Software Technology of Internet of Things

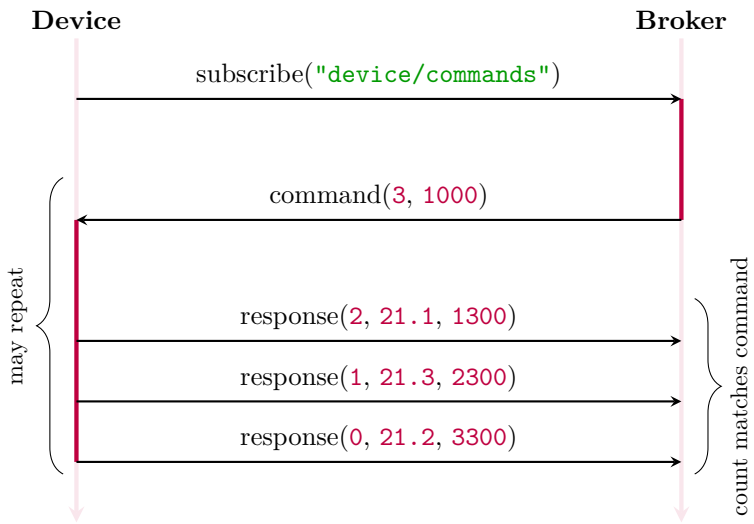
Portfolio Handin #2: Temperature Publication

Aslak Johansen <asjo@mmmi.sdu.dk>

April 8, 2024

Design and implement a program for your ESP32 device which is capable of periodically publishing temperature readings from an analog LMT86 temperature sensor to an MQTT broker.

It should be testable using the following protocol:



There are four phases:

- ***Phase 0 - Initialization*** First the device needs to initialize. This includes connecting to a WiFi network.
  Note: This phase is not shown in the above figure.
- ***Phase 1 - Subscription*** The device connects to an MQTT broker and subscribes to a specific topic. This topic is compile-time configurable.
- ***Phase 2 - Command Reception*** Over this topic, the device will receive a command containing two values. The first value represents how

many measurements to make, and the last represents how much time they should be apart[1]. In case of 3 repetitions that are 1000 ms apart, the payload of the received message will look like this (quotes excluded): `"measure:3,1000"`.

- ***Phase 3 - Response*** Once the device receives the command, it will immediately start executing it. To do so, it samples an analog temperature sensor using the built-in ADC, and converts the raw value to a temperature. The resulting floating point value is published to a *response* topic[2] along with the remaining number of messages and the current uptime[3] of the device in ms. Following the example from phase 2, three messages will be produced. The structure of each message is three values (remaining number of messages, temperature and uptime) separated by commas. Their payloads could look like this (quotes excluded):
  - `"2,21.1,1300"`
  - `"1,21.3,2300"`
  - `"0,21.2,3300"`

It is guaranteed that at most one command is active at any point in time.

Details

Your job is to program the device side of this system and make sure that it inplements this protocol. In particular:

1. ***Analog-to-Temperature Conversion*** from analog temperature sensor.
   - https://embeddedexplorer.com/esp32-adc-esp-idf-tutorial/
   - Consider using the following functions to get the voltage:
     - `esp_adc_cal_characterize`
     - `esp_adc_cal_raw_to_voltage`
   - Find a voltage to temperature conversion function in the sensor datasheet.
2. ***Dynamic Configuration*** Define, in the `main/Kconfig.projbuild` config file, the following definitions (and make sure that your code is informed by them):
   - `WIFI_SSID` The SSID to connect to. *[string]*
     Example: `SkyNet`
   - `WIFI_PASSWORD` The password to be used for connecting to the WiFi network. *[string]*
     Example: `"Summer"`

[1] This should be considered an exact average period. So, the more measurements the command calls for, the closer to this period one should get.
[2] This topic is also compile-time configurable.
[3] The uptime is the time since last boot.

- `MQTT_BROKER` The MQTT broker to connect to (without auth). *[string]*
  Example: `"mqtt://broker.hivemq.com"`
- `MQTT_COMMAND_TOPIC` The topic on the MQTT broker to subscribe to. *[string]*
  Example: `"org/sdu/2024/iot/my/name/command"`
- `MQTT_RESPONSE_TOPIC` The topic on the MQTT broker to publish to. *[string]*
  Example: `"org/sdu/2024/iot/my/name/response"`

3. ***Static Configuration*** The following configurations will – for convenience reasons – be fixed for the exercise.
   - `ADC_CHANNEL` You should use `ADC_CHANNEL_6` (i.e., pin 34).
   - `ADC_UNIT` You should use `ADC_UNIT_1`.

As documentation for how the functionality of this will be evaluated, you will soon see a Livebook notebook detailing the testing procedure on itslearning. An announcement will be made when it is available.

Note: You will not be expected to write or execute this elixir code, but you should read the part about how you code will be build, configured and flashed.

Handin Form

A report must be written and handed in as a single PDF file. You only have a single page for the main text, but you can put as many things as you like in appendices. So, this is where you should place your figures. The LaTeX template provided on itslearning in the portfolio plan must be used. The report should cover:

- An description of the (software) steps involved in converting a physical temperature to a floating point value representing that temperature in degree centigrade. This should be done at a level of detail that would allow someone else to reproduce code equivalent to yours.
- A description of the full flow of the program, including the event dispatch mechanism. This could involve a diagram. Again, this should allow someone else to implement functionally equivalent code. It should also provide insights to how the dispatch mechanism function.
- An argument for why the publication period is achieved[4].

Note: This report relates to a subset of the learning objectives of the course, and it is your opportunity to demonstrate fulfillment of them (to the degree expressed in this document).

In addition, your code must be handed in as a plain ZIP archive. It must be possible to configure and build the code using the `esp-idf` toolchain. See the previously mentioned Livebook notebook for an executable description of how the toolchain should be buildable.

[4] This should cover why it doesn't drift.

Programming Languages

The device code should be written in C/C++ and be buildable using the esp-idf toolchain without the use of Arduino libraries. Please ask your course responsible (Aslak Johansen) if you would like to deviate from this by using a different language[5]. In that case, you will be required to add an appendix detailing how to build and deploy the project, and how the dynamic configuration is implemented.

Do consider, that choosing a different language may represent a significant increase to your workload without adding to your ability to demonstrate fulfillment of your learning objectives[6].

How to Handin?

Hand in using `https://digitaleksamen.sdu.dk`. The deadline is May 13, 2024.

Note: This is an individual handin.

[5] Rust and Zig are pre-approved. Due to its stop-the-world garbage collector, Go will not be approved.

[6] It might be more interesting though.