Industrial IoT Solutions

Lobaro & Lobaro CoAP

27.10.2016

The Lobaro Team

Dipl.-Ing. Tobias Rohde

- Founder of Lobaro (CEO)
- Embedded Design
- Backend Programming

Dipl.-Ing. Tobias Kaupat

- Software dev. Head
- Backend & Frontend
- Server Management

Other Team-Members:

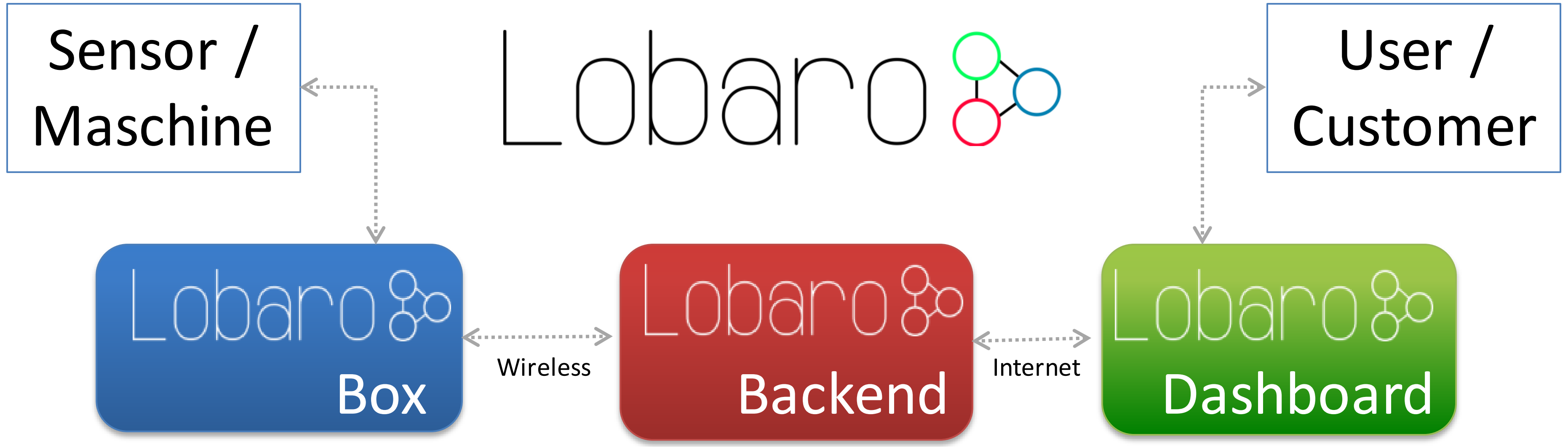
Alexander Zahn

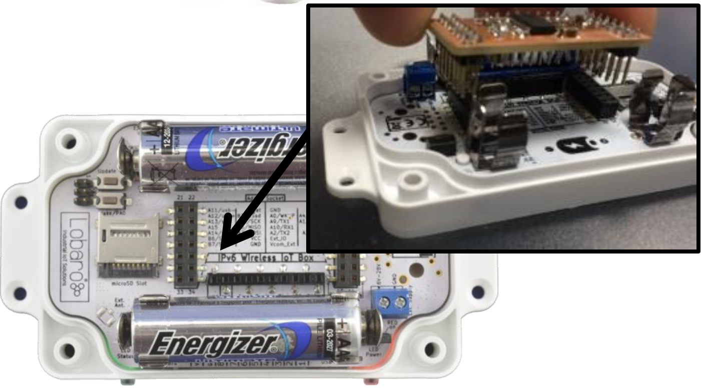
- Website
- Online Marketing
- Customer Service

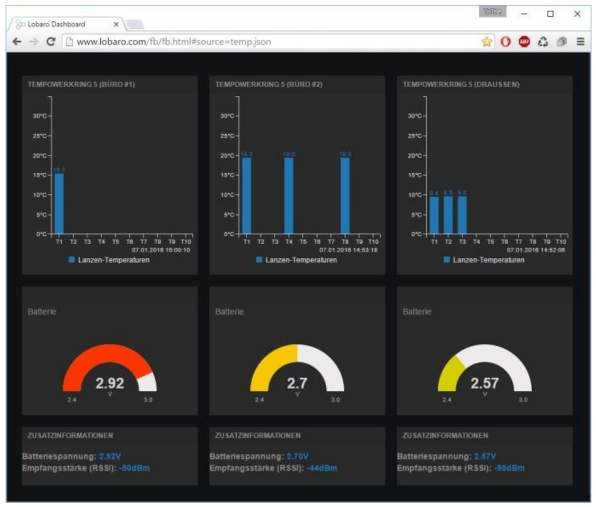
Dipl.-Ing. Kai Gillmann

- Hardware Manufacturing
- Quality Assurance / EMC
- Consulting for
Electronic Development

“Fullstack“ Internet of Things







TEMPOWERKRING 5 (BÜRO #1)

ZUSATZINFORMATIONEN
Batteriespannung: 2.92V
Empfangsstärke (RSSI): -50dBm

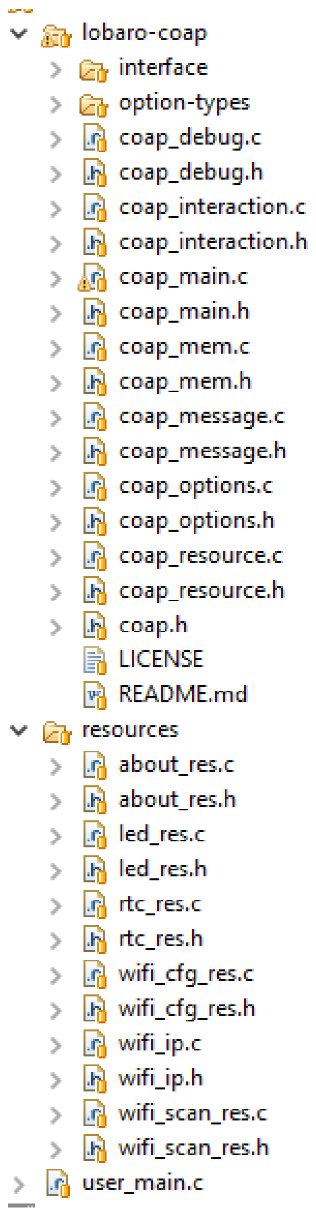
TEMPOWERKRING 5 (BÜRO #2)

ZUSATZINFORMATIONEN
Batteriespannung: 2.70V
Empfangsstärke (RSSI): -44dBm

TEMPOWERKRING 5 (DRAUSSEN)

ZUSATZINFORMATIONEN
Batteriespannung: 2.57V
Empfangsstärke (RSSI): -98dBm

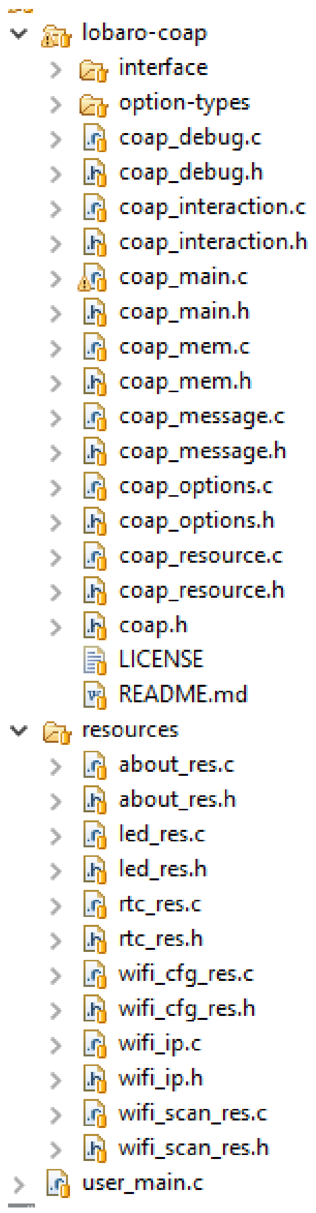
Lobaro CoAP: Overview



```
lobaro-coap
  interface
  option-types
  coap_debug.c
  coap_debug.h
  coap_interaction.c
  coap_interaction.h
  coap_main.c
  coap_main.h
  coap_mem.c
  coap_mem.h
  coap_message.c
  coap_message.h
  coap_options.c
  coap_options.h
  coap_resource.c
  coap_resource.h
  coap.h
  LICENSE
  README.md
resources
  about_res.c
  about_res.h
  led_res.c
  led_res.h
  rtc_res.c
  rtc_res.h
  wifi_cfg_res.c
  wifi_cfg_res.h
  wifi_ip.c
  wifi_ip.h
  wifi_scan_res.c
  wifi_scan_res.h
user_main.c
```

- CoAP for embedded devices (e.g. Cortex-M3/M0)

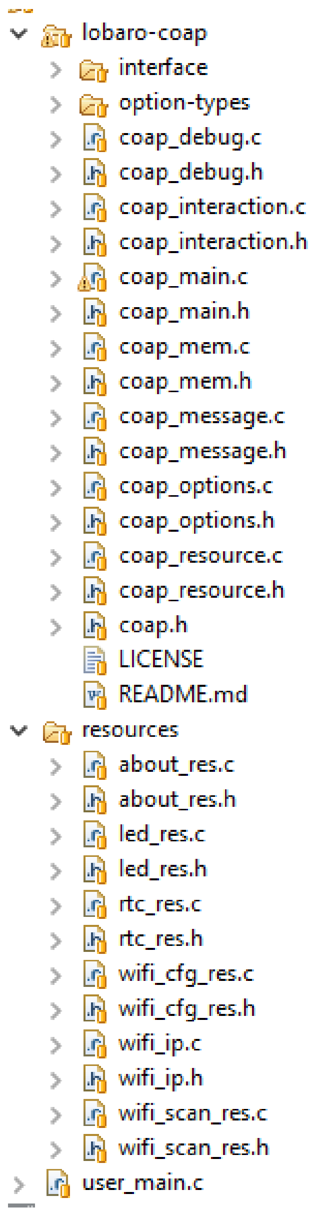
Lobaro CoAP: Overview



```
lobaro-coap
  interface
  option-types
  coap_debug.c
  coap_debug.h
  coap_interaction.c
  coap_interaction.h
  coap_main.c
  coap_main.h
  coap_mem.c
  coap_mem.h
  coap_message.c
  coap_message.h
  coap_options.c
  coap_options.h
  coap_resource.c
  coap_resource.h
  coap.h
  LICENSE
  README.md
resources
  about_res.c
  about_res.h
  led_res.c
  led_res.h
  rtc_res.c
  rtc_res.h
  wifi_cfg_res.c
  wifi_cfg_res.h
  wifi_ip.c
  wifi_ip.h
  wifi_scan_res.c
  wifi_scan_res.h
user_main.c
```

- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)

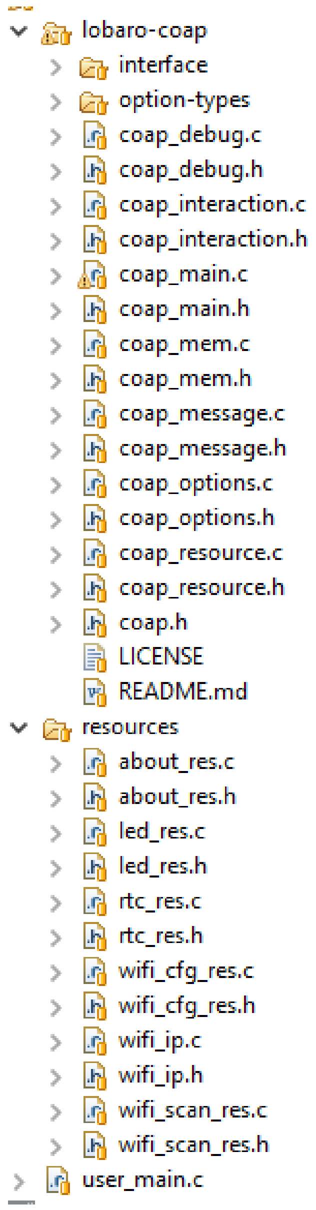
Lobaro CoAP: Overview



```
lobaro-coap
    interface
    option-types
    coap_debug.c
    coap_debug.h
    coap_interaction.c
    coap_interaction.h
    coap_main.c
    coap_main.h
    coap_mem.c
    coap_mem.h
    coap_message.c
    coap_message.h
    coap_options.c
    coap_options.h
    coap_resource.c
    coap_resource.h
    coap.h
    LICENSE
    README.md
resources
    about_res.c
    about_res.h
    led_res.c
    led_res.h
    rtc_res.c
    rtc_res.h
    wifi_cfg_res.c
    wifi_cfg_res.h
    wifi_ip.c
    wifi_ip.h
    wifi_scan_res.c
    wifi_scan_res.h
user_main.c
```

- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support

Lobaro CoAP: Overview



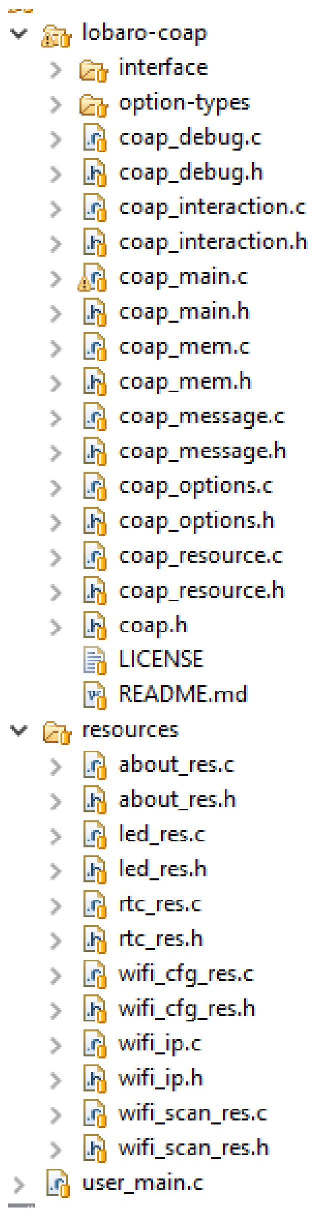
lobaro-coap
- interface
- option-types
- coap_debug.c
- coap_debug.h
- coap_interaction.c
- coap_interaction.h
- coap_main.c
- coap_main.h
- coap_mem.c
- coap_mem.h
- coap_message.c
- coap_message.h
- coap_options.c
- coap_options.h
- coap_resource.c
- coap_resource.h
- coap.h
- LICENSE
- README.md

resources
- about_res.c
- about_res.h
- led_res.c
- led_res.h
- rtc_res.c
- rtc_res.h
- wifi_cfg_res.c
- wifi_cfg_res.h
- wifi_ip.c
- wifi_ip.h
- wifi_scan_res.c
- wifi_scan_res.h

user_main.c

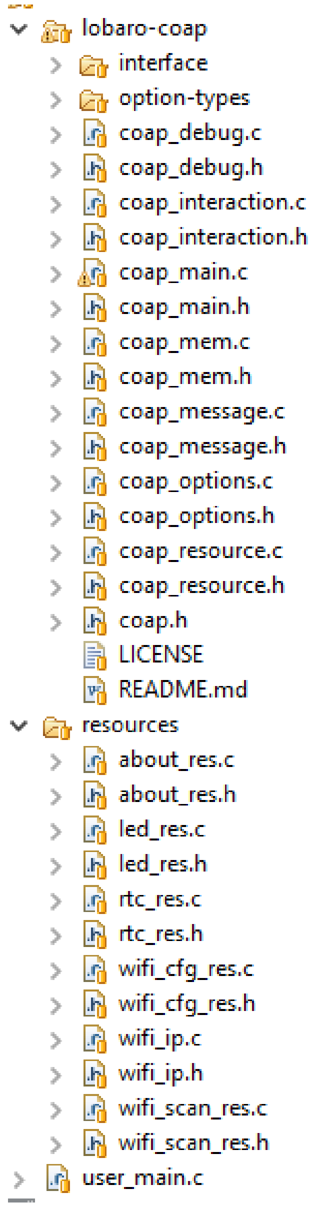
- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser

Lobaro CoAP: Overview


```
lobaro-coap
  interface
  option-types
  coap_debug.c
  coap_debug.h
  coap_interaction.c
  coap_interaction.h
  coap_main.c
  coap_main.h
  coap_mem.c
  coap_mem.h
  coap_message.c
  coap_message.h
  coap_options.c
  coap_options.h
  coap_resource.c
  coap_resource.h
  coap.h
  LICENSE
  README.md
resources
  about_res.c
  about_res.h
  led_res.c
  led_res.h
  rtc_res.c
  rtc_res.h
  wifi_cfg_res.c
  wifi_cfg_res.h
  wifi_ip.c
  wifi_ip.h
  wifi_scan_res.c
  wifi_scan_res.h
user_main.c
```

- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser
- One simple „doWork“ loop – no RTOS needed

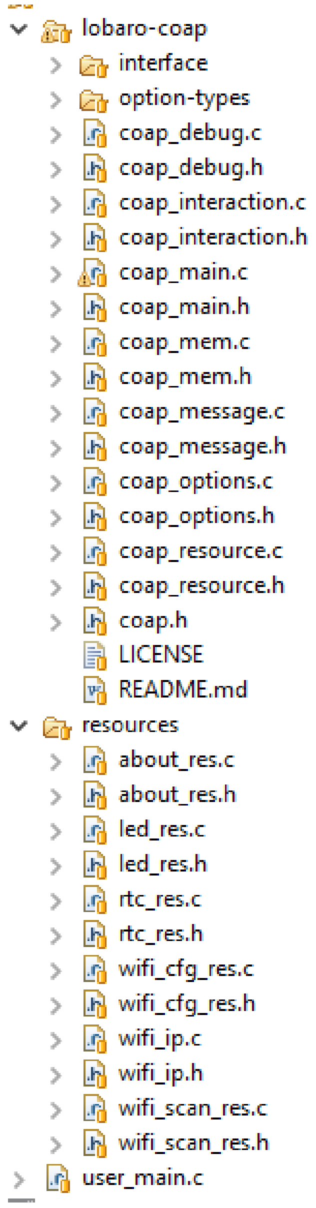
Lobaro CoAP: Overview



```
lobaro-coap
  interface
  option-types
  coap_debug.c
  coap_debug.h
  coap_interaction.c
  coap_interaction.h
  coap_main.c
  coap_main.h
  coap_mem.c
  coap_mem.h
  coap_message.c
  coap_message.h
  coap_options.c
  coap_options.h
  coap_resource.c
  coap_resource.h
  coap.h
  LICENSE
  README.md
resources
  about_res.c
  about_res.h
  led_res.c
  led_res.h
  rtc_res.c
  rtc_res.h
  wifi_cfg_res.c
  wifi_cfg_res.h
  wifi_ip.c
  wifi_ip.h
  wifi_scan_res.c
  wifi_scan_res.h
user_main.c
```

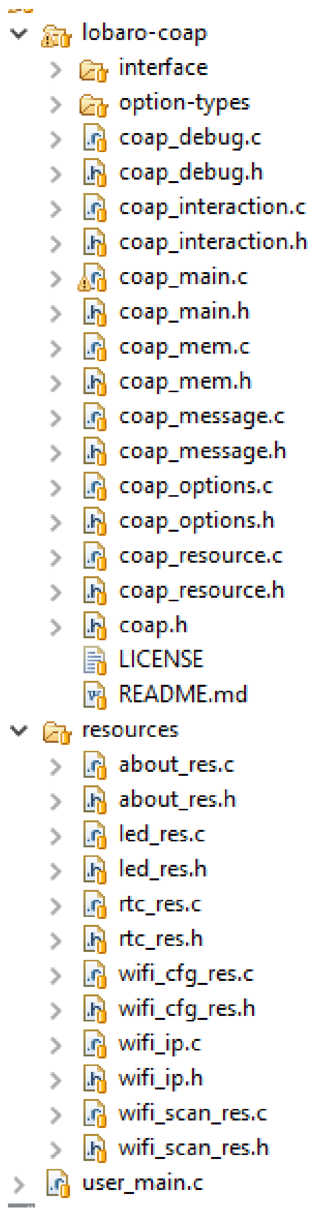
- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser
- One simple „doWork“ loop – no RTOS needed
- Internal memory allocator on <u>static</u> Array (BGET)

CoAP: Overview


lobaro-coap
interface
option-types
coap_debug.c
coap_debug.h
coap_interaction.c
coap_interaction.h
coap_main.c
coap_main.h
coap_mem.c
coap_mem.h
coap_message.c
coap_message.h
coap_options.c
coap_options.h
coap_resource.c
coap_resource.h
coap.h
LICENSE
README.md
resources
about_res.c
about_res.h
led_res.c
led_res.h
rtc_res.c
rtc_res.h
wifi_cfg_res.c
wifi_cfg_res.h
wifi_ip.c
wifi_ip.h
wifi_scan_res.c
wifi_scan_res.h
user_main.c

- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser
- One simple „doWork“ loop – no RTOS needed
- Internal memory allocator on static Array (BGET)
- To be used with any packet oriented transport

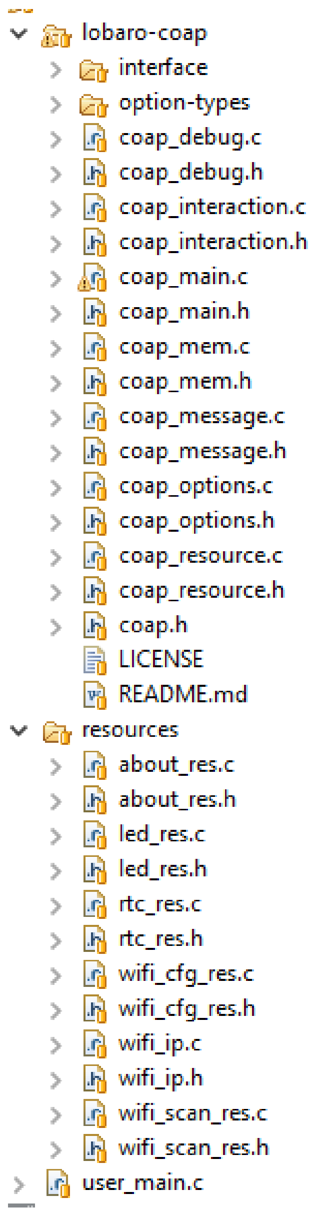
Lobaro CoAP: Overview



```
lobaro-coap
    interface
    option-types
    coap_debug.c
    coap_debug.h
    coap_interaction.c
    coap_interaction.h
    coap_main.c
    coap_main.h
    coap_mem.c
    coap_mem.h
    coap_message.c
    coap_message.h
    coap_options.c
    coap_options.h
    coap_resource.c
    coap_resource.h
    coap.h
    LICENSE
    README.md
resources
    about_res.c
    about_res.h
    led_res.c
    led_res.h
    rtc_res.c
    rtc_res.h
    wifi_cfg_res.c
    wifi_cfg_res.h
    wifi_ip.c
    wifi_ip.h
    wifi_scan_res.c
    wifi_scan_res.h
user_main.c
```

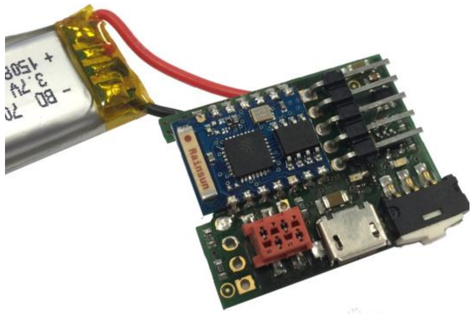
- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser
- One simple „doWork“ loop – no RTOS needed
- Internal memory allocator on static Array (BGET)
- To be used with any packet oriented transport
- Main objective: Easy usage & Ressource definition

Lobaro CoAP: Overview



```
lobaro-coap
  interface
  option-types
  coap_debug.c
  coap_debug.h
  coap_interaction.c
  coap_interaction.h
  coap_main.c
  coap_main.h
  coap_mem.c
  coap_mem.h
  coap_message.c
  coap_message.h
  coap_options.c
  coap_options.h
  coap_resource.c
  coap_resource.h
  coap.h
  LICENSE
  README.md
resources
  about_res.c
  about_res.h
  led_res.c
  led_res.h
  rtc_res.c
  rtc_res.h
  wifi_cfg_res.c
  wifi_cfg_res.h
  wifi_ip.c
  wifi_ip.h
  wifi_scan_res.c
  wifi_scan_res.h
user_main.c
```

- CoAP for embedded devices (e.g. Cortex-M3/M0)
- CoAP Client & Server in one stack! (C-lang)
- Observe & blockwise support
- More than just a CoAP packet builder/parser
- One simple „doWork“ loop – no RTOS needed
- Internal memory allocator on <u>static</u> Array (BGET)
- To be used with any packet oriented transport
- Main objective: Easy usage & Ressource definition

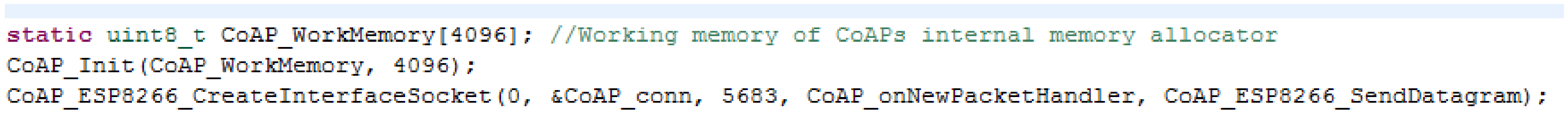


ESP8266 (WIFI) Port

ZWIR4512 (802.15.4 + 6LoWPAN) Port

CoAP: Easy usage

1. Provide packet oriented binding (e.g. UDP, SLIP) + memory



```c
static uint8_t CoAP_WorkMemory[4096]; //Working memory of CoAPs internal memory allocator
CoAP_Init(CoAP_WorkMemory, 4096);
CoAP_ESP8266_CreateInterfaceSocket(0, &CoAP_conn, 5683, CoAP_onNewPacketHandler, CoAP_ESP8266_SendDatagram);
```

Lobaro CoAP: Easy usage

1. Provide packet oriented binding (e.g. UDP, SLIP) + memory

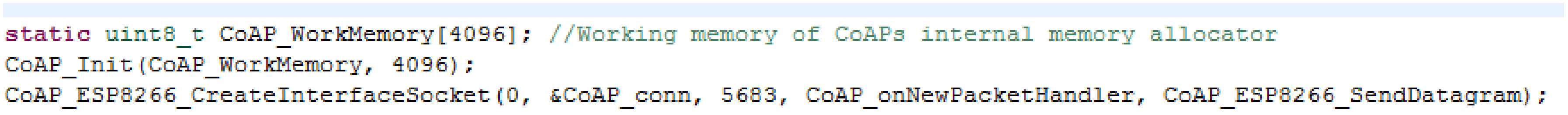
```
static uint8_t CoAP_WorkMemory[4096]; //Working memory of CoAPs internal memory allocator
CoAP_Init(CoAP_WorkMemory, 4096);
CoAP_ESP8266_CreateInterfaceSocket(0, &CoAP_conn, 5683, CoAP_onNewPacketHandler, CoAP_ESP8266_SendDatagram);
```

2. Create resources + bind to Request/Observe Handlers

```
CoAP_Res_t* ICACHE_FLASH_ATTR Create_About_Resource() {
    CoAP_ResOpts_t Options = {.Cf = COAP_CF_TEXT_PLAIN, .Flags = RES_OPT_GET};
    return (pAbout_Res=CoAP_CreateResource("about/coap", "CoAP Description",Options, RequestHandler, NULL));
}
```

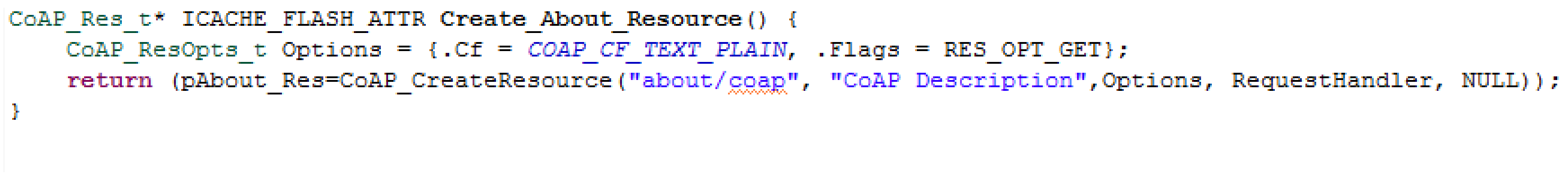
CoAP: Easy usage

1. Provide packet oriented binding (e.g. UDP, SLIP) + memory



```
static uint8_t CoAP_WorkMemory[4096]; //Working memory of CoAPs internal memory allocator
CoAP_Init(CoAP_WorkMemory, 4096);
CoAP_ESP8266_CreateInterfaceSocket(0, &CoAP_conn, 5683, CoAP_onNewPacketHandler, CoAP_ESP8266_SendDatagram);
```

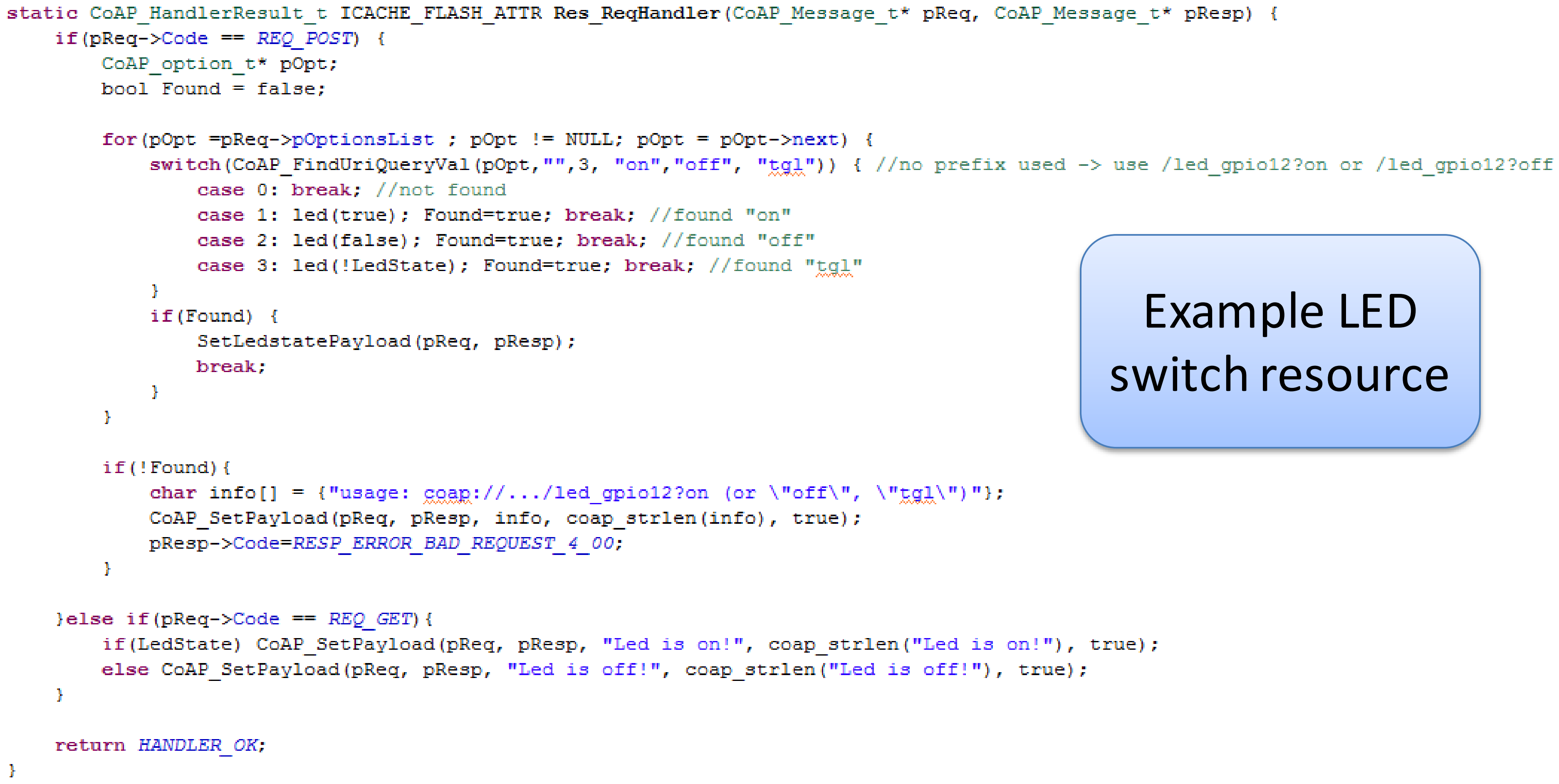
2. Create resources + bind to Request/Observe Handlers



```
CoAP_Res_t* ICACHE_FLASH_ATTR Create_About_Resource() {
    CoAP_ResOpts_t Options = {.Cf = COAP_CF_TEXT_PLAIN, .Flags = RES_OPT_GET};
    return (pAbout_Res=CoAP_CreateResource("about/coap", "CoAP Description",Options, RequestHandler, NULL));
}
```

3. Implement resource handlers…

CoAP: LED Request Handler



```c
static CoAP_HandlerResult_t ICACHE_FLASH_ATTR Res_ReqHandler(CoAP_Message_t* pReq, CoAP_Message_t* pResp) {
    if(pReq->Code == REQ_POST) {
        CoAP_option_t* pOpt;
        bool Found = false;

        for(pOpt =pReq->pOptionsList ; pOpt != NULL; pOpt = pOpt->next) {
            switch(CoAP_FindUriQueryVal(pOpt,"",3, "on","off", "tgl")) { //no prefix used -> use /led_gpio12?on or /led_gpio12?off
                case 0: break; //not found
                case 1: led(true); Found=true; break; //found "on"
                case 2: led(false); Found=true; break; //found "off"
                case 3: led(!LedState); Found=true; break; //found "tgl"
            }
            if(Found) {
                SetLedstatePayload(pReq, pResp);
                break;
            }
        }

        if(!Found){
            char info[] = {"usage: coap://.../led_gpio12?on (or \"off\", \"tgl\")"};
            CoAP_SetPayload(pReq, pResp, info, coap_strlen(info), true);
            pResp->Code=RESP_ERROR_BAD_REQUEST_4_00;
        }

    }else if(pReq->Code == REQ_GET){
        if(LedState) CoAP_SetPayload(pReq, pResp, "Led is on!", coap_strlen("Led is on!"), true);
        else CoAP_SetPayload(pReq, pResp, "Led is off!", coap_strlen("Led is off!"), true);
    }

    return HANDLER_OK;
}
```

Example LED switch resource

CoAP logic (e.g. retries, options) is transparent to user!

Lobaro CoAP: Blockwise Request Handler

```
static const char CoapInfoStringInFlash[] = {"\
The Constrained Application Protocol (CoAP) is a specialized web \
transfer protocol for use with constrained nodes and constrained \
(e.g., low-power, lossy) networks.  The nodes often have 8-bit \
microcontrollers with small amounts of ROM and RAM, while constrained \
networks such as IPv6 over Low-Power Wireless Personal Area Networks \
(6LoWPANs) often have high packet error rates and a typical \
throughput of 10s of kbit/s.  The protocol is designed for machine- \
to-machine (M2M) applications such as smart energy and building \
automation.\r\n\r\n\
\
CoAP provides a request/response interaction model between \
application endpoints, supports built-in discovery of services and \
resources, and includes key concepts of the Web such as URIs and \
Internet media types.  CoAP is designed to easily interface with HTTP \
for integration with the Web while meeting specialized requirements \
such as multicast support, very low overhead, and simplicity for \
constrained environments.\
"};

CoAP_Res_t* pAbout_Res = NULL;

static CoAP_HandlerResult_t ICACHE_FLASH_ATTR RequestHandler(CoAP_Message_t* pReq, CoAP_Message_t* pResp) {
    static uint16_t payloadSize = sizeof(CoapInfoStringInFlash)-1;

    CoAP_SetPayload(pReq, pResp, (uint8_t*)&(CoapInfoStringInFlash[0]), payloadSize, false);

    return HANDLER_OK;
}
```

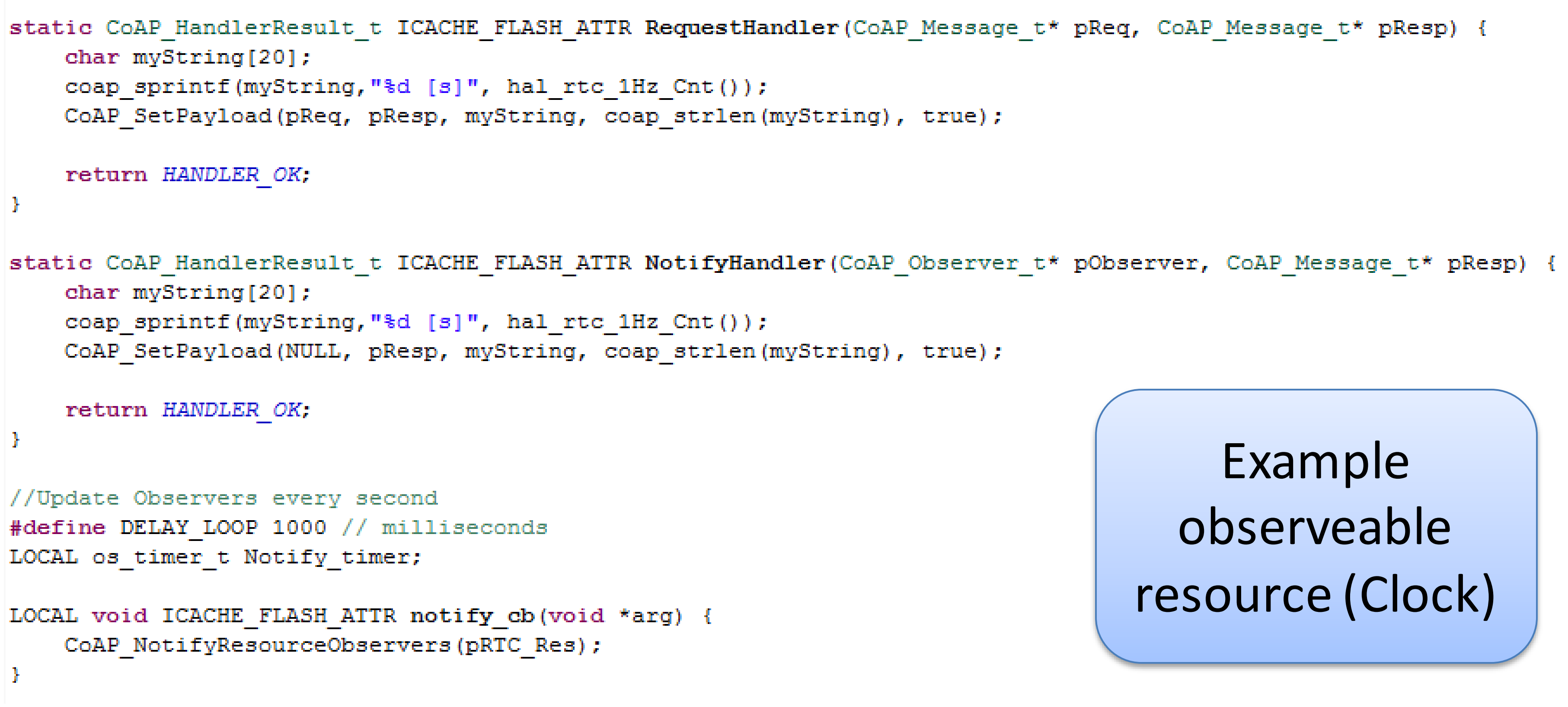
Example blockwise resource

CoAP: Postponed Request Handler

```c
static CoAP_HandlerResult_t ICACHE_FLASH_ATTR Res_ReqHandler(CoAP_Message_t* pReq, CoAP_Message_t* pResp) {
  if(pReq->Code == REQ_GET) {

      switch(ScanState){
          case SCAN_STATE_IDLE:
              wifi_station_scan(NULL,scan_done);
              ScanState = SCAN_STATE_RUNNING;
              return HANDLER_POSTPONE; //coap stack will comeback later

          case SCAN_STATE_RUNNING:
              return HANDLER_POSTPONE; //coap stack will comeback later

          case SCAN_STATE_DONE_OK:
              ScanState = SCAN_STATE_IDLE;

             pResp->Payload = pScanResultStr; //will be freed together with response msg, same for overwritten memory location
             pResp->PayloadLength = lastScanResultLen;

              //we don't allow non-atomic/blockwise transfers here and do some work to give client diagnostic payload
              CoAP_blockwise_option_t B2opt;
              if(GetBlock2OptionFromMsg(pReq, &B2opt) == COAP_OK) {
                if((int)B2opt.BlockSize < lastScanResultLen) {
                    pResp->Code=RESP_BAD_OPTION_4_02;
                    if(pResp->PayloadBufSize>16) {
                        coap_sprintf(pResp->Payload, "blk2 n/a [%d]",lastScanResultLen);
                        pResp->PayloadLength = coap_strlen(pResp->Payload);
                    }
                    return HANDLER_OK;
                }
              }

            return HANDLER_OK;

          case SCAN_STATE_DONE_FAIL:
            ScanState = SCAN_STATE_IDLE;
            return HANDLER_ERROR;
      }
  }
  return HANDLER_ERROR;
}
```

Example postponed resource

CoAP: Observeable Request Handler



```
static CoAP_HandlerResult_t ICACHE_FLASH_ATTR RequestHandler(CoAP_Message_t* pReq, CoAP_Message_t* pResp) {
    char myString[20];
    coap_sprintf(myString,"%d [s]", hal_rtc_1Hz_Cnt());
    CoAP_SetPayload(pReq, pResp, myString, coap_strlen(myString), true);

    return HANDLER_OK;
}

static CoAP_HandlerResult_t ICACHE_FLASH_ATTR NotifyHandler(CoAP_Observer_t* pObserver, CoAP_Message_t* pResp) {
    char myString[20];
    coap_sprintf(myString,"%d [s]", hal_rtc_1Hz_Cnt());
    CoAP_SetPayload(NULL, pResp, myString, coap_strlen(myString), true);

    return HANDLER_OK;
}

//Update Observers every second
#define DELAY_LOOP 1000 // milliseconds
LOCAL os_timer_t Notify_timer;

LOCAL void ICACHE_FLASH_ATTR notify_cb(void *arg) {
    CoAP_NotifyResourceObservers(pRTC_Res);
}
```

Example observeable resource (Clock)

Lobaro CoAP: Issues & dev direction

<u>Known-Issues</u>

- Implementing new transport bindings / ports is hard
- Client implementation too minimalistic (e.g. no blockwise receive

Lobaro CoAP: Issues & dev direction

Known-Issues

- Implementing new transport bindings / ports too hard
- Client implementation too minimalistic (e.g. no blockwise receive

Planned Improvements

- Built-in serial port transport binding (e.g. for device config)
- Include C Code in Go-lang Wrapper for better (unit) testing
- Use SMS Transport
- Integrate in FreeRTOS & use tickless kernel with periodic sleeps
- Use with our LoRa based Point to Point X-MAC variant
- Write integration guide to support opensource community

Industrial IoT Solutions

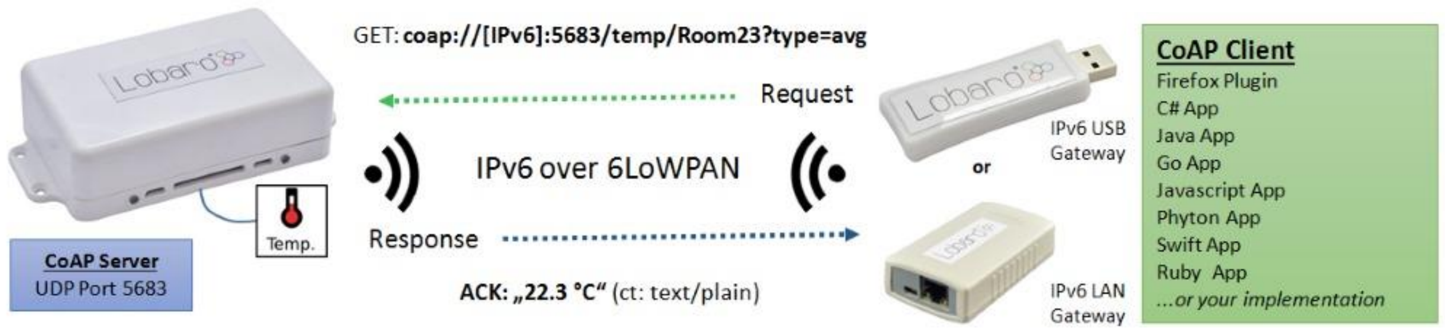
Thank you!

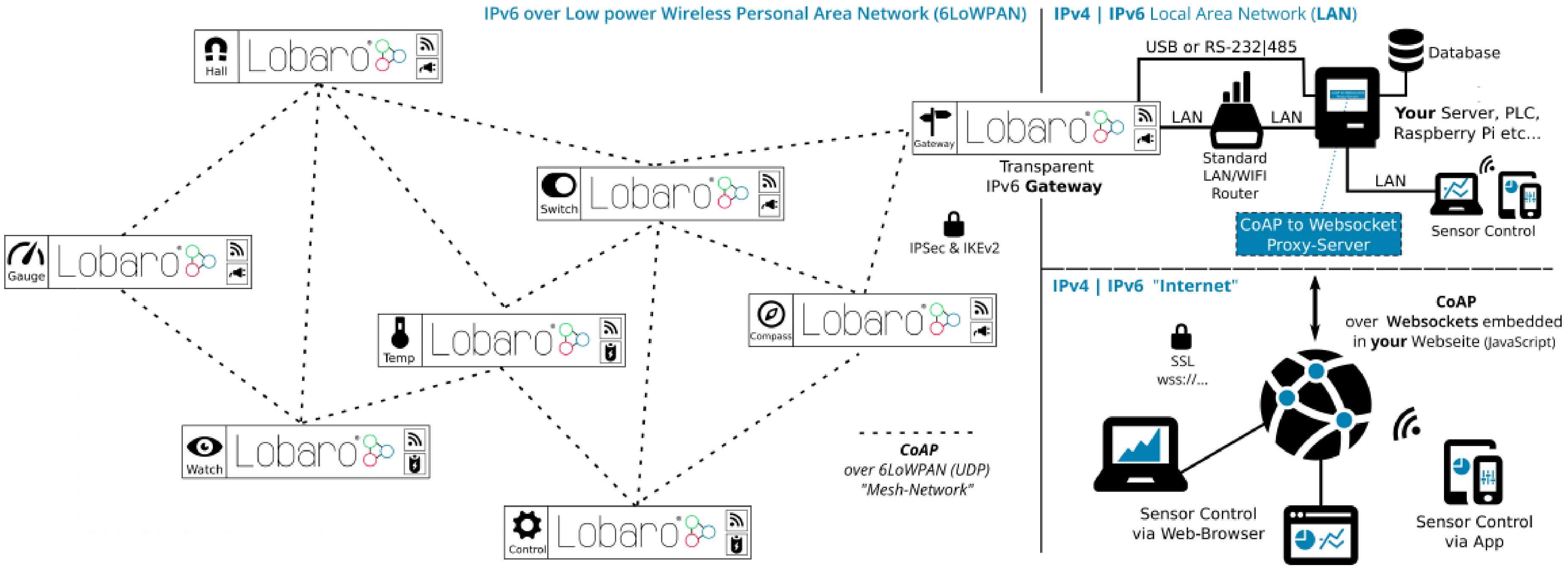
Questions?

Die Lobaro Technikdetails

Meshnetzwerk-Sensorik mit Lobaro-Boxen über IPv6 (6LoWPAN) und CoAP

Lobaro Industrial IoT Solutions


GET: coap://[IPv6]:5683/temp/Room23?type=avg
Request
IPv6 over 6LoWPAN
Response
ACK: „22.3 °C“ (ct: text/plain)
Temp.
CoAP Server
UDP Port 5683
IPv6 USB Gateway
or
IPv6 LAN Gateway
CoAP Client
Firefox Plugin
C# App
Java App
Go App
Javascript App
Phyton App
Swift App
Ruby App
...or your implementation


IPv6 over Low power Wireless Personal Area Network (6LoWPAN)
IPv4 | IPv6 Local Area Network (LAN)
Hall
Switch
Gauge
Temp
Compass
Watch
Control
Gateway
Transparent IPv6 Gateway
IPSec & IKEv2
CoAP
over 6LoWPAN (UDP)
"Mesh-Network"
USB or RS-232|485
LAN
LAN
Standard LAN/WIFI Router
Database
Your Server, PLC, Raspberry Pi etc...
LAN
Sensor Control
CoAP to Websocket Proxy-Server
IPv4 | IPv6 "Internet"
SSL
wss://...
CoAP
over Websockets embedded in your Webseite (JavaScript)
Sensor Control via Web-Browser
Sensor Control via App