CoAP - Sensor: Laboratory exercise

Tim Krämer, Jonas Müller, Distributed Systems SS24, Prof Böck

Fill in the missing code at the right locations, to play a CoAP based game!

Getting started with the project:

https://github.com/tim03-git/KraemerMueller_VS_CoAP.git

Preparing the python file, there might be the need to perform the following actions:

1. Import “aiocoap” python library to your system using “pip install aiocoap” or “python3 install aiocoap”
2. Change hardcoded IP inside the py file, try to find yourself where to do this!
   (Hint: It’s in the lower third of the program)

Preparing the CCS project, there might be the need to perform the following actions:

1. Adapt “SW_ROOT” and “SW_FREERTOS” under Project Properties > Build > Variables so that they point to your TivaWare folder, and to the “third party” folder inside TivaWare folder.

2. Next we need to perform some includes so the program knows the necessary functions and code

```
#include "mongoose.h"

#include "ADCGyro.h"

#include "CFAF128128B0145T/CFAF128128B0145T.h"
```

3. Now we need to create the mongoose main manager:

```
struct mg_mgr g_mgr;
```

4. Now we want to create some simple Output and logging functions, so we can se what is happening!

   ```
   UARTprintf("Protocol and IP: coap://%s/\r\n", ipaddr_ntoa((const
   ip_addr_t *) &ip));

   CFAF128128B0145T_text(10, 50, "IP:", CFAF128128B0145T_color_white,
   CFAF128128B0145T_color_black, 1, 1);

   CFAF128128B0145T_text(10, 60, ipaddr_ntoa((const ip_addr_t *) &ip),
   CFAF128128B0145T_color_cyan, CFAF128128B0145T_color_black, 1, 1);
   ```

5. Next we need to create a coap message structure, so we can handle incoming messages:

   ```
   struct mg_coap_message *cm = (struct mg_coap_message *) ev_data;
   ```

6. Now we should get the values from the Gyroscope:

   ```
   volatile int *GyroValues = ADCGyro();
   ```

7. Next, we want to prepare the piggybacked response:

   ```
   cm_resp.msg_id = cm->msg_id;

   cm_resp.msg_type = MG_COAP_MSG_ACK;

   cm_resp.code_class = 2;

   cm_resp.code_detail = 5;

   cm_resp.token.p = cm->token.p;

   cm_resp.token.len = cm->token.len;

   cm_resp.payload.p = payloadBuffer;

   cm_resp.payload.len = strlen(payloadBuffer);
   ```

8. We want to init lwIP next:

   ```
   lwIPInit(g_ui32SysClock, pui8MACArray, 0, 0, 0, IPADDR_USE_DHCP);
   ```

9. Now we finally init the mongoose manager:

   ```
   mg_mgr_init(&g_mgr, NULL);
   ```

10. Since we are using FreeRTOS, we need to create some tasks for for this too:

    ```
    xTaskCreate(vTaskDisplay, (const portCHAR *)"displaytask",
    configMINIMAL_STACK_SIZE, NULL, 1, NULL);
    ```

    ```
    xTaskCreate(vTaskMongoose, (const portCHAR *)"mongoose",
    configMINIMAL_STACK_SIZE, NULL, 1, NULL); // Mongoose Task new
    ```

11. Last but not least, we finally want to create a mongoose-CoAP connection, while assigning our CoAP handler to this connection:

    ```
    struct mg_connection *nc = mg_bind(&g_mgr, s_coap_address,
    coap_handler);
    ```