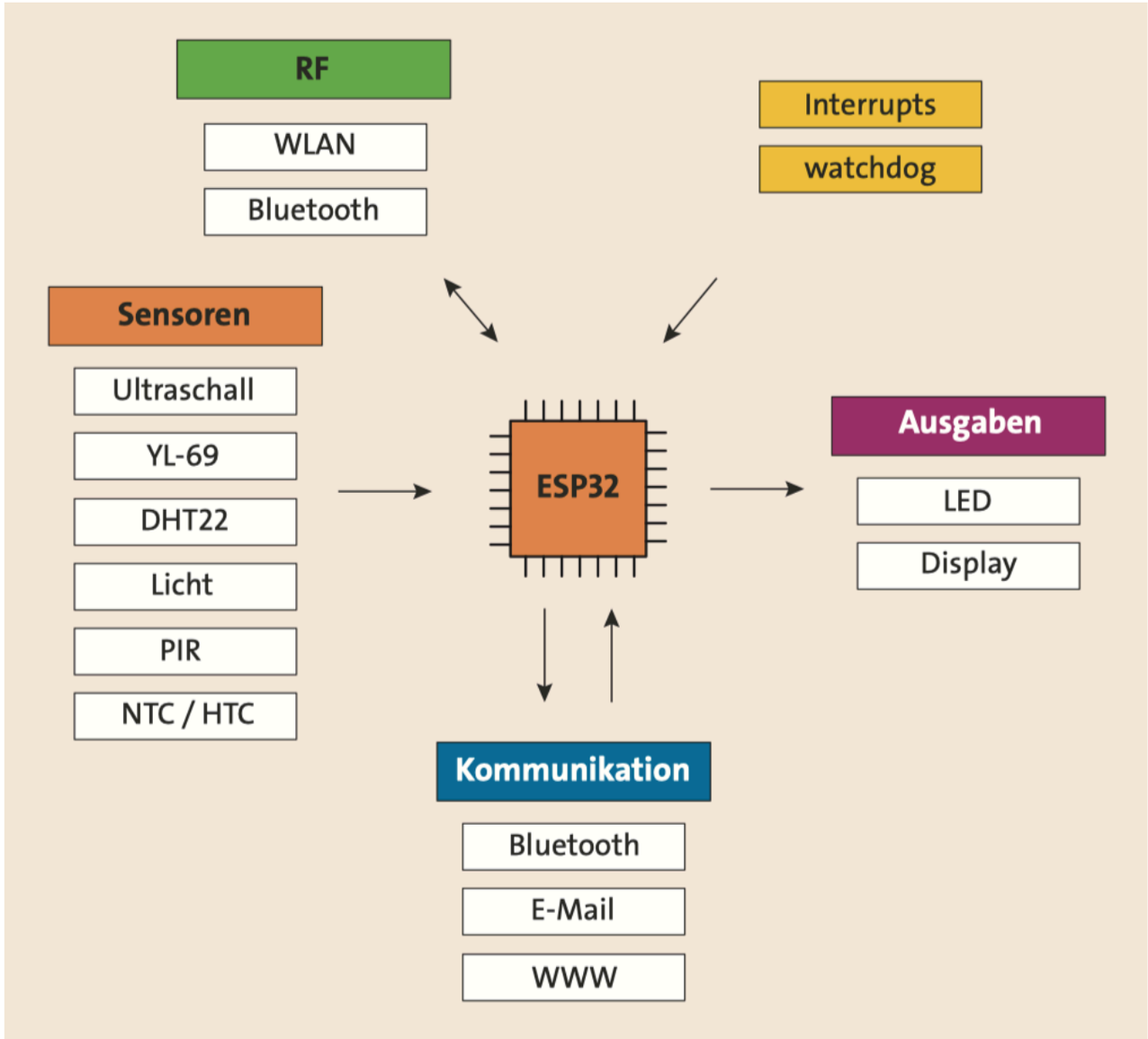
Main Technologies

- Microcontroller ESP32 with WLAN and Bluetooth
- Photoelectric barrier for sensing
- IoT platform provided by previous lab courses
- Data modeling and prediction algorithms
- Cloud for edge VM

Microcontroller vs Microprocessor

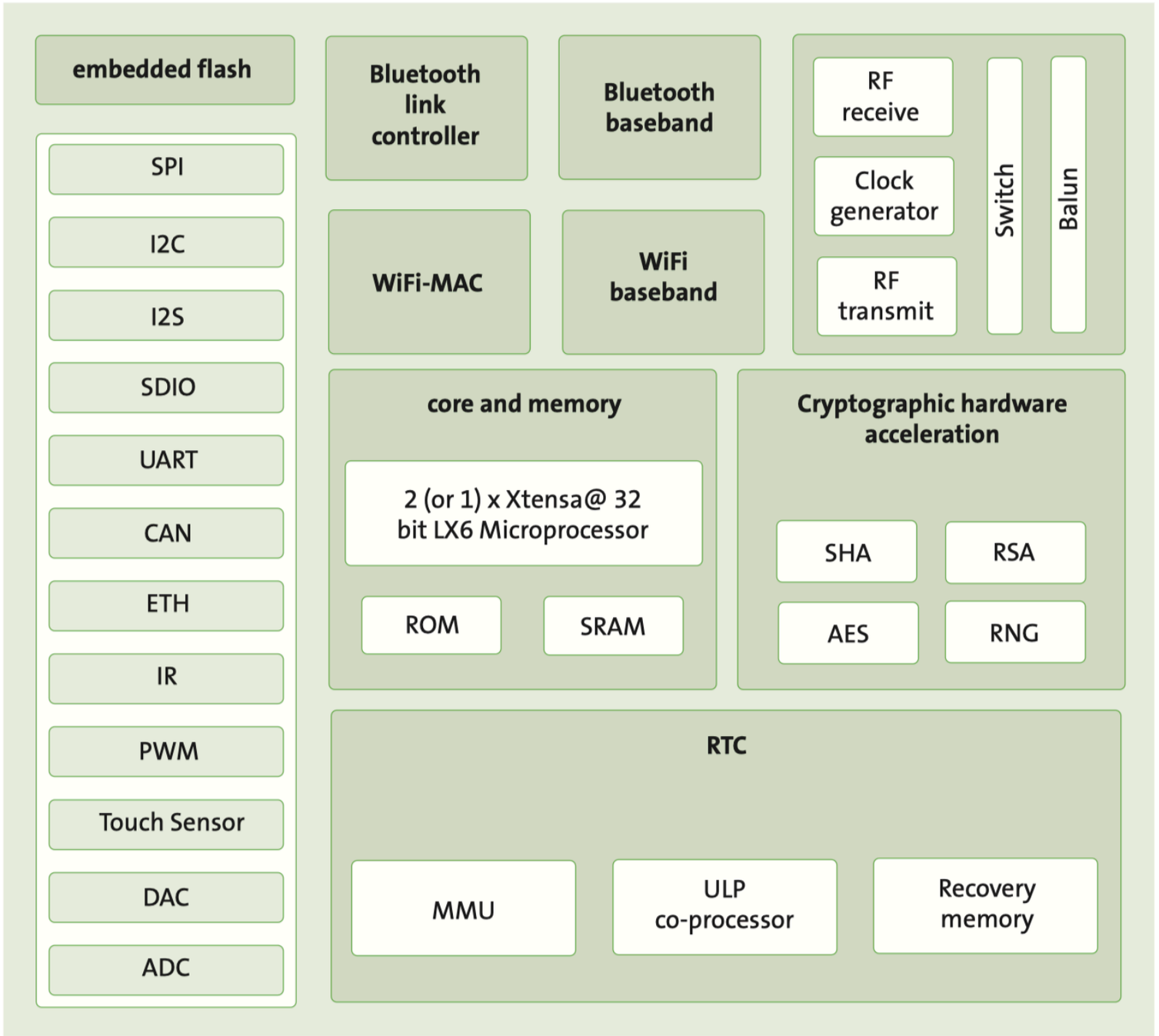
- Microprocessor
  - Processor with external memory
- Microcontroller
  - Processor with integrated memory
  - Commonly flash + SRAM
  - A lot of I/O interfaces

Espressif ESP32



https://www.espressif.com/en/products/socs/esp32

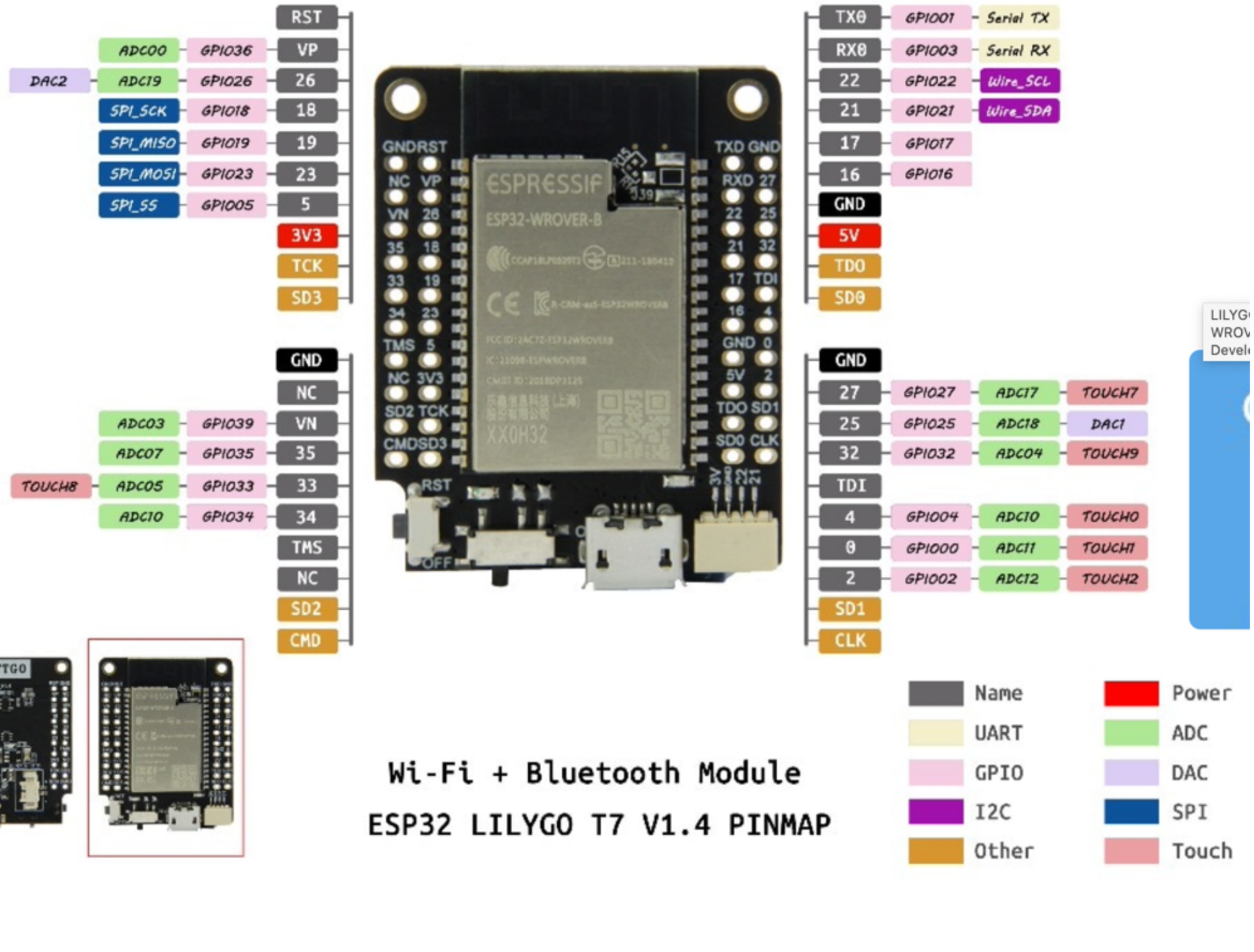
ESP32 Block Diagram



ESP32 Memory

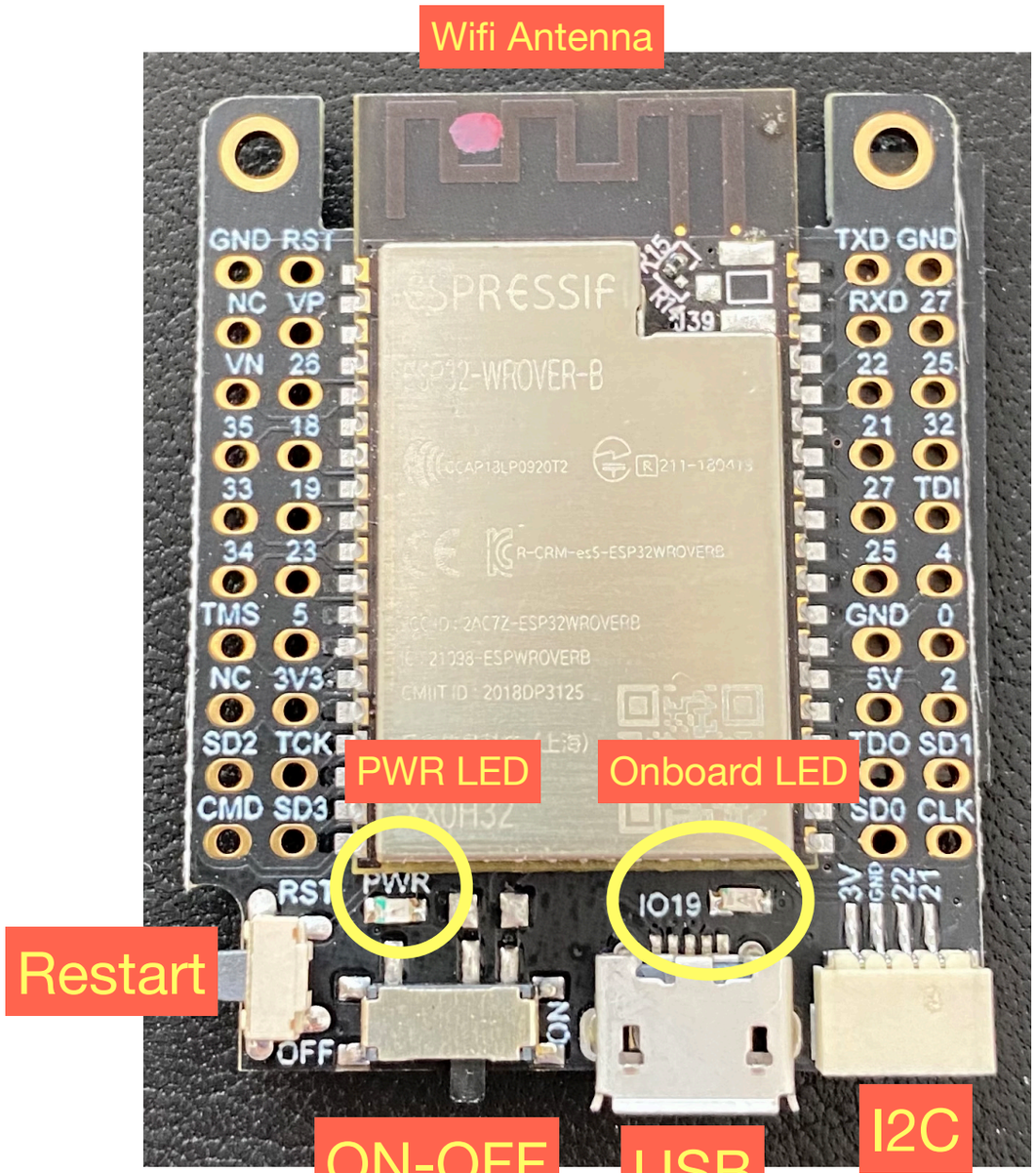
- 520 KB SRAM for data and instructions
- 448 KB ROM for boot and kernel functions
- 8 KB SRAM in RTC (RTC-FAST Memory) safe data over deep sleep
- 8 KB SRAM in RTC (RTC-SLOW Memory) for coprocessor during deep sleep
- ESP Wrover-B
  - 4 MB QSPI Flash memory
  - Additional 8 MB PSRAM (Pseudo Static RAM)

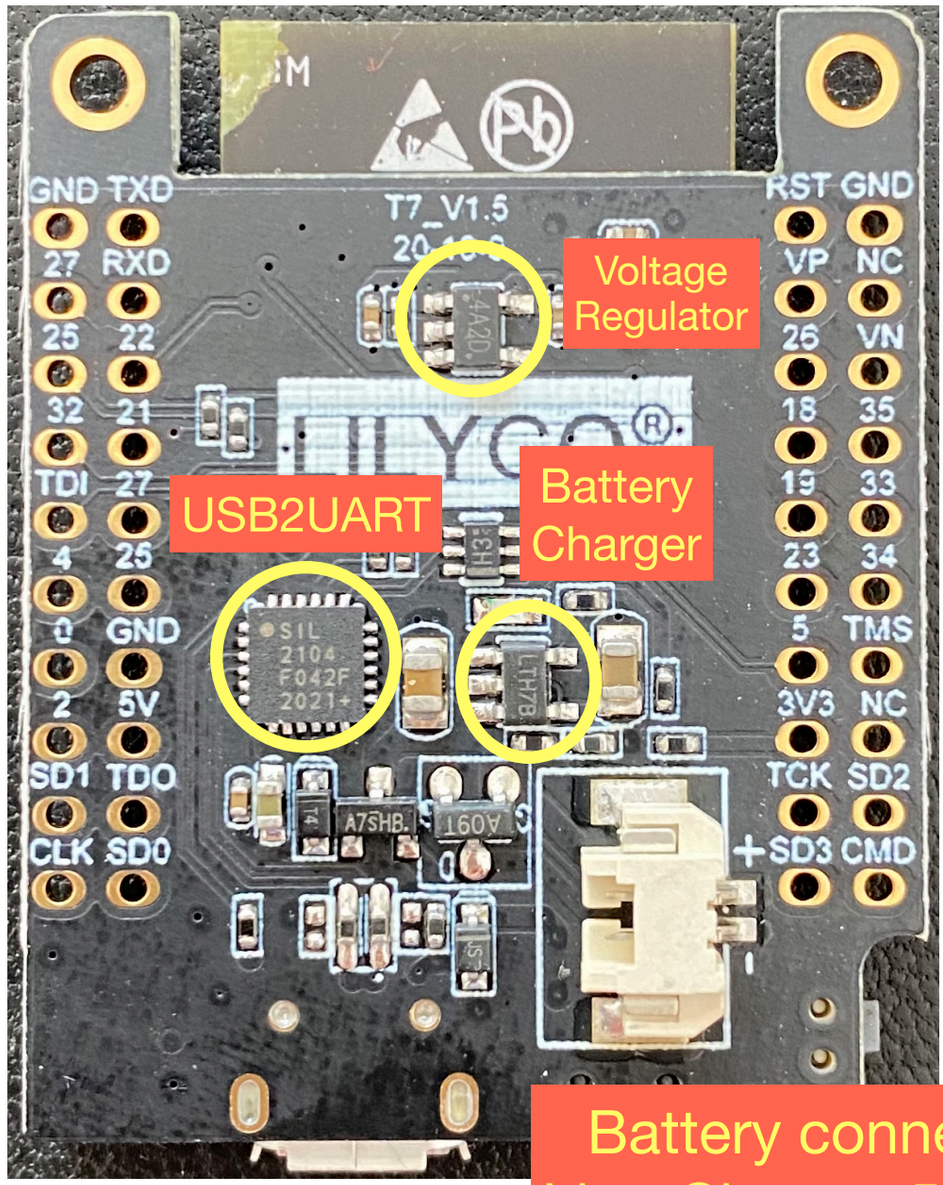
LILYGO Wrover-B



http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=978

LILYGO Wrover-B Board





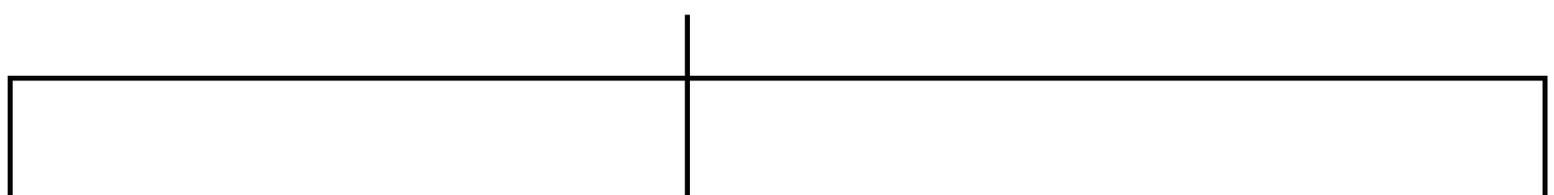
ESP IDF

- Several development environments available
  - **ESP IDF**
  - Arduino
  - MS Visual Studio Code
  - platform.io
- Espressif IoT Development Framework (ESP-IDF)
  - Provided by Espressif: https://docs.espressif.com/projects/esp-idf/
  - Installation instructions for Windows, Linux, Mac OS
  - Combines
    - Compiler
    - Build environment
    - freeRTOS
    - Components: MQTT, Camera, ...

Application Structure

Project directory

CMakeLists.txt
README.md
sdkconfig



main

CMakeLists.txt
.c source files
.h header files
Kconfig.projbuild

components

Subdirectories for libraries. They have own CMakeLists.txt

build

Directory constructed from cmake for building.

Project Building

- Python script idf.py
  - build: Triggers configuration and building
  - flash: Triggers linking and flashing to board
  - monitor: Triggers serial connection to board for monitoring
  - fullclean: clean the whole generated files
- idf.py is based on cmake and make

FreeRTOS

- ESP IDF is based on the Free Real Time Operating System (FreeRTOS)
  - freertos.org
  - Maintenance taken over by Amazon from Real Time Engineers ltd.
    - Integration of sensors with the Greengrass edge runtime
  - It is free: No need to publish your code if developed with FreeRTOS, no fees
- Basically a runtime system linked to the application
  - Managing resources: CPU, memory, timers, IO
  - Supports soft and hard realtime requirements

FreeRTOS

- Terminology
  - What is usually called a **thread** is a **task** in FreeRTOS
- Provides
  - Multitasking: time slicing with task preemption
    - The task scheduler is the heart of FreeRTOS
    - Soft and hard realtime constraints are supported through task prioritization
  - Synchronization: locks, semaphores, ...
  - Software timers: countdown clocks triggering an attached callback function
  - HEAP management
- Write embedded software without a kernel
  - For simple applications may be easier: Arduino style (setup and loop routine)
  - In case of multiple tasks to be run with a certain frequency it becomes complex. The frequency has to be implemented in the code of the loop, e.g., display task, read sensors, handle input from Wifi ...

ESP IDF and FreeRTOS

- ESP IDF provides a port of FreeRTOS for ESP32
- Extended version for multicore system
  - Two application cores
  - Low power coprocessor

Hardware

You last purchased this item on 21. February 2020.
Number Of Items: 1 | View this order

Roll over image to zoom in

AZDelivery OLED Parent 128 x 64 Pixel 0.96 Inches

Visit the AZDelivery Store
4.7 out of 5 stars 1,288 ratings | 28 answered questions
Amazon's Choice for "az-delivery"

Price: €5.79
Prices for items sold by Amazon include VAT. Depending on your delivery address, VAT may vary at Checkout. For other items, please see details.

Promotion Message €5 on Arduino book with microcontroller. 1 Promotion(s)

New (3) from €5.79 & FREE Shipping on orders over €29.00

Number Of Items: 1

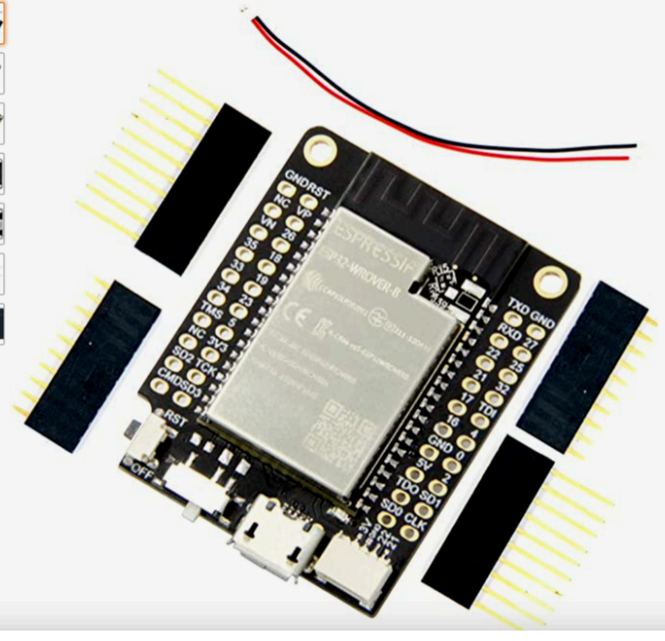
1	3	5	25
€5.79	€9.99 (€3.33 / Item)	€17.99 (€3.60 / Item)	€84.99 (€3.40 / Item)

Brand	AZDelivery
Computer memory size	1 MB
RAM type	EEPROM

About this item

- Strong Contrast the using the innovative OLED technology in the big 0.96 inch display.
- Due to high resolution of 128 Mal 64 Pixels provides space for easy displaying the screen elements.
- Easier connect the displays with Arduino, Raspberry Pi and co. using the I2 °C interface via only 4 pins.
- Libraries for Arduino Raspberry Pi and Co. thanks to standard controller (SSD 1306) already available, was programming a piece of cake.
- Usually high quality and fast delivery thanks to when buying Azdelivery. Also free: Getting Started know-how of Azdelivery experts in eBook format.

You last purchased this item on 24. February 2021.
View this order



ICQUANZX Mini32 V1.3 ESP32-WROVER-B PSRAM Wi-Fi Bluetooth Module Development Board

Brand: ICQUANZX
4.2 out of 5 stars 13 ratings

Price: €16.99
Prices for items sold by Amazon include VAT. Depending on your delivery address, VAT may vary at Checkout. For other items, please see details.

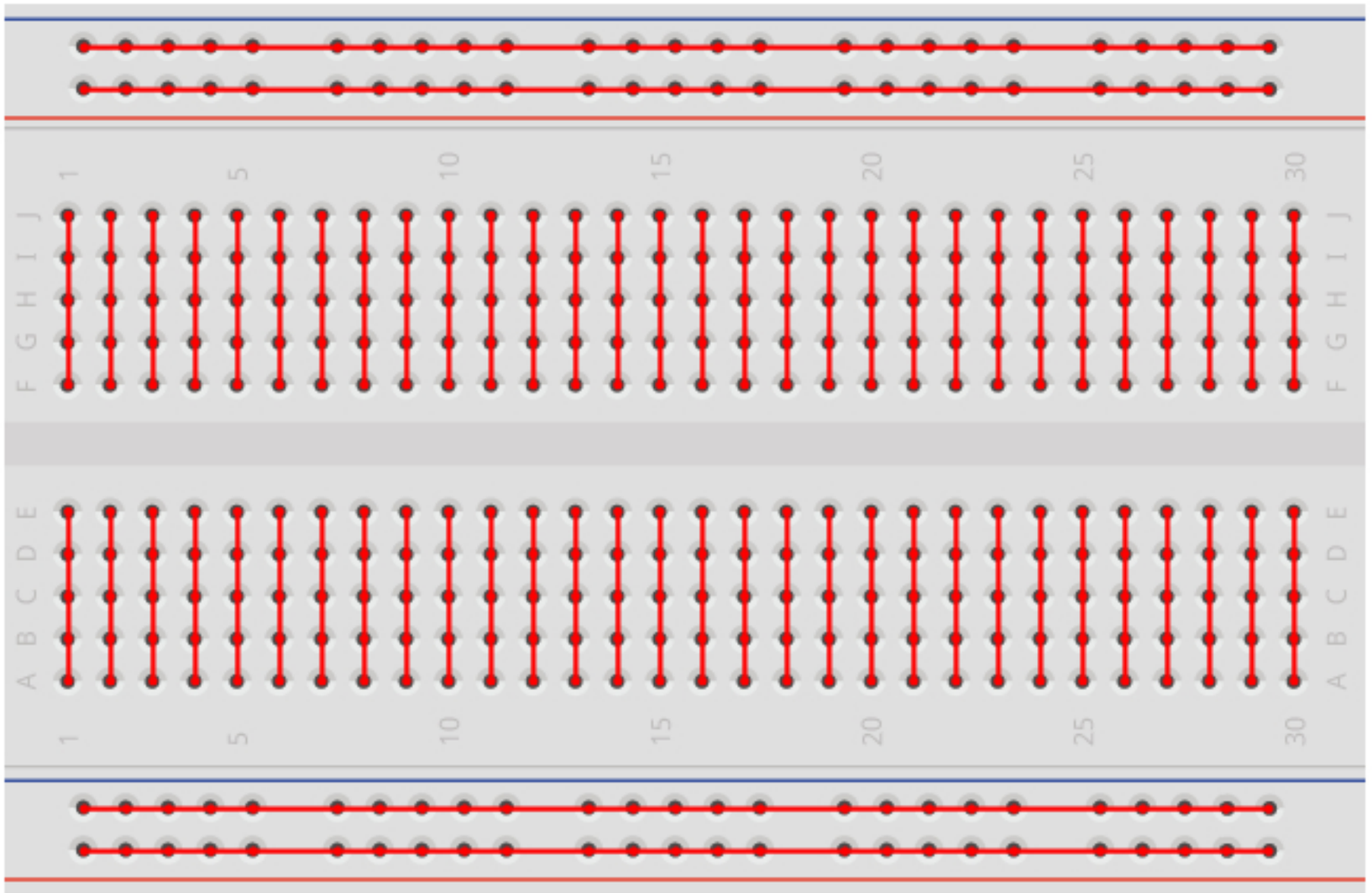
- Chipset: ESPRESSIF-ESP32 (Wi-Fi & Bluetooth) 240MHz Xtensa Single / Dual Core 32-bit LX6 microprocessor
- FLASH: QSPI-Flash 4 MB / PSRAM 8 MB
- Working current: over 30mA, sleep current: over 10uA.
- JST connection: 2-pin, 1.25 mm, 4-pin, 1.0 mm
- Modular Interface: UART, SPI, SDIO, I2C, LED PWM, TV PWM I2S, IRGPIO, ADC, capacitor touch sensor, DACLNA preamplifier FLASH: QSPI-Flash 4MB / PSRAM 8MB

Report incorrect product information.

Want to recycle your electrical or electronic appliance for free? (Do it here.)

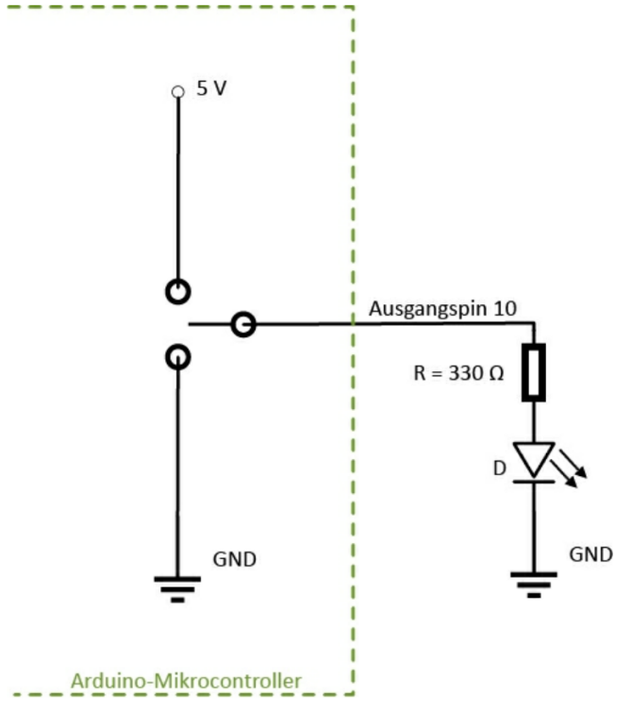
Hardware

- Breadboard

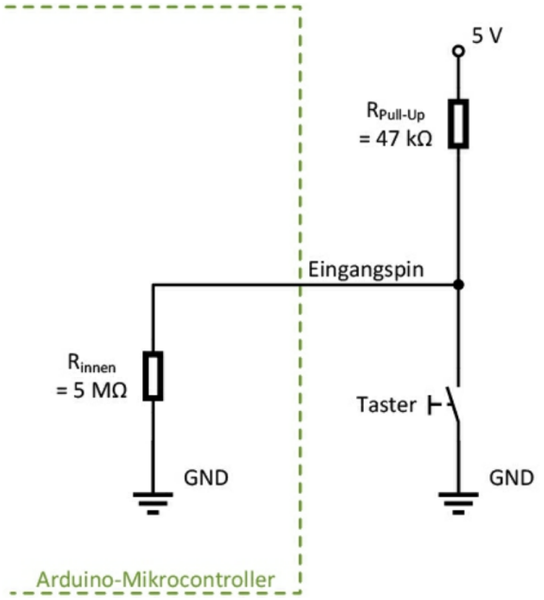


I/O Pins

- Digital Outputs
  - Switches pin between LOW and HIGH
  - Be careful of maximum current
  - E.g. trigger an LED



- Digital Inputs
  - Sense LOW and HIGH
  - Be careful that input is always defined.
  - E.g. button



GPIO Handling

```
esp_err_t gpio_set_level(gpio_num_t gpio_num, uint32_t level)
// level is 0 or 1

int gpio_get_level(gpio_num_t gpio_num)

esp_err_t gpio_set_direction(gpio_num_t gpio_num, gpio_mode_t mode)
//GPIO_MODE_INPUT, GPIO_MODE_OUTPUT
//pins 34 - 39 can only be used for input

esp_err_t gpio_pullup_en   (gpio_num_t gpio_num)
esp_err_t gpio_pullup_dis  (gpio_num_t gpio_num)
esp_err_t gpio_pulldown_en (gpio_num_t gpio_num)
esp_err_t gpio_pulldown_dis(gpio_num_t gpio_num)

//pins 34 - 39 do not have software pullup/pulldown resistors
```

Visual Studio

- Visual Studio is an integrated development environment on Windows, Linux, Mac
- Microsoft
- Market place with extensions with tools for various programming languages
- Espressif ESP32 extension
- Configuration files in .vscode
  - *settings.json*: paths, ESP32 port, baud rate, ...
  - *c_cpp_properties.json*: settings for intellisense
  - *launch.json*: settings for debugging
  - *tasks.json*: define tasks that can be run from the menu
- Blink source code comes with configuration files for visual studio
  - replace user specific paths in settings.json

Blink Example

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "sdkconfig.h"
#include "esp_log.h"

#define BLINK_GPIO CONFIG_BLINK_GPIO

static const char *TAG = "BLINK";

void app_main(void){
    //esp_log_level_set("BLINK", ESP_LOG_ERROR);
    esp_log_level_set("BLINK", ESP_LOG_INFO);

    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);

    while(1) {
        /* Blink on (output high) */
        ESP_LOGI(TAG, "Turning on the LED");
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        /* Blink off (output low) */
        ESP_LOGI(TAG, "Turning off the LED");
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Demo Blink Basic

1. cd into root directory of the example
2. ls /dev/cu.usb-*
3. export ESPPORT=/dev/cu.usbserial-0215EAEE
4. idf.py menuconfig: on board led pin 19
5. idf.py build
6. idf.py flash
7. idf.py monitor
8. Exit monitor ctrl-]
9. idf.py menuconfig: led pin 32
10. idf.py build flash monitor
11. ctrl-] to close the monitoring

Assignment for next week

1. Install ESP IDF V4.4
2. Download blink sources from moodle. It is a modified version of the blink example .../esp-idf/examples/get-started/blink
3. Try on board LED and external LED with resistor
4. Develop a version that polls a button. Press button->LED on
   - Configure GPIO with pullup or pulldown resistor
5. Make the length and frequency configurable through idf.py menuconfig

Group Building

Schedule

Next Meeting

- Bring you hardware and demonstrate the blink example

Questions