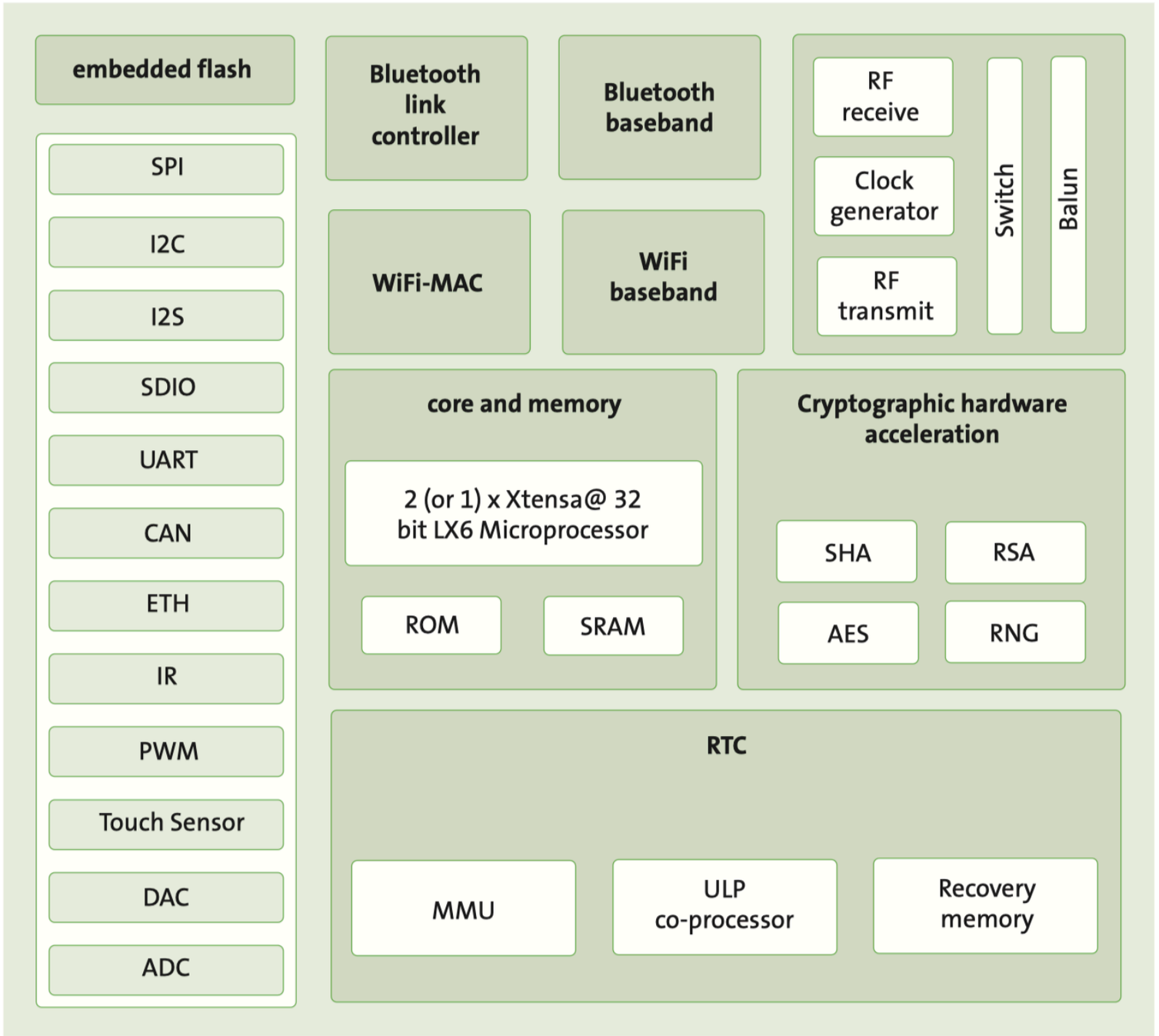
PIR Application

ESP32 Block Diagram



// ESP 32: 2 microprocessor, normally use one. The other one runs operating systems.

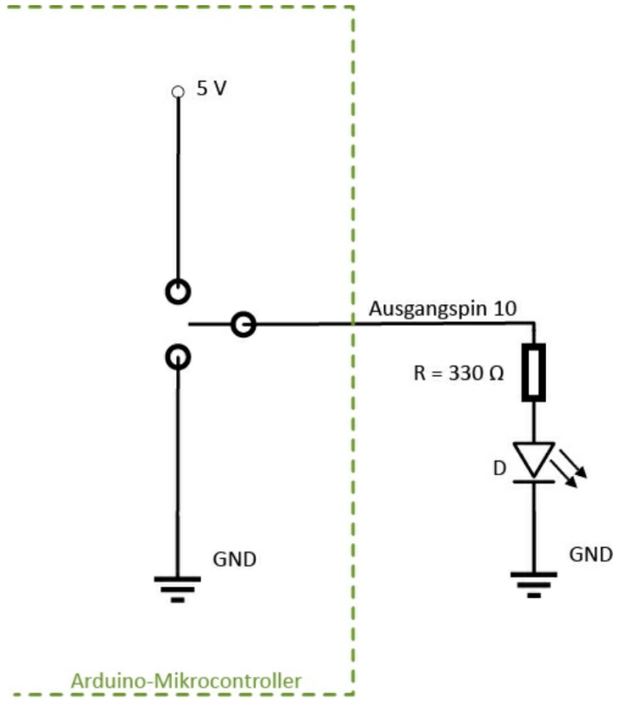
// data stored in flash. Then sent to SRAM

ESP32 Memory

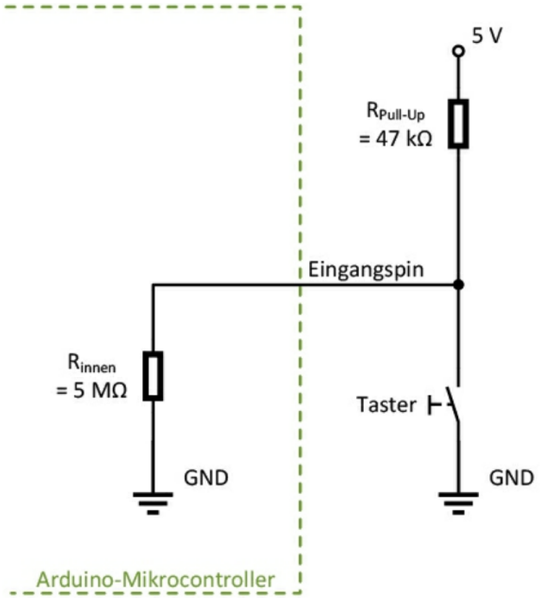
- 520 KB SRAM for data and instructions
- 448 KB ROM for boot and kernel functions
- 8 KB SRAM in RTC (RTC-FAST Memory) safe data over deep sleep
- 8 KB SRAM in RTC (RTC-SLOW Memory) for coprocessor during deep sleep
- ESP Wrover-B
  - 4 MB QSPI Flash memory
  - Additional 8 MB PSRAM (Pseudo Static RAM)

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
```

enable / disable

Error Handling

ESP_ERROR_CHECK(x)

Macro which can be used to check the error code, and terminate the program in case the code is not ESP_OK. Prints the error code, error location, and the failed statement to serial output.

Disabled if assertions are disabled.

assertions: in code, sth. that can be enabled and disabled

ESP_ERROR_CHECK_WITHOUT_ABORT(x)

In comparison with ESP_ERROR_CHECK(), this prints the same error message but isn't terminating the program.

Interrupt Handling

interrupt: always come from outside.
Intrution: from system inside

```
esp_err_t gpio_install_isr_service(int intr_alloc_flags)

//ESP_INTR_FLAG_IRAM: interrupt handler needs to be installed
 with IRAM_ATTR in the instruction RAM
```

Interrupt Handling

```
esp_err_t gpio_set_intr_type(gpio_num_t gpio_num,
                             gpio_int_type_t intr_type)

//GPIO_INTR_POSEDGE, ... _NEGEDGE, ..._ANYEDGE,
//..._LOW_LEVEL, ..._HIGH_LEVEL
```

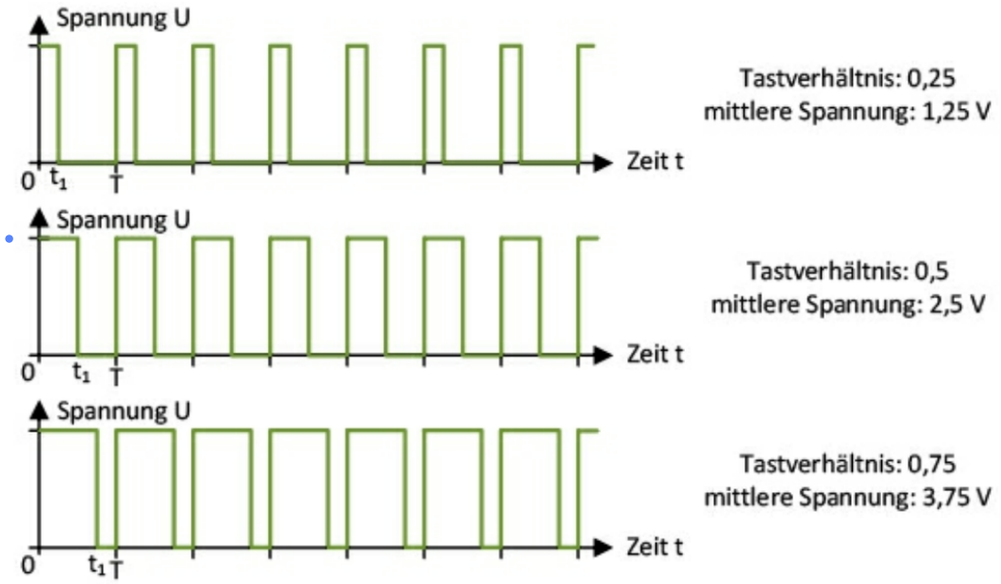
```
esp_err_t gpio_isr_handler_add(gpio_num_t gpio_num, gpio_isr_t
 isr_handler, void *args)
```

```
void IRAM_ATTR outISR(void* arg){
   ets_printf("Interrupt OUT.\n");
   ...
}
```

I/O Pins

- Analog Digital Converter (ADC)
  - Input is a voltage. 18 channels.
  - E.g. The range of 0..3,3V is represented as 0..4096 for a 12 bit resolution
- Digital Analog Converter (DAC)
  - Converts digital value to analog voltage.
  - Two 8 bit converters connected to Pins 25 and 26.
  - Based on a 3.3 V for maximum
- Pulse Width Modulation (PWM)
  - Mainly for LED control on ESP32
  - Digital output is pulsed according to the digital input (0...255)
  - 16 channels

brighter, darker...



I/O Pins

- Serial Interface (UART or RS-232) E.g. monitoring
  - Fixed baud rate (bit/s)
  - 2 lines (input and output)
  - Connects two systems
- I2C - Inter-Integrated Circuit Bus frequently used
  - Serial bus
  - 2 lines: data (SDA) and clock (SCL)
  - Connecting master with multiple slaves identified by addresses
- SPI - Serial Peripheral Interface (SPI) High communication demand
  - Data transmission on separate input and output line
    - SCK: clock
    - MISO: master in slave out
    - MOSI: master out slave in
  - Requires additional signal to select a slave on the bus.

ESP Logging

- ESP IDF provides logging macros
  - `ESP_LOGI(TAG,"Free stack space: %d",freestack);`
  - `static const char* TAG = "MyModule"`
  - Verbosity levels: error (lowest), warning, info, debug, verbose (highest)
  - `#include "esp_log.h"`
- Logging can be controlled at compile time and run time.
  - Compile time
    - Default log verbosity set via menuconfig (component config/log output)
    - Can be overwritten by `#define LOG_LOCAL_LEVEL ESP_LOG_VERBOSE`
  - Run time
    - `esp_log_level_set("*", ESP_LOG_ERROR)`
    - `esp_log_level_set("MyModule", ESP_LOG_WARN)`

Try code: idf menuconfig

Deep Sleep

- External Wakeup (ext0)
  - Can be connected to a single RTC GPIO pin

```
esp_err_t esp_sleep_enable_ext0_wakeup(gpio_num_t gpio_num,
                                       int level)
    //input level which will trigger wakeup (0=low, 1=high)
```

- External Wakeup (ext1)
  - Can be connected to multiple RTC GPIO pins
  - Can be triggered by all low or any high

```
esp_err_t esp_sleep_enable_ext1_wakeup(uint64_t mask,
                                       esp_sleep_ext1_wakeup_mode_t mode)
    //ESP_EXT1_WAKEUP_ALL_LOW or ... ANY_HIGH
```

How to use EXT1

```
#define PIR_PIN_MASK ((1ULL<<PIR1_PIN)|(1ULL<<PIR2_PIN))

esp_sleep_enable_ext1_wakeup(PIR_PIN_MASK,ESP_EXT1_WAKEUP_ANY_HIGH);

uint64_t wakeup_mask=esp_sleep_get_ext1_wakeup_status();
  if ((wakeup_mask & (1ULL<<PIR1_PIN)) !=0){
      ……
  }
```

RTC GPIO

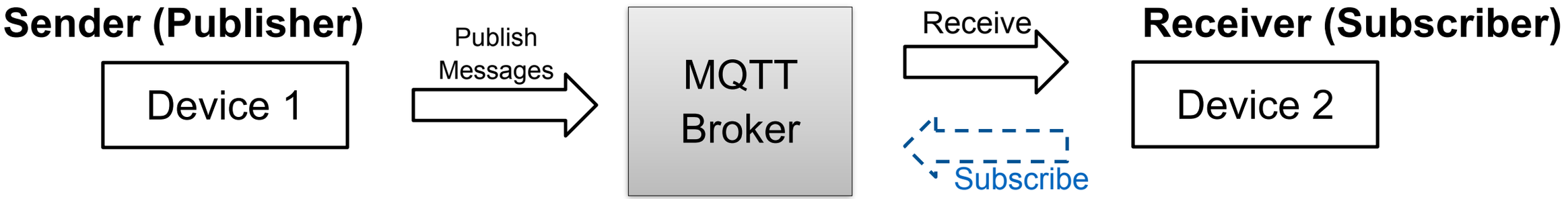
- Only some of the GPIOs can be used in the RTC module
- The API routines use the GPIO pin numbers

GPIO	Analog Function	RTC GPIO	Comments
GPIO0	ADC2_CH1	RTC_GPIO11	Strapping pin
GPIO1			TXD
GPIO2	ADC2_CH2	RTC_GPIO12	Strapping pin
GPIO3			RXD
GPIO4	ADC2_CH0	RTC_GPIO10	
GPIO5			Strapping pin
GPIO6			SPI0/1
GPIO7			SPI0/1
GPIO8			SPI0/1
GPIO9			SPI0/1
GPIO10			SPI0/1
GPIO11			SPI0/1
GPIO12	ADC2_CH5	RTC_GPIO15	Strapping pin; JTAG
GPIO13	ADC2_CH4	RTC_GPIO14	JTAG
GPIO14	ADC2_CH6	RTC_GPIO16	JTAG
GPIO15	ADC2_CH3	RTC_GPIO13	Strapping pin; JTAG
GPIO16			SPI0/1
GPIO17			SPI0/1
GPIO18			
GPIO19			
GPIO20			This pin is only available on ESP32-PICO-V3 chip package
GPIO21			
GPIO22			
GPIO23			
GPIO25	ADC2_CH8	RTC_GPIO6	
GPIO26	ADC2_CH9	RTC_GPIO7	
GPIO27	ADC2_CH7	RTC_GPIO17	
GPIO32	ADC1_CH4	RTC_GPIO9	
GPIO33	ADC1_CH5	RTC_GPIO8	
GPIO34	ADC1_CH6	RTC_GPIO4	GPI
GPIO35	ADC1_CH7	RTC_GPIO5	GPI
GPIO36	ADC1_CH0	RTC_GPIO0	GPI
GPIO37	ADC1_CH1	RTC_GPIO1	GPI
GPIO38	ADC1_CH2	RTC_GPIO2	GPI
GPIO39	ADC1_CH3	RTC_GPIO3	GPI

check: there must be RTC_GPIO connected. Therefore can use 25 but not 23

MQTT

- MQ Telemetry Transport or Message Queuing Telemetry Transport
- Lightweight, publish-subscribe network protocol that transports messages between devices.



1. Device 1 publishes to a **topic**

2. Device 2 subscribes to **same topic**
3. Device 2 receives messages.

- Topics are represented with strings and separated by slashes. Slashes indicate topic level.
- Example: office/room123/lamp (for lamp in room 123 in office)
- Broker is responsible for receiving the messages, filtering them and publishing to subscribed clients (For example Mosquitto broker).

MQTT QoS

- QoS levels in MQTT:
  - At most once (0)
  - At least once (1)
  - Exactly once (2).
- When you talk about QoS in MQTT, you need to consider the two sides of message delivery:
  - Message delivery form the publishing client to the broker.
  - Message delivery from the broker to the subscribing client.
- Higher QoS level requires more overhead
  - e.g. QoS 1

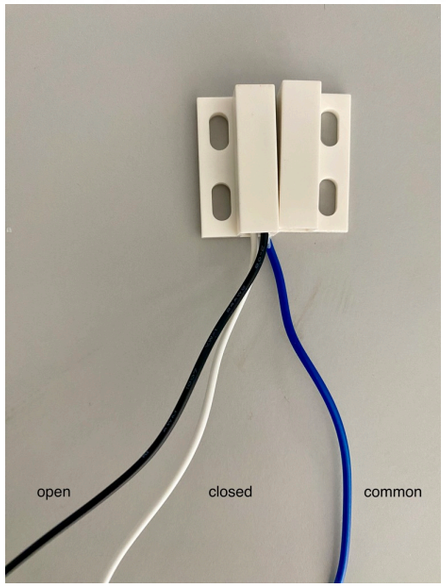
Quality of Service level 1: delivery at least once

Today’s Activity

- Get the digital twin ready for your first sensor
- Compile and flash the PIR code
- See your first sensor values

Distribution of Hardware

- Distribution of hardware
  - RPi
  - 2 ESP32
    - Golden with battery gauge
    - Green without
  - 4 PIR sensors (active: signal high)
  - 1 Door contact sensor with 4 m cable (open when door is closed)
  - 3 Batteries + 1 USB power supply (Battery can be charged via the board.), battery holder
  - Jumper cable (red: VCC, black: GND, other: signal)



Raspberry Pi

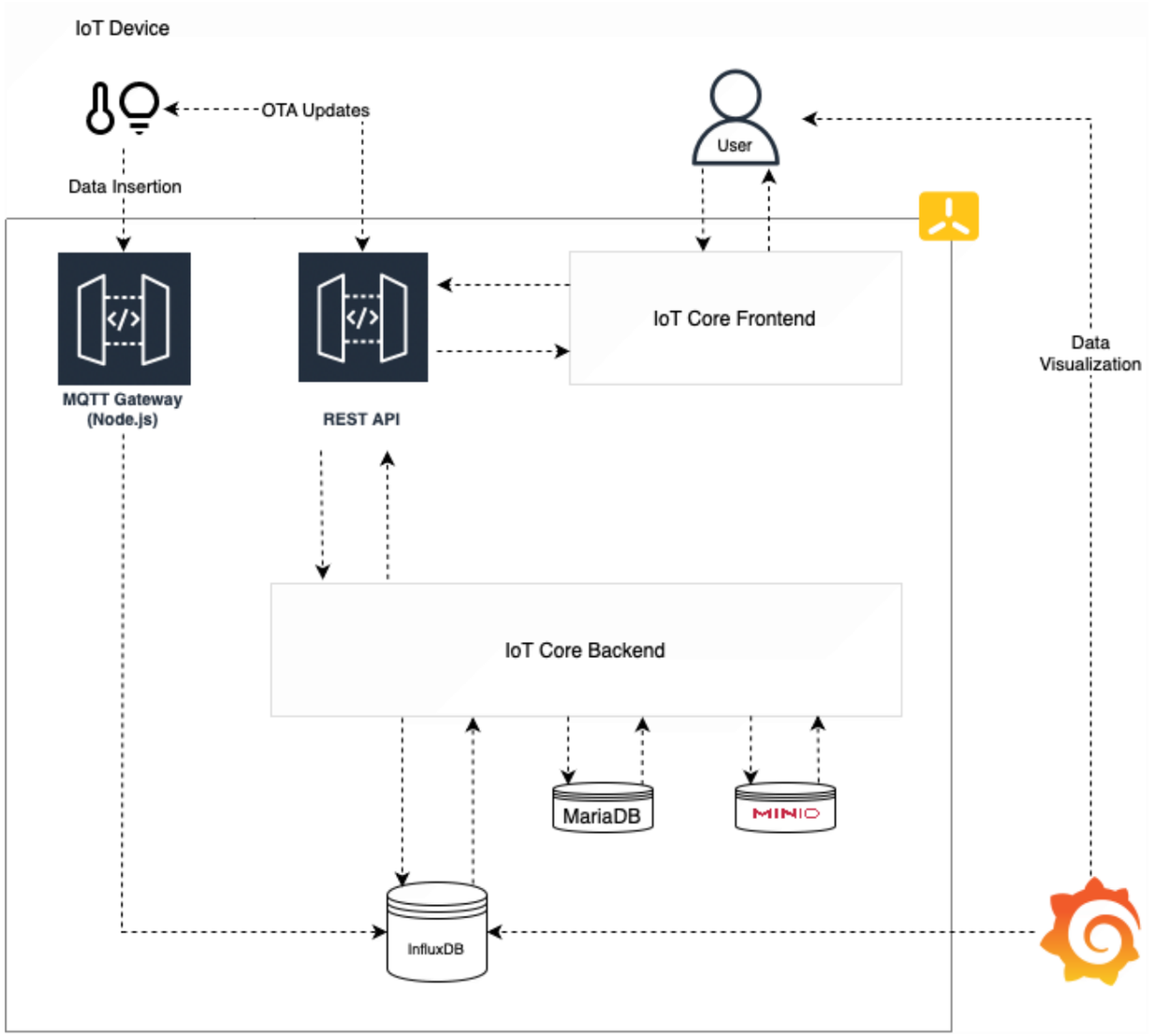
- Get the RPi's IP address
  - Plugin the RPi to power
  - Wait for one of us with a keyboard
  - **`ifconfig bond0`** will show the IP address
- Install the virtual keyboard
  - Login to your RPi via ssh
  - run **`sudo apt install onboard at-spi2-core`**
  - You can now access the virtual keyboard from
    - Raspberry Pi logo -> Universal Access -> Onboard

CAPS IoT Platform

- Manage devices and sensors of devices
  - Device is an ESP32
  - Sensor is PIR, door sensor, battery gauge
- Provides the MQTT broker to ingest data into influx

- Any other functionality important for the lab?

Architecture



IoT Platform: Device Management

- Use ifconfig on RPi to get your IP address
  - ifconfig bond0
- On your laptop use the browser to access the IoT platform
  - http://<your IP>:30020
  - user: iot_user
  - password: iot-user

Schema: PIR

IoT Platform

IoT User

- Sensor-Schemas
- Devices
- OTA Updates
- FaaS Functions
- Configuration
- Sign out

Edit schema Schema

Name

PIR

[Optional] Description

```
{
  sensors: array required
  [
    struct required
    {
      name: string required
      values: array required
      [
        struct required
        ^{
          timestamp: int64 required
          - roomID : string required
          +
        }
      ]
    }
  ]
}
```

Schema: battery

Name

battery

[Optional] Description

```
{
    sensors: array required
    [
        struct required
        {
            name: string required
            values: array required
            [
                struct required
                ^{
                    timestamp: int64 required
                    - voltage : double required
                    - soc : double required
                    +
                }
            ]
        }
    ]
}
```

Device: Corridor

corridor Edit Device Download Device Key

Description

External MQTT Broker

This device uses the internal MQTT Broker

Sensors Add Sensor

- PIR CLEAN UP Edit Delete
- battery CLEAN UP Edit Delete

Download Device Key

{"**token**":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE3Mjk1ODMzNjcsImlzcyI6ImlvdHBsYXRmb3JtIiwic3ViIjoiMS8yIn0.QcbylziV7ZW44-wi2JciCnh5ooUPKiPWzFAbBN6Ibf9bGbly8438E7LEU6H0vWO56cbN9LGYn32g2EJmfaCp9tbBV0mZ_G07rHTX5Dt0YxTSxKblxpjvkhPqHrZejr7nmo7Wn3i14OA25y8T8SRcZqePBkcoj60IQr2k5CSkFqck8W24c4vo2IlG8CuL0VOOs6k8UkATyqkR_mnjGiYNtoN5JX-iK2rR40PDoCESnqsZE4CMSY7NL5_DcvL8crynf_SczcEqcvdhPeMGSQylretIlWWmxgsT2caEXByI6483E7-FaZpGSP7iPb4zMQ3RWlDJvs607nd13PWQ9kCi58phQkAOHGEbGnMj5jad4Lx25LEOdnmA-j29sqRAHwoVZ32vQO_SnTaVXZHh8bEAGkT-95xDSWOs5OojnL5-GLT9mYjImzsGcOcQe9IWXG2NueDmhOXLJN1cLPyIoJGBqEEWNnqRChp8LclRExlwpIIUxdwX1z8l_x1bIscRSclW6jn79i4qh2CoRtb0e7oVwkI9q-bHZkICU-3Wuky9sd63zTJB8_jridXQ7cZpiTab1bh4Qydn7U6pw6ecxjUcMnRwFejTYMSHznBzxKpsiGPKVxIUlyhY9unK0lJWN8p8LjpFq01ChBJn9bjU5f7WBMpoWYQGjoKFC4VR1NO7cW8","**user_id**":1,"**device_id**":2}

Download Software

- Get zip file of the firmware
- Put it into a directory
- esp-idf-lib is provided. It is used for the battery gauge of the golden ESP32 board.

PIR Application

Prepare your board

Breadboard								Breadboard
20			S_VP		UART TxD			20
19			S_VN		UART RxD			19
18			GND		GND			18
17			Vin		Vin			17
16			IO35		IO19			16
15			IO34		IO18	LED B		15
14			IO33		IO17	LED G		14
13			IO32		IO16	LED R		13
12			3.3V		3.3V			12
11			DAC0		IO15			11
10			DAC1		IO14			10
9	PIR signal		IO27		IO13			9
8					IO12			8
7			IO23		IO5			7
6		I2C SCL	IO22		IO4			6
5		I2C SDA	IO21		IO2			5
4			EN		IO0			4
3	VDD		3.3V		3.3V			3
2			Vin		Vin			2
1	GND		GND		GND			1

ESP32 WROOM-32E

Modify main.hpp

- Update device information

```
#define DEVICE_KEY                    „<your device key>“
#define DEVICE_ID                     "2"
#define DEVICE_TOPIC                  „1/2/data"
```

- Update the address of the MQTT broker

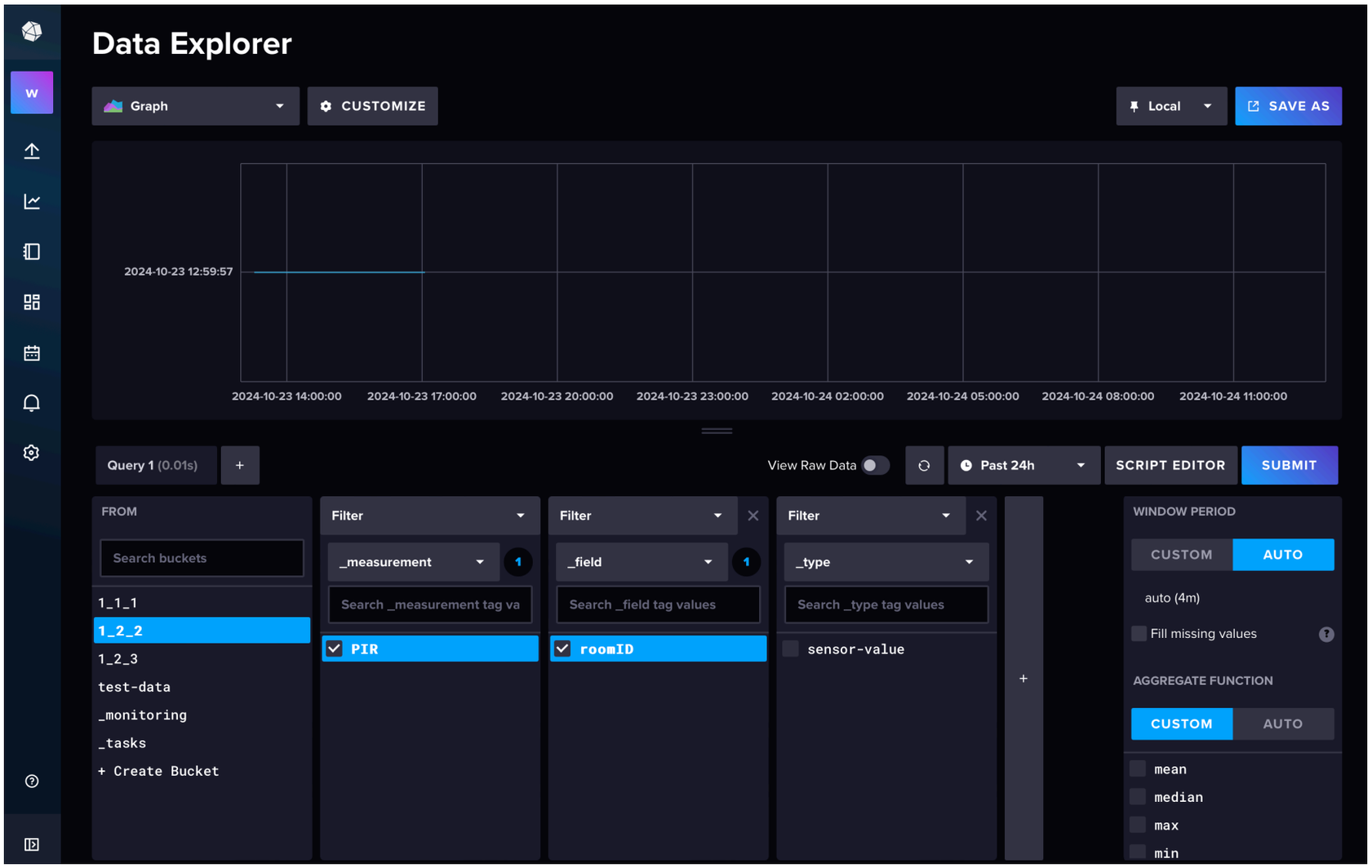
```
#define MQTT_BROKER                   „<your RPi’s IP address>“
```

Influx used to Store Events

- Influx database
  - Time-series database
  - Stores and enables querying of time-stamped data
  - Flux query language
  - Support for retention policies
  - Data are structured into Buckets (former called Databases), Measurements (e.g., CPU utilization), Tags (giving the context: hsot=server01) used to index the data
  - CAPS platform:
    - bucket: „<user number>/<device number>/<sensor number>
    - measurement: <sensor name>
    - tags: not used
- Flux
  - Functional specification: series of functions that transform data
  - Queries are structured into pipes (|>) that connect different functions
  -
```
from(bucket: "example-bucket")
|> range(start: -1h)
|> filter(fn: (r) => r._measurement == "cpu")
|> mean()
```

Using Influx

- Connect to http://<your IP address>:8086
  - user: admin
  - password: secure_influx_iot_user



In case of troubles

- If the platform or influx cannot be reached from outside although you can ping and ssh the RPi
  - Restart the pod network connecting the digital twin components
    - **`sudo find /var/lib/cni/results -size 0 -delete`**
    - **`sudo find /var/lib/cni/flannel/results -size 0 -delete`**
  - See the status of the pods
    - **`kubectl get pods -A`**

delete 0 size files

sudo reboot: restart raspi

Homework

- Create additional devices with sensors
- Use MAC address for identification
  - Include „esp_mac.h“
  - use **`esp_read_mac`** to retrieve the mac address
  - check the address with **`memcmp(...)`** to do device specific settings
- Add magnetic switch sensor
  - Support for multiple sensor on one ESP requires usage of EXT1 wakeup

Questions