ICT for Industrial Applications – Lab Experience 15-17 May

Introduction

The goal of this experience is to make two development boards communicate with each other via MQTT and to make them communicate with a PLC, also via MQTT.

The experience simulates a scenario where a motor has a temperature sensor and the motor need to be turned off in case of overheating.

The experience starts from the communication between the two development boards and increases in complexity by adding components and features to the system.

Development boards

We are going to use two development boards:

- Arduino MKR WiFi 1010 (Figure 1) (https://docs.arduino.cc/hardware/mkr-wifi-1010/)
- ESP32 Pico Kit (Figure 2) (https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-pico-kit.html)

Figure 1: Arduino MKR WiFi 1010

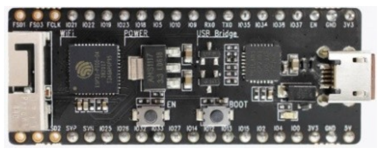

Figure 2: ESP32 Pico Kit

For what concerns the lab experience, the two development boards have the same features, except for the user-controllable LED that is missing on the ESP32 Pico. Pay also attention to the description of the code, there are some differences between the two development boards.

Hardware preparation

For the ESP32 Pico

Attach an LED to the development board using two female-female jumper wire. Connect the side with the resistor to one of the GND pin, the other must be connected to IO2.

For all boards

Attach the temperature sensor. We are using an MCP9701 sensor (https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP970X-Family-Data-Sheet-DS20001942.pdf). Attach the pins as in the following table (refer to page 2 of the MCP9701 datasheet):

Dev. board pin	MCP9701 pin
VCC or 3V3	V_{DD}
GND	GND
A1 or IO37	V_{OUT}

Please make sure that no metallic parts are touching each other!

Then connect the micro USB cable to the computer and the development board.

Software introduction

Both development boards will be programmed using the Arduino IDE. Be aware that some code is different between the two development boards.

Once opened the Arduino IDE, remember to:

- install the libraries ***ArduinoMqttClient*** **and** ***WiFiNINA*** using the library manager
- install the support for SAMD boards ***Arduino - SAMD Boards*** using the board manager

In the Arduino IDE, create a new project (sketch). Two sections of code are present: *setup* and *loop*. *setup* is used to initialise the board and will be run only once, before any *loop* running. *loop* is run continuously after setup. When *loop* finishes running, another instance of *loop* is run.

At the beginning of *setup* add the following to allow the development board to send debug messages to the computer with the Arduino IDE:

```
  //Initialize serial and wait for port to open:
  Serial.begin(115200);
  delay(10);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port
only
  }
```

Following this, you need to setup the LED port as outputs. Add the following to *setup*, depending on whether you have the Arduino or the ESP32:

```
  // set the LED pins as output
  pinMode(2, OUTPUT); // for ESP32
  WiFiDrv::pinMode(25, OUTPUT); // for Arduino, define RED LED
  WiFiDrv::pinMode(26, OUTPUT); // for Arduino, define GREEN LED
  WiFiDrv::pinMode(27, OUTPUT); // for Arduino, define BLUE LED
```

In *loop* you can add the following code to test that the development board is working:

```
  Serial.println("Ready");
  delay(1000);
```

In this way, every second, the string “Ready” will be printed to the Serial Monitor.

Arduino
On the dropdown beside the circular buttons, select “MKR WiFi 1010”. If this does not appear in the list, check the USB connection between the PC and the device.

ESP32
After connecting the device to the computer via USB, select “Select Other Board and Port”. On the window that appears, select “ESP32 PICO-D4” on the left and /dev/ttyACM0 or /dev/ttyUSB0 on the right.

All devices
Compile and load the code to the device and open the Serial Monitor from the Tools menu. Make sure the serial speed has been selected as 115200. You should be able to see the string appearing.

Connecting to WiFi

Arduino
The WiFiNINA library is used to connect to WiFi.

Documentation: https://docs.arduino.cc/tutorials/communication/wifi-nina-examples/

At the very top of your code, add

```
#include <WiFiNINA.h>
```

Then store the WiFi SSID and network you want to connect to:

```
char ssid[] = "RETITLC";
char pass[] = "rdclab2024";
```

Create a WiFiClient object and use it to connect to the WiFi network.

ESP32
The ESP32 board support package includes the drivers for WiFi.

Documentation:
https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html

At the very top of your code, add

```
#include <WiFi.h>
```

Then store the WiFi SSID and network you want to connect to:

```
char ssid[] = "RETITLC";
char pass[] = "rdclab2024";
```

Create a WiFiClient object and use it to connect to the WiFi network.

Connect to the MQTT broker

The ArduinoMQTTClient library is used to communicate via MQTT. The IP address of the MQTT server is 192.168.10.24

Documentation: https://www.arduino.cc/reference/en/libraries/arduinomqttclient/

Add the following at the top of the file, immediately following the WiFiNINA include:

```
#include <ArduinoMqttClient.h>
```

Use the MqttClient object to communicate via MQTT.

Each group has been assigned a numeric id (group_id). An id is also assigned to each subgroup (subgroup_id, which can be A or B). Now use MQTT to send a message to the topic lab/*group_id*/*subgroup_id*/ping. Also, subscribe to the topic of your other subgroup and listen for messages.

Send and receive temperature data

The temperature sensor provides an output voltage that is proportional to its temperature.

1. Use the *analogRead* function to read the voltage value from the sensor: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/
2. As explained in Section 4.0, page 10 of the sensor datasheet, convert the voltage value to temperature in degrees Celsius: $T_A=(V_{OUT}-V_0)/T_C$. Use the following values: $V_0=0.4$ V, $T_C=0.0195$ V.

Output the value to both the Serial Terminal and MQTT. For MQTT, use the topic lab/*group_id*/*subgroup_id*/temperature (e.g., lab/A/0/temperature) while the payload should only contain the temperature value (i.e., do not use JSON).

Receive the temperature values from the other subgroup and output it to the Serial Terminal. Also, check that the temperature value is under a given threshold and send the output of this check via MQTT, to the topic lab/*group_id*/*subgroup_id*/temperature_status (e.g., lab/A/0/temperature_status). The payload must be 0 if the temperature is under the threshold, 1 otherwise. In addition, output the same message to the topic lab/PLC/*led_id*/temperature_status to act on the PLC LED. *led_id* will be assigned to each group during the lesson.

Plot the temperature data

You can use Grafana to receive the MQTT data and create a dashboard showing a plot of the temperature value and status.

The MQTT plugin for Grafana can be installed directly from inside Grafana. Documentation is at https://grafana.com/grafana/plugins/grafana-mqtt-datasource/.