TECHNISCHE UNIVERSITÄT WIEN

Wireless in Automation
183.632 (VU 3,0) Semester: 2022S

Smart Thermostat Proof of Concept

June 21, 2022

Student data

Matrikelnummer	Firstname	Lastname
01525189	Christoph	Lehr

1 Project Description

In this Proof of Concept (PoC) we want to build a first demo to utilize the new Thread network. As an initial project an smart thermostat with a set of sensors shall be designed.

1.1 Zephyr and OpenThread

Zephyr is a Real-Time Operating System (RTOS) which is free and open-source. It provides an abstraction layer to the hardware to simplify the access to sensors and other connected devices. Additionally, it provides a network stack and implements the open source Thread network stack called OpenThread. Thread is a low power wireless network protocol targeted for devices with constrained resources.

1.2 Sensors

In the current state of the project a set of sensors is used which are described in this section. The *Grove - Passive Infrared (PIR) Motion Sensor* is a passive sensor which detects movement via an infrared lens and sets its output accordingly. The *Grove - Light Sensor* is a indoor luminance sensor.

Additionall, the *Grove - Temperature Humidity Pressure Gas Sensor(BME680)* is used, it provides air quality, air pressure, humidity and temperature readings. The air quality reading is exported as a resistance which can be converted to the Air Quality Index (AQI).

1.3 Display

To provide some simple output and *SSD1306* Organic Light Emitting Diode (OLED) display is used. It provides an Inter-Integrated Circuit (I2C) bus interface to be interacted with. It has a resolution of 64x128 pixels.

1.4 Constrained Application Protocol (CoAP) Introduction

Constrained Application Protocol (CoAP) is a protocol that was designed to be used in constrained devices and builds up on User Datagram Protocol (UDP) as its transport layer protocol. It was designed to be easily translatable to Hypertext Transport Protocol (HTTP) which simplifies the interaction with internet services.

A CoAP server exposes resources which can be interacted with HTTP like requests. A *GET* request returns the resource, if it exists. With *PUT* requests resources can be modified and with a *POST* request can be use to create new resources or modify an existing resource. All three types return the value of the resource. An *DELETE* request requests to delete a resource. Additionally, to requesting resources, a client can also register itself as an observe of said resource. Depending on the configuration of the server, the client receives a notification everytime the value of the resource changes. Each resource of a server can be accessed via an Uniform Resource Identifier (URI) similar to MQ Telemetry Transport (MQTT). Unlike mqtt wildcards in the URI paths are not allowed, but depending on the used CoAP implementation may be available.

URI	Method	Resource Description
`echo`	`PUT`	Takes the provided payload and returns it to the sender
`sensors/temperature`	`GET`	Returns the last read temperature value
`sensors/humidity`	`GET`	Returns the last read humidity value
`sensors/air_quality`	`GET`	Returns the last calculated AQI value
`sensors/air_pressure`	`GET`	Returns the last read air pressure value
`sensors/presence`	`GET`	Returns the last read PIR sensor value
`sensors/luminance`	`GET`	Returns the last read luminance value

Table 1: Server CoAP resources

2 PoC Design

The system is split between two devices the sensor unit and the thermostat. The sensor unit takes over collecting data and the thermostat takes over using this data to control the environment.

2.1 Sensor Unit

As the name already implies the sensor unit has connected the previous listed sensors connected to it. It has two main modules, the CoAP server and the sensor module. Each runs on its own thread.

The CoAP server module binds a UDP socket for the port 5683. It exposes the following sensor values with the URIs-paths defined in Table 1.

In the sensor module each sensor gets read out every 5 seconds. If a PIR sensor changes its state, an interrupt will be executed, which triggers an immediate re-sampling of all the sensors. If the integer part of a sensor value between two samples changes the sensor module triggers the sending of a CoAP notification.

2.1.1 Project directory

- `prj.conf`: Default project configuration file
- `overlay-ot.conf`: OpenThread configuration file
- `CMakeLists.txt`: `cmake` file to define source files
- `build.sh`: Build script
- `nrf52840dk_nrf52840.overlay`: Device overlay which defines pins and sensor interfaces
- `src`: Source directory for the source and include files

2.2 Thermostat

The thermostat consists of three modules, the CoAP client, the Heating Ventilation and Air Conditioning (HVAC) module and the display module.

The thermostat unit shall connect to the CoAP server and start observing the servers air quality, humidity, presence and temperature resources. With the values gathered, it starts acting on the environment in order to keep the temperature in a desired range. This range is dependent whether someone is present or not. Additionally, it shall start acting if the humidity or the AQI exceeds a certain maximum value. Additionally, the thermostat shall display the current sensor values, the interface is depicted in Figure 3.

2.2.1 Project directory

- `prj.conf`: Default project configuration file
- `overlay-ot.conf`: OpenThread configuration file
- `overlay-display.conf`: Display configuration file

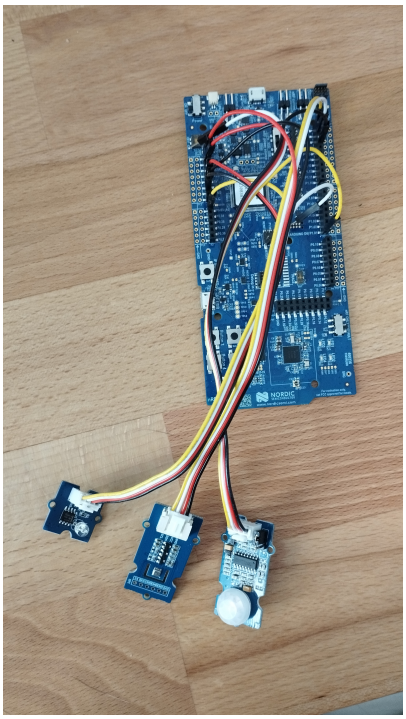

Figure 1: Wired up sensor unit

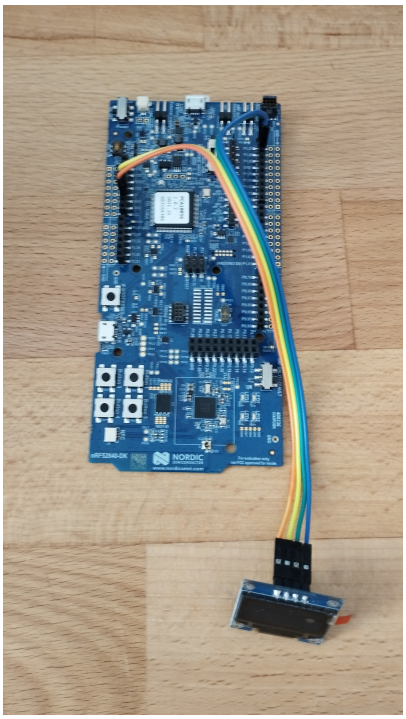

Figure 2: Wired up thermostat

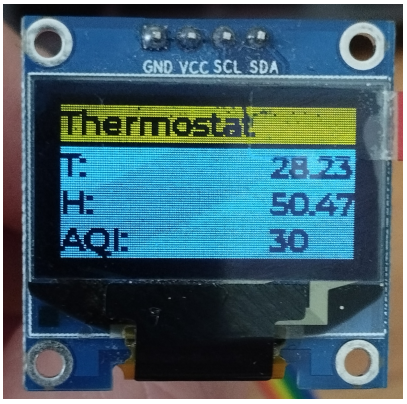

Figure 3: Thermostat Display

- `CMakeLists.txt`: `cmake` file to define source files
- `build.sh`: Build script
- `dummy_dc.overlay`: Device overlay for the display
- `src`: Source directory for the source and include files

3 Building

For both devices there exists a `build.sh` file which simplifies the building and sets all required variables. In general this executes Zephyrs build command `west build`, sets the used board variable for the nRF52840 DK as follows `-b nrf52840dk_nrf52840`. Furthermore the additional configuration files will be set. As the thermostat also contains a display, the corresponding shield variable is also set.

The full command for the sensor unit looks as follows: `west build -b nrf52840dk_nrf52840 - -DCONF_FILE="prj.conf overlay-ot.conf"`.

The full command for the thermostat looks as follows: `west build -b nrf52840dk_nrf52840 - -DCONF_FILE="prj.conf overlay-ot.conf overlay-display.conf" -DSHIELD=ssd1306_128x64`

4 Wiring

Figure 1 shows how the sensor unit shall be wired up, the detailed pin connection are depicted in Table 2. Figure 2 shows how the sensor unit shall be wired up, the detailed pin connection are depicted in Table 3.

5 Running the Demo

To run the demos first execute the build script in the corresponding device directories and then run `west flash` to download the project to the development board. Directly after flashing the demo starts running.

Pin nRF52840 DK	**Pin Device**
Grove - Temperature Humidity Pressure Gas Sensor(BME680)	
P0.26	SDA
P0.27	SCL
VDD	VCC
GND	GND
Grove - PIR Motion Sensor	
P1.01	SIG
VDD	VCC
GND	GND
Grove - Light Sensor	
P0.03	SIG
VDD	VCC
GND	GND

Table 2: Wiring for the Sensor Unit

Pin nRF52840 DK	**Pin Device**
I2C SSD1306 OLED Display	
P0.26	SDA
P0.27	SCL
VDD	VCC
GND	GND

Table 3: Wiring for the Thermostat

5.1 Thermostat

First the thermostat starts to initiate a connection to the server by sending *PUT* request to the servers `echo` resource. As the setup of the OpenThread network takes a moment and until both devices can reach each other, some requests will be dropped. As soon as the thermostat gets a response, it starts setting up its observers for the presence detection as well as the sensor values for temperature, humidity and air quality. During this procedure the HVAC module gets set up with parameters provided during build time and the display gets initialized. When the initialization of the CoAP client, the HVAC and the display is done, the thermostat waits for updates from the sensor units and displays temperature, humidity and the AQI, see Figure 3. The HVAC module then sets its outputs, mimicked here with the Light Emitting Diodes (LEDs) of the board. LED1 indicates that the heating is active, LED0 indicates that cooling is active and LED3 indicates whether venting is active. As one probably assumes, heating will be enabled when the temperature falls below a certain point and cooling will be enabled if the temperature exceeds a certain point. Venting will be enabled if the AQI or the humidity exceed a certain threshold. Default values are

- Minimum temperature: 24°C
- Maximum temperature: 28°C
- Minimum temperature when someone is present: 25°C
- Maximum temperature when someone is present: 27°C
- Maximum humidity: 75%
- Maximum AQI: 100

5.2 Sensor Unit

The sensor unit start with initializing the CoAP server and setting up the sensors. After the initial setup, the server periodically reads out the single sensor values. Now the server waits until a client wants to observe any of

its resources. If any sensor value changes in the integer range, the updated value will be sent to its observers.

6 Further Topics

Currently the PoC consists of two devices, with the current sensor setup the luminance sensor is read out but not utilized yet. The demo could be extended with a third device which use this sensor to control lighting and shading.

If the sensor unit is restarted, the thermostat needs to be restarted to as there is currently no mechanism implemented to check if the server is lost. The thermostat can further send echo request to check whether the server is still active, like a heartbeat signal. The client then restarts its state an newly registers its observers at the server.

Currently the configuration values are not configurable, the thermostat should either provide some resources for commissioning or enable a Bluetooth and create a service which sets the data.

At the moment, the implementation uses sockets and UDP, but there is also a Lightweight M2M (LWM2M) implementation provided by Zephyr on top of OpenThread. This may be better for this usecase.

7 Pitfalls

Reading out the PIR sensor value in the interrupt context causes I2C error and prohibits readings of the BME680.

Executing `sprintf` or `snprintf` with a `float` or `double` leads to a buffer overflow.