**Technical & Functional analysis**

**Breakout game with gyroscope controller using Thread/Matter**

|  |  |
| --- | --- |
|  |  |

Date : 12/7/2023

Contacts : Dekimo : Gysen Bart

Table of Contents

[1. Project overview 4](#_Toc152840114)

[1.1. Matter protocol 4](#_Toc152840115)

[1.2. OpenThread network protocol 4](#_Toc152840116)

[2. Game controller 5](#_Toc152840117)

[2.1. EFR32MG24 - BRD2601B Dev Kit 5](#_Toc152840118)

[2.2. EFR32MG24 Wireless SoC 6](#_Toc152840119)

[2.3. ICM-20689 6-Axis sensor 7](#_Toc152840120)

[2.4. User Button 8](#_Toc152840121)

[2.5. HAL Config 9](#_Toc152840122)

[3. SBC 9](#_Toc152840123)

[4. Thread 9](#_Toc152840124)

[4.1. Rights – OpenThread – Thread 9](#_Toc152840125)

[5. Matter protocol – specification 10](#_Toc152840126)

[5.1. Terms 10](#_Toc152840127)

[5.2. About 10](#_Toc152840128)

[5.3. Documents 10](#_Toc152840129)

[5.4. Architecture 11](#_Toc152840130)

[5.4.1. Network 11](#_Toc152840131)

[5.4.2. Device Data Model 14](#_Toc152840132)

[5.5. Matter Hub & Brigde 18](#_Toc152840133)

[6. Silabs Matter SDK 18](#_Toc152840134)

[6.1. Locations 18](#_Toc152840135)

[6.2. Custom clusters 18](#_Toc152840136)

[7. Issues 18](#_Toc152840137)

[7.1. Gyroscope sensor problem 18](#_Toc152840138)

[7.1.1. Possible solution 18](#_Toc152840139)

[7.2. Connection between matter device (using chip-tool) and the “game” 19](#_Toc152840140)

[7.2.1. Possible solution 19](#_Toc152840141)

[8. Tests 20](#_Toc152840142)

[8.1. Simple on/off device 20](#_Toc152840143)

[8.2. Level control test 23](#_Toc152840144)

[8.2.1. Capture data using chip-tool 25](#_Toc152840145)

[9. Notes 26](#_Toc152840146)

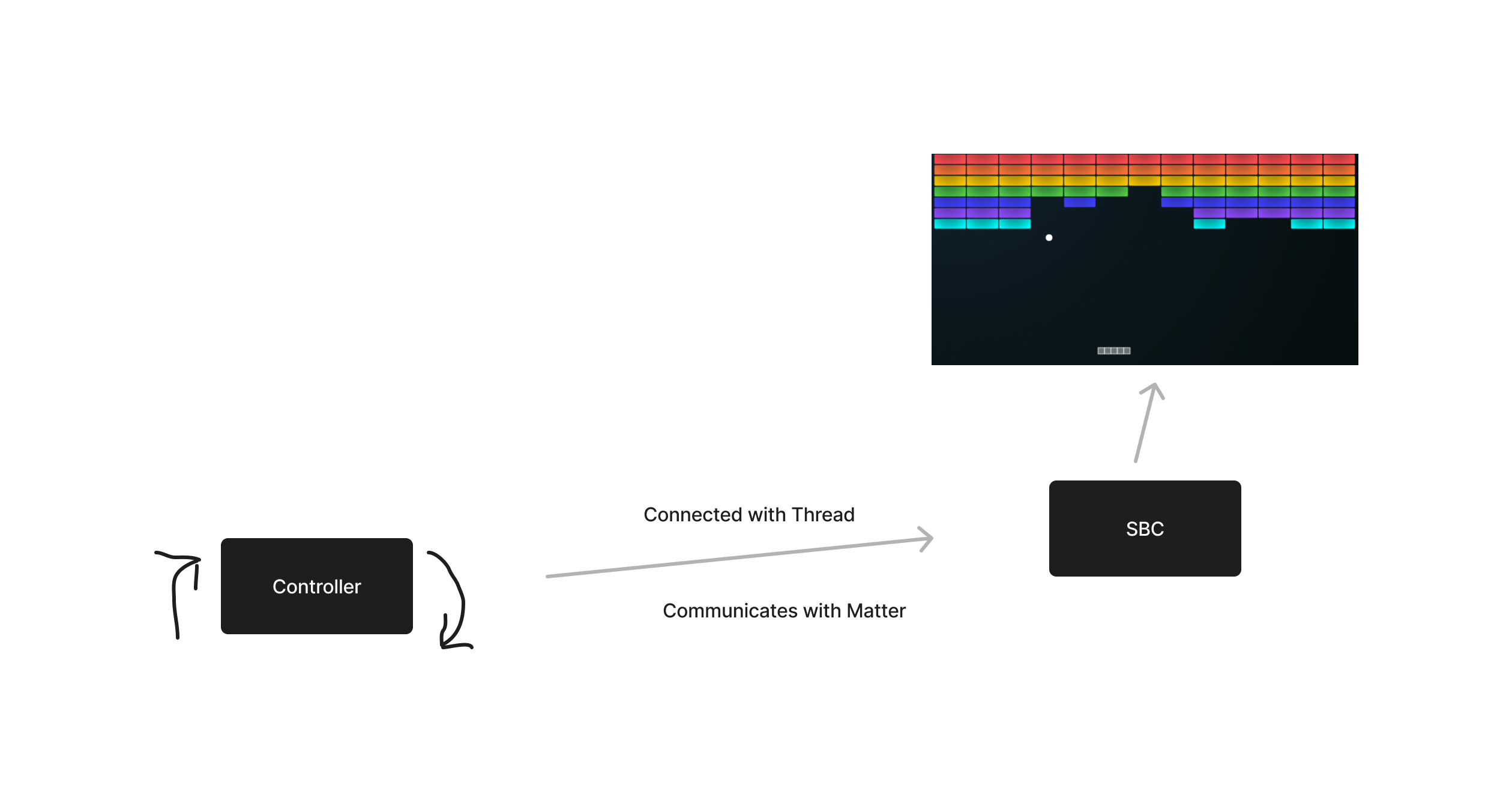
[10. Document Information 27](#_Toc152840147)

[10.1. Version History 27](#_Toc152840148)

[10.1. Related Documents 27](#_Toc152840149)

# Project overview

This project is an interactive breakout game controlled by motion with a wireless controller. The controller is connected to a Thread network and communicates with the Matter protocol. The breakout game is displayed on a tv using a SBC (Single Board Computer) that is also connected to a Thread network.



## Matter protocol

Matter, formerly known as “Project CHIP” (Connected Home over IP) is an open-source, royalty-free connectivity standard designed to make it easier for various smart devices to communicate with each other. It aims to create a unified standard for the Internet of Things devices. A Matter device can connect to a network with Thread, WIFI and Ethernet.

## OpenThread network protocol

OpenThread is an low-power mesh networking protocol that is based on IPv6. OpenThread is a open standard and is built for IoT applications. It uses 6LoWPAN which uses IEEE 802.15.4 (2.4Ghz) wireless protocol with mesh communication.

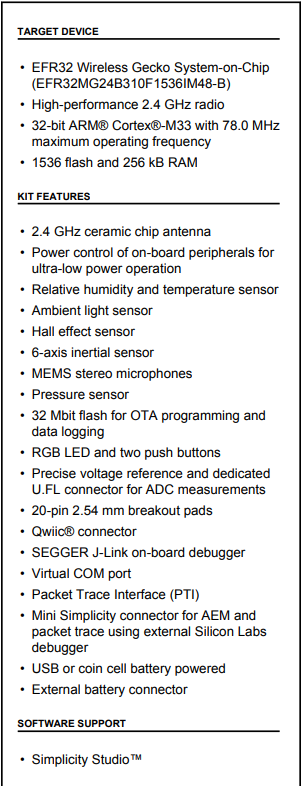
# Game controller

The wireless controller must be equipped with:

* 2.4Ghz wireless controller used for the OpenThread connection
* Accelerometer used to interact with the game (moving the bar)
* Button used for interact with the game (pause, start, etc.)
* A battery to provide power to the controller

## EFR32MG24 - BRD2601B Dev Kit

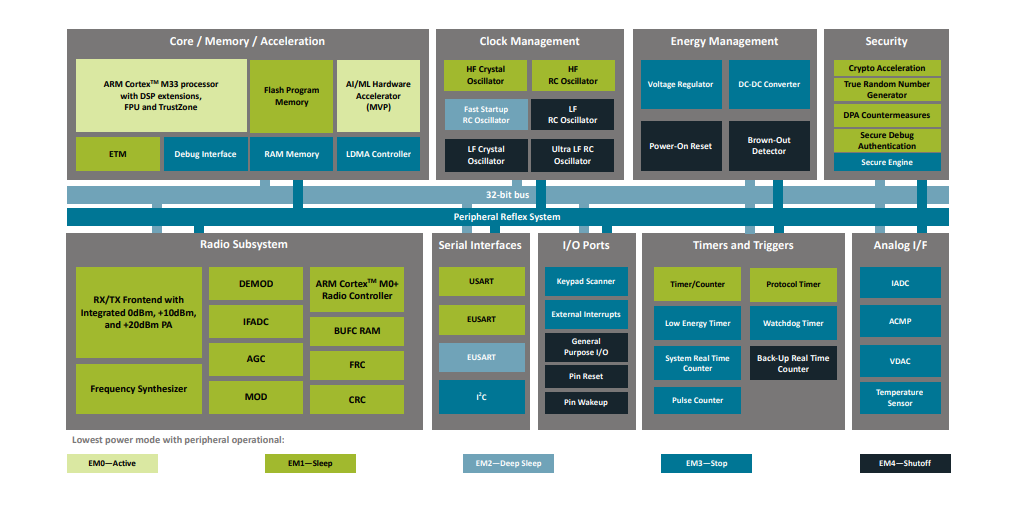
The EFR32MG24 Dev Kit board (BRD2601B) supports al the elements that is necessary for the controller. So there is no need to develop a PCB with al this components.

## EFR32MG24 Wireless SoC

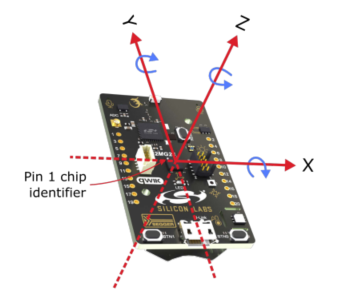
The development board uses the EFR32MG24 wireless SoC (System on a Chip). This SoC is ideal for meshing wireless solutions using Matter and OpenThread. It provides al the built in features that are relevant for this project, like:

* High performance 32-bit 78 MHz Arm Cortex®-M33 processor
* 1536 kB flash and 256 kB of RAM
* High performance 2.4GHz Radio
* OpenThread and Matter support
* Security features for protection against hardware and software attacks
* A wide range of peripherals like I²C, SPI, USART, ADC, Timers, GPIO’s, etc.

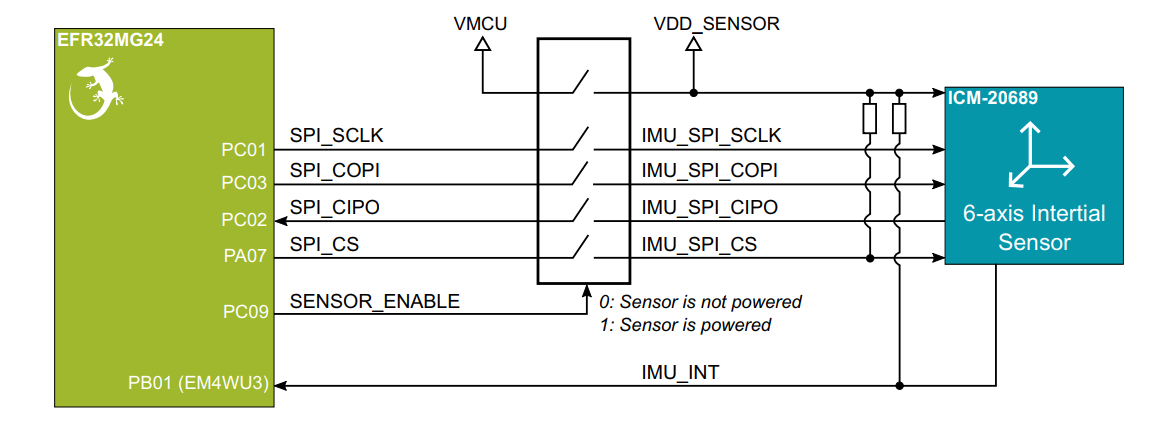


## ICM-20689 6-Axis sensor

The development board contains a 6-axis sensor ICM-20689. This 6-axis sensor combines a 3-axis gyroscope and a 3-axis accelerometer. It detects acceleration and angular rate in the X, Y and Z axes.

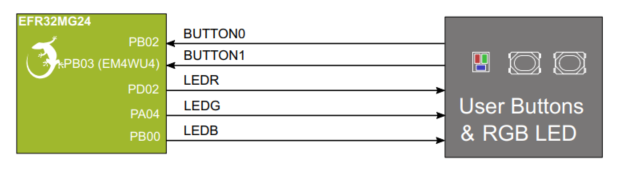


In the development board the sensor is connected an communicates over SPI. The SPI lines are interrupted trough a switch to prevent power consumption when not used. Before the sensor can be used in the application it must be enabled by setting PC09 high.

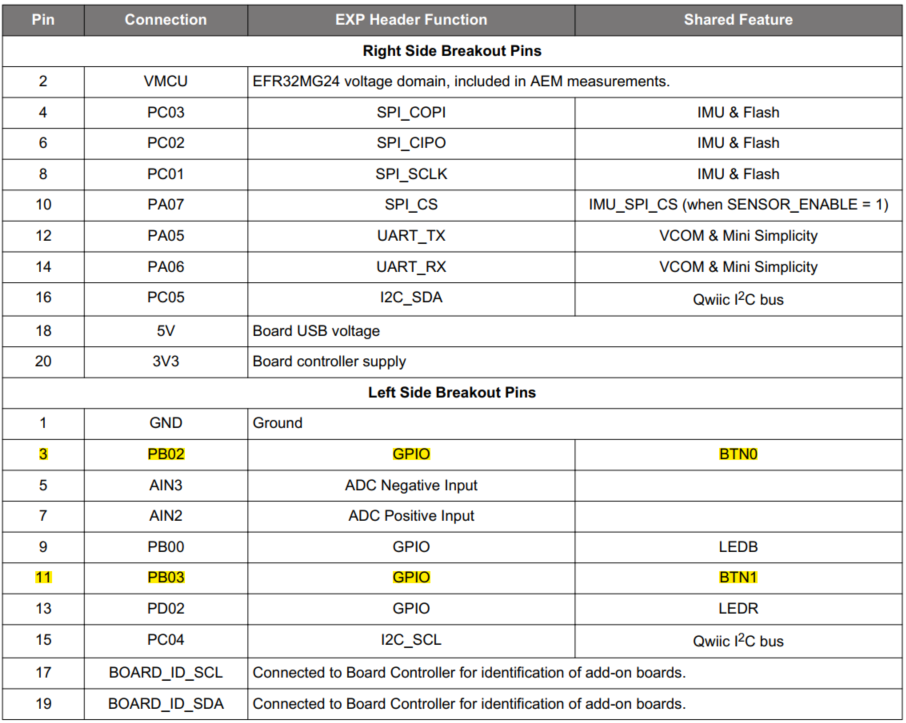


## User Button

The development board has two user buttons, BTN0 and BTN1. They are directly connected to the EFR32MG24 SoC and are debounced by RC filters.



The buttons are connected to the pins PB02 and PB03. The pins are also available on the expansion header. The buttons on the board can be used during the development and later on we can use buttons in a case that are connected to the expansion header.



## HAL Config

In the Matter project you can find the configuration file where all the onboard sensors is listed in. you can find this under config/sl\_board\_control\_config.h.

~~To enable the gyroscope we have to set the define SL\_BOARD\_ENABLE\_SENSOR\_IMU~~

To enable the gyroscope we have to install the software component in the gecko SDK.

# SBC

# Thread

## Rights – OpenThread – Thread

**Needs to modify in own words**

What Is OpenThread?

OpenThread is an open-source implementation of the Thread networking protocol technology developed and released by Google and maintained on GitHub.

Close Answer

What Is The Difference Between Thread And OpenThread?

OpenThread released by Google is an open-source implementation of Thread technology. Developers who choose to implement OpenThread in shippable products must join the Thread Group in order to certify those products and market them as Thread-certified.

Close Answer

Is An End-Product Based On OpenThread Automatically Thread-Certified?

No. If a company uses OpenThread to build a product, they need to be a member of the Thread Group in order to gain the Intellectual Property (IP) rights to ship Thread products and to complete product certification, which ensures that products using Thread work together effortlessly and securely right out of the box.

Close Answer

What Would Prevent A Company From Shipping A Product Based On OpenThread Without Joining The Thread Group?

If developers choose not to join Thread Group and ship products using Thread technology, they are not conferred the IP rights required to practice and ship Thread technology, and may subject themselves to legal action, including but not limited to licensing fees.

# Matter protocol – specification

## Terms

In order to understand this Matter document we need to know the definitions of several terms.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Device | A piece of equipment containing one or more Nodes |
| Node | An addressable entity which supports the Matter protocol stack and has its own Operational Node ID and Node Operational credentials. A Device may host multiple Nodes. |
| Endpoint | A particular component within a Node that is individually addressable. |
| Fabric | A logical collection of communicating Nodes, sharing a common root of trust,and a common distributed configuration state. |
| Cluster |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## About

Vertel hier kort wat Matter is, wie het ontworpen heeft. Voor wie het bedoeld is, wie ermee aan de slag kan en hoever ze met de ontwikkeling zijn etc…

## Documents

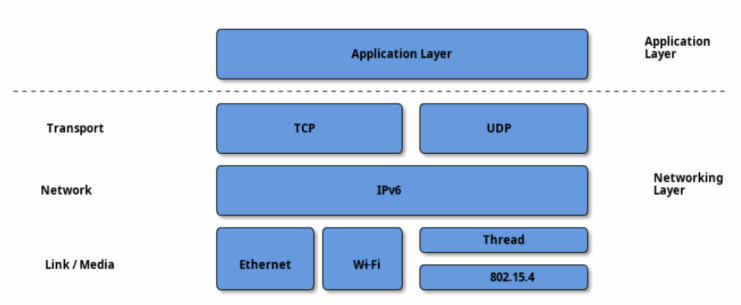
The Matter protocol is specified in Three main documents. These documents are very useful when developing a product. With every update these documents changes so make sure when developing a product you get the latest documents.

* **Matter-(version)-Core-Specification**In this document you can find all about the features of the protocol, the requirements, technical specifications etc.
* **Matter-(version)-Device-Library-Specification**This document tells which device-types are included in this specific version, it also tells what clusters can and must be used with a specific device-type.
* **Matter-(version)-Application-Cluster-Specification**This document tells all the available clusters in this specific version. It tells what attributes and commands are part of a clusters and which attributes and commands are mandatory or optional for a specific cluster.

## Architecture

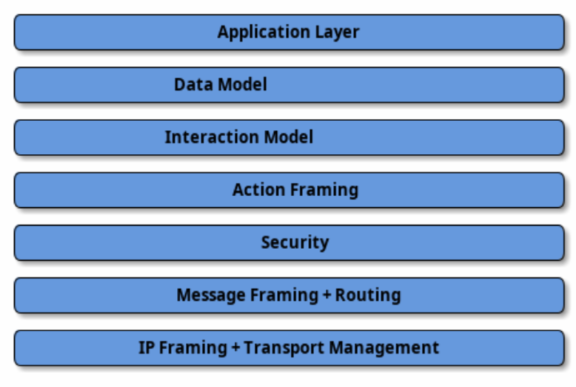
### Network

Matter is a universal IPv6-based communication protocol designed for smart home and the Internet of Things devices. Matter works on the Application Layer of the Network Stack.



Matter Protocol Stack

The Matter protocol stack is divided into layers to separate the different responsibilities and introduce a good level of encapsulation amongst the various parts of the protocol stack.



* **Application Layer:** High-order business logic of a device. For example, an application that is focused on lighting might contain logic to handle turning on/off the light as well as its color characteristics.
* **Data Model:** The data layer corresponds to the data and verb elements that help support the functionality of the application. The Application operates on these data structures when there is an intent to interact with the device.
* **Interaction Model:** The Interaction Model layer defines a set of interactions that can be performed between a client and server device. For example, reading or writing attributes on a server device would correspond to application behavior on the device. These interactions operate on the elements defined at the data model layer.
* **Action Framing:** Once an action is constructed using the Interaction Model, it is serialized into a prescribed packed binary format to encode for network transmission.
* **Security:** An encoded action frame is then sent down to the Security Layer to encrypt and sign the payload to ensure that data is secured and authenticated by both sender and receiver of a packet.
* **Message Framing & Routing:** With an interaction encrypted and signed, the Message Layer constructs the payload format with required and optional header fields; which specify the message's properties and some routing information.
* **IP Framing & Transport Management:** After the final payload has been constructed, it is sent to the underlying transport protocol for IP management of the data.

Network Topology

In principle, any IPv6 network is suitable for Matter deployment butt the focus is on three link layer technologies: Ethernet, Wi-Fi and Thread.

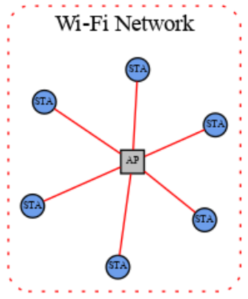
Matter treats networks as shared resources: it makes no stipulation of exclusive network ownership or access. As a result, it is possible to overlay multiple Matter networks over the same set ofconstituent IP networks.

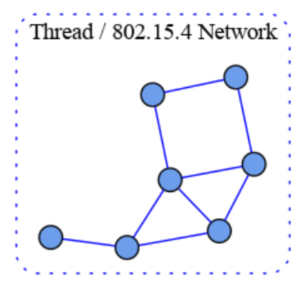
This protocol may operate in the absence of globally routable IPv6 infrastructure. This requirementenables operation in a network disconnected or firewalled from the global Internet. It also enablesdeployment in situations where the Internet Service Provider either does not support IPv6 on consumer premises or where the support proves otherwise limiting.

This protocol supports local communications spanning one or more IPv6 subnets. Canonical networks supporting a fabric may include a Wi-Fi/Ethernet subnet, or one or more low power and lossy network (LLN) subnets. In this version of the specification, Thread is the supported LLN standard.

**Single network**

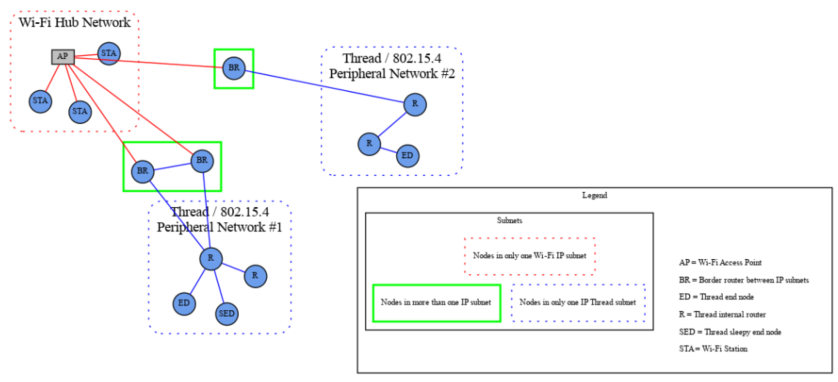
In the single network topology, all Matter devices are connected to a single network. This could by a Thread network, Wi-Fi or Ethernet network. In the case of WiFi/Ethernet, the network could in fact span multiple Wi-Fi and/or Ethernet segments provided that all the segments are bridged at the link layer.





**Star network topology**

The star network topology consists of multiple peripheral networks joined together by a single hub network. The hub network will be the customer’s home network. while the peripheral networks can be of any supported network type. A peripheral network must always be joined directly to the hub network via one or more Border Routers.



In the star network topology any number of peripheral network may be present in a single fabric, including networks of the same type. Nodes may have interfaces onto any network and can communicate directly to other nodes on the same network. If a node needs to cross a network to communicate it must flow through a Border Router.

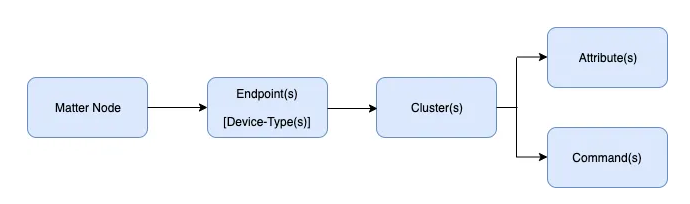
This protocol places a set of requirements on the Border Router. These requirements pertain to

address assignment, route assignment and advertisement, multicast support, and discovery proxying.

### Device Data Model

Need to complete this section.

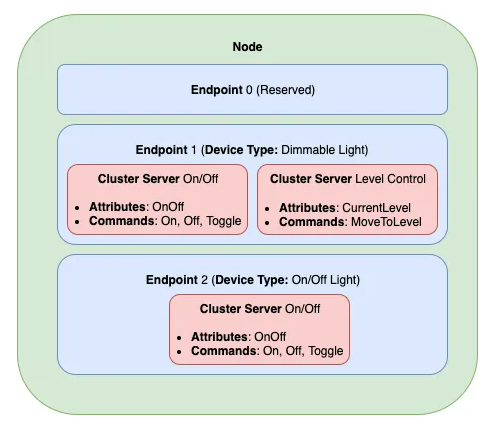
The data model in Matter is a hierarchical modeling of a devices features.



**Node**

This is a uniquely network addressable entity that exposes some functionality. This is typically a physical device that a user can recognize as a whole device. The role of a node is a set of related behaviors. A node can contain one or more roles including:

* Commissioner: Refers to the process of assigning Fabric credentials to a new device.
* Controller: A node that can control one or more nodes such as a On/Off switch
* Controlee: A node that can be controlled by one or more nodes. Such as a actor. Devices that have the controller role cannot be a controlee.
* OTA Provider: Provides OTA software updates.
* OTA Requestor: Requests OTA software updates.



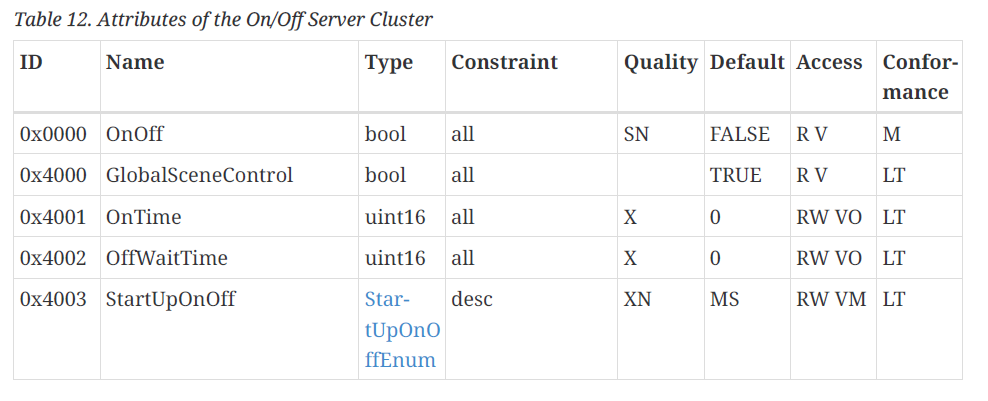
**Endpoint**

Each node has one or more endpoints. A endpoint contain a set functionality’s of a single device. In the example above endpoint 1 is a dimmable light that have the functionality turning on or off AND have a functionality level control, that controls the brightness of the light. Endpoint 2 have only the functionality turning on or off. Note that endpoint 0 is reserved for utility functions.

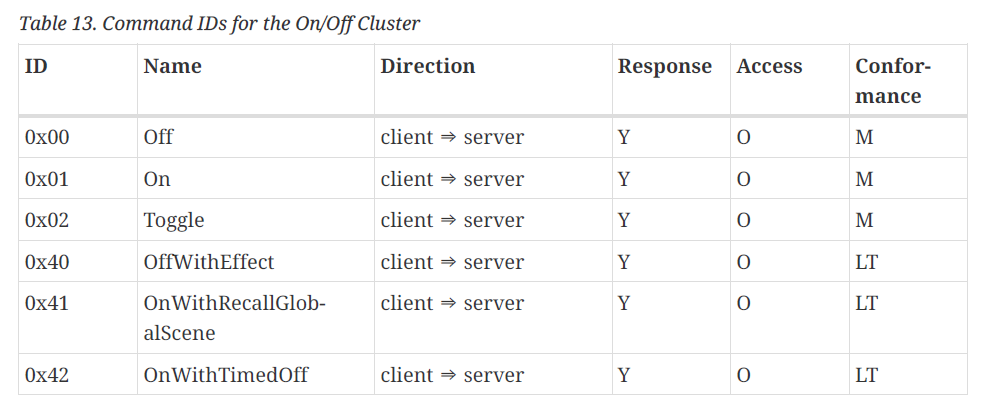
**Clusters**

A cluster groups together commonly used functionality in a reusable building block. Endpoint 1 has two cluster which describes their own functionality. Within the clusters they contain:

* **Attributes**: Attributes are data entities that represent a physical quantity or state. Each attribute is listed in a table with data quality columns: ID, Name, (Data) Type, Constraint, other Quality, Access, Default (value), and Conformance. An attribute also defines its associated semantics and behavior.



* **Commands**: A cluster command provides an ability to invoke a specific behavior on the cluster. A command may have parameters that are associated with it. Each command SHALL be listed in a table with data quality columns: ID, Name, Direction, Response, Access, Conformance. The command table SHALL define the direction of the command as either client to server or server to client. The command table SHALL define the access privileges for each request command or omit the privileges for the default.



The clusters that are supported can be found at the **Matter-(version)-Application-Cluster-Specification** document.

It is possible to make a cluster that is suitable for a specific use case. For example, in your project you need a specific attribute that is not available in the cluster that you gonna use. You can extend that cluster so de attribute is available.

**Events**

Events are a type of attributes that communicate device state changes. They can also be treated as historical data records of something that happened on the device in the past.

**Cluster Client / Server**

A cluster server is stateful and holds attributes, events and commands while a client is stateless and is responsible to initiate interactions with a cluster server.

**Device Type**

A Matter device type is an officially defined collection of requirements for one or more endpoints. Device types are intended to ensure interoperability of different device brands on the market. Each device type definition is composed of the following elements:

* Device type ID
* Device type revision
* One or more mandatory clusters, including each cluster’s minimum revision

All the device types are defined in the 22-27351-002\_matter\_1-1-device-library-specification.pdf document.

## Matter Hub & Brigde

Vertel wat dit is en waarom we dit nodig hebben.

# Silabs Matter SDK

**This documentation is based on the Simplicity Studio 5 IDE!!**

## Locations

* Endpoint configuration file can be found in the project under Workspace/projectfolder/autogen/zap-generated/endpoint\_config.h

## Custom clusters

There is an possibility to add a custom Cluster. The best way to do this is to modify an existing cluster xml file. Clusters xml files can be find in:

* **If using simplicity studio and using the Gecko SDK:** path\_to\SimplicityStudio\SDKs\gecko\_sdk\extension\matter\_extension\src\app\zap-templates\zcl\data-model\chip

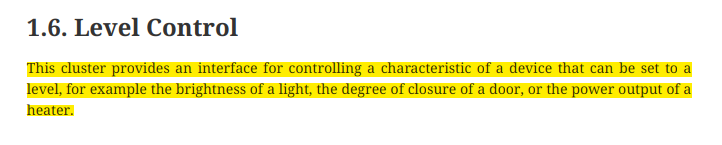
# Issues

## Gyroscope sensor problem

When doing the research of the Matter protocol I was facing a big potential problem for the project. The protocol has pre-build “device type’s” these are officially defined and they are not customizable. The only thing we can adjust is a custom cluster. A cluster contains a set of functionalities. This looks like a major problem because we want to read a sensor that is not included into the officially device type’s and we DON’T want to manipulate a existing device type to get to our use case.

### Possible solution

* When I dived deeper in the Device Library I’ve noticed that there is actually one device type that we can use, the On/Off Sensor. The On/Off sensor contain a cluster Level control. The level control can be anything and is not something specific.



This can solve the problem on the side of the Client (the client can manipulate the server, the server only holds value’s and do something with it).

* Another solution that I found was to create a custom app, combine this with the device type and the Level control cluster. with the instructions from Silabs, It can be found in the Silicon Labs Matter Github repo.

## Connection between matter device (using chip-tool) and the “game”

First I thought that the interaction between the matter protocol and the game can be done by the read commands or to subscribing to an attribute of a endpoint (for example the on/off attribute that holds the on/off state) using the chip-tool. While doing some [tests](#_Simple_on/off_device) it seems that we get a lot of information and not only the state of a device, for example a ‘True’ value as the button is pressed. So getting that values from the chip-tool in the command line isn’t that simple. I don’t even know it is possible to filter everything so I get only the information I want. For controlling the game I need only the value of the attribute, whether it is two on/off “devices” one for left one for right or it is a level control.

### Possible solution

Maybe to make this work I can built another device that can be controlled by the game controller and then use gpio’s (with or without pwm) that can be connected to the raspberry PI and then read these pin’s so we can control the breakout game with this pins?

# Tests

## Simple on/off device

The first test was a simple on/off device to see what the protocol does. Before we can do any tests a matter hub is necessary. I followed the instructions from the silabs demo developer documentation. <https://docs.silabs.com/matter/1.0.1/matter-thread/demo-overview>

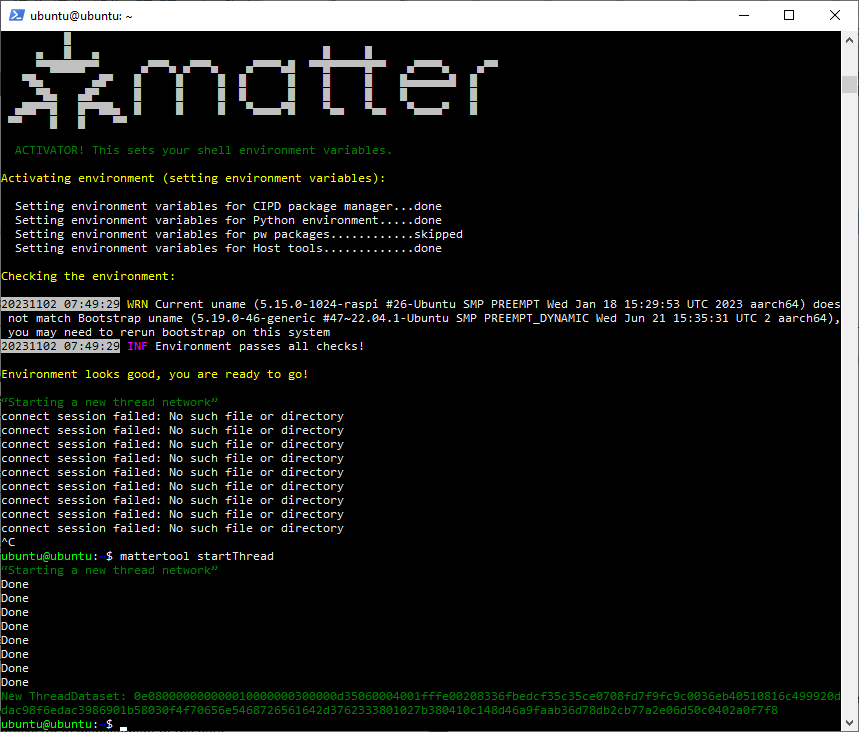
The first thing is to setup a Matter Hub. The Matter Hub consists of the Open Thread Border Router (OTBR) and the chip-tool running on a Raspberry Pi. Silicon Labs has developed a Raspberry Pi image combining the OTBR and chip-tool that can be downloaded and flashed onto an SD Card.

The seconds step was flashing the Radio Co-Processor (RCP) into the board. This was very easy with Simplicity Studio v5. Open SSv5, connect the board its auto detected. Then under “Example projects & demos” select “Thread” in Wireless Technology and than click on create. After the project is created, flash the project into the board.

The third step was flashing the demo “SoC Lighting over Thread” into the board. The demo is located under “Example projects & demos” under the section Matter. The demo can be directly flashed into the board.

Once the image is flashed and inserted into the RPI I searched for the IP-address. Then logged in using SSH. When logged in I had to use a few commands that are listed in <https://docs.silabs.com/matter/1.0.4/matter-thread/chip-tool>

First we need is to start the Thread Network using **mattertool startThread** command. This creates a new Thread Network.



When the network is created, the device can be commissioned by using the command **mattertool bleThread**. When the device is successful created it gives the Node-id.

Geef screenshot

After that we can control the light using the command **mattertool on** and **mattertool off**. With this command all the devices that are commissioned changes there on/off state to on or off.

Screenshot hier

So now I can control the light using this commands but I want to know if I can monitor this. So I searched in the silabs matter docs under the section “matter over Thread” and then under “Using the Chip-Tool” if there was a possibility to monitor the endpoint. What I found was that mattertool is just an alias of chip-tool. So I searched for this and found the documentation on Github: <https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/chip_tool_guide.md>

In this guide there is described how to subscribe to an attribute and that was exactly what I needed. When subscribing to an attribute every change displayed in the command line. To subscribe we have to follow a few steps.

1. Start the Chip-tool in interactive mode using **mattertool interactive start** after this command we can see “>>>” this means that we are in interactive mode.
2. Use the command “**<cluster-name of choice> subscribe**”. The list of all available attributes for the cluster will appears.

Geef hier een screenshot

1. I need the On/Off cluster to monitor the changes. The base command that I used is:

**<cluster-name> subscribe <argument> <min-interval> <max-interval> <node\_id> <endpoint\_id>**

The parameters of this command:

<cluster-name> is the name of the cluster.

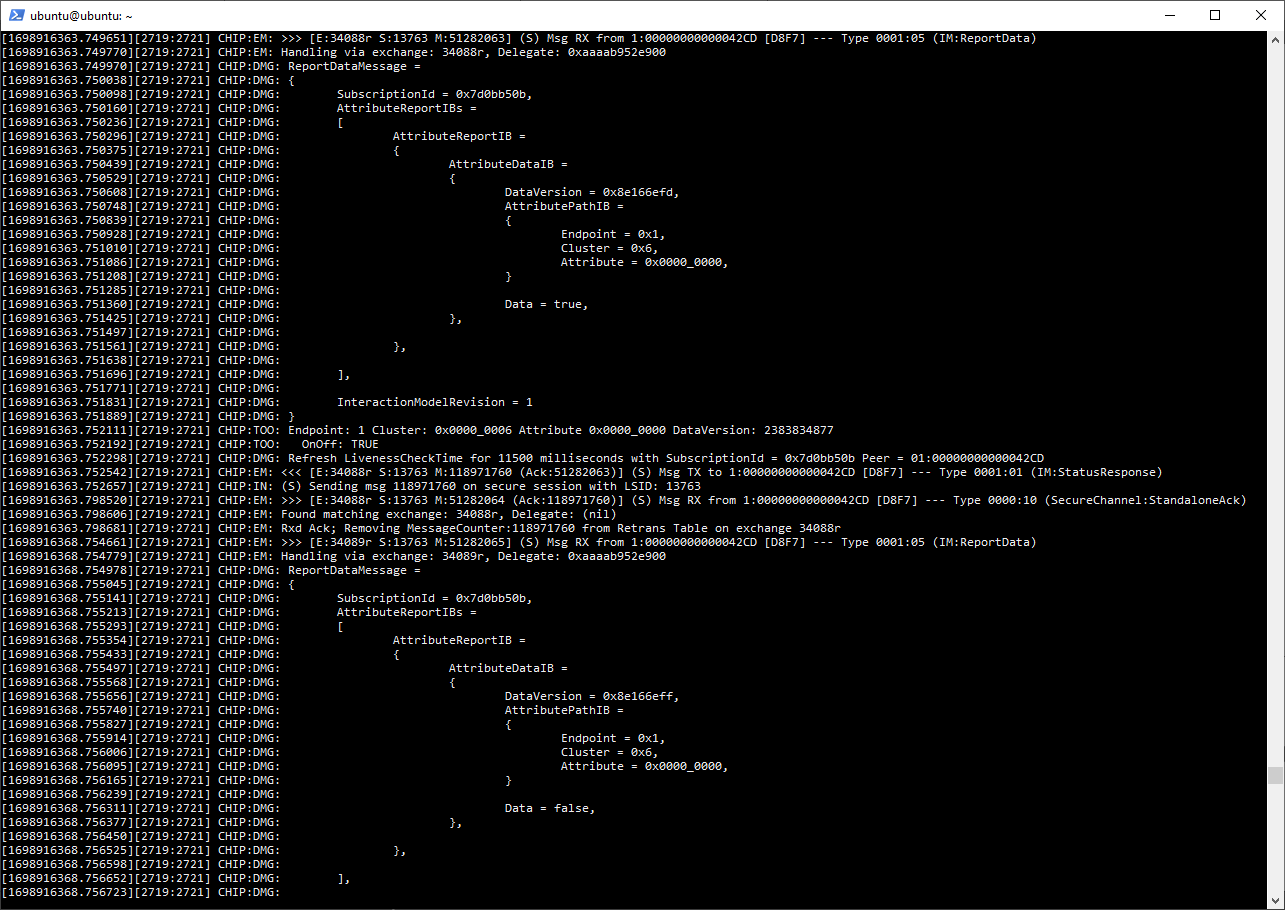
<event-name> is the name of the chosen event.

<min-interval> specifies the minimum number of seconds that must elapse since the last report for the server to send a new report.

<max-interval> specifies the number of seconds that must elapse since the last report for the server to send a new report.

<node\_id> is the user-defined ID of the commissioned node.

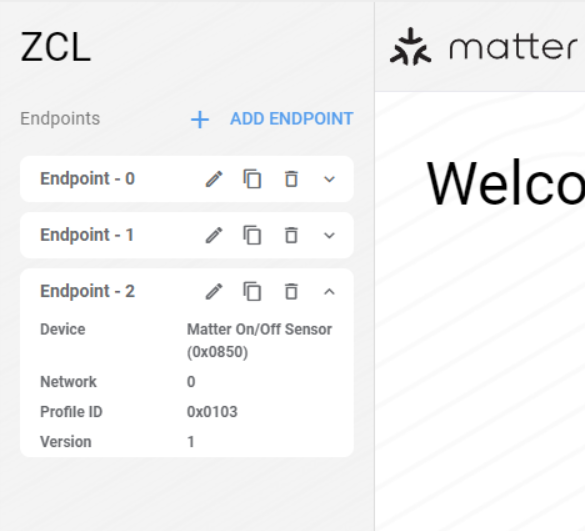
<endpoint\_id> is the ID of the endpoint where the chosen cluster is implemented.



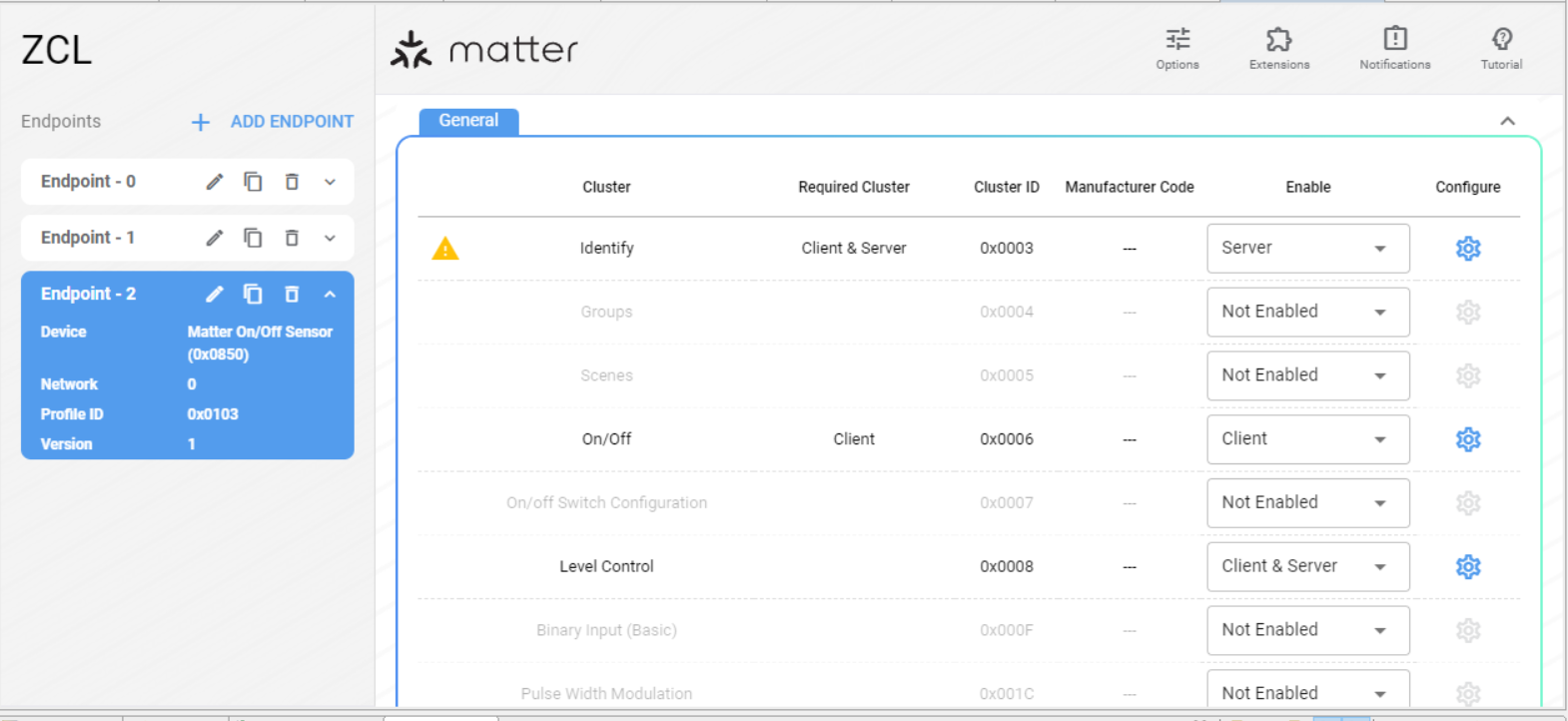
## Level control test

To test the level control I’ll started with the MatterSensorOverThread example and modify the endpoints in the Zigbee Cluster Configuration tool.

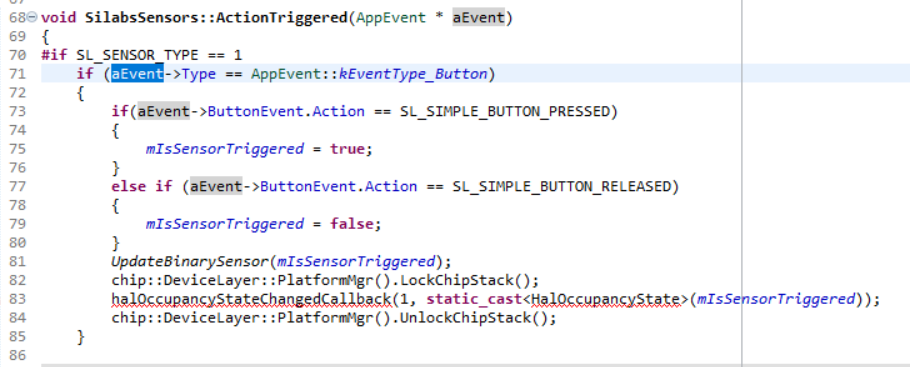
First step was the Zigbee Cluster Configuration tool in Simplicity Studio v5. I added a new endpoint with the device type “Matter On/Off Sensor”. I want to use that cluster to store the Gyroscope data in the future.



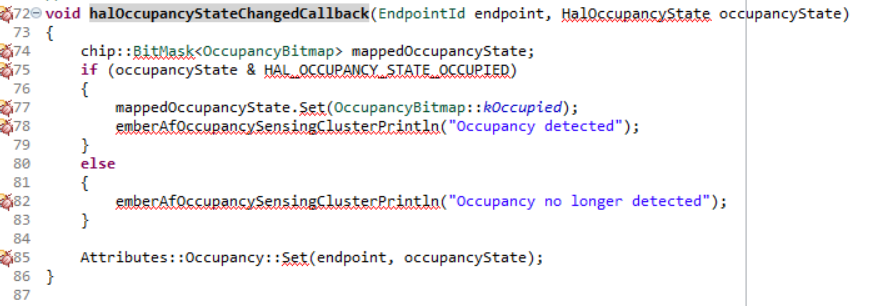
After that I enabled the level control cluster as a Client & Server in the General tab, actually we need only the Server cluster but I select both to test this.



In the example project I found how they used a sensor en stored the data into an Attribute of a Cluster. In the SensorsCallback file we can see what they do when a button is pressed to activate the sensor.



We can see that they call another callbackfunction “halOccupancyStateChangedCallback”. This function is located in the “occupancy-server.h” file in the clusters folder.



On line 85 they set the state by using Attributes::Occupancy::Set(). This is what we are going to need for our sensor. We are going to do the same for the level-control.

chip::app::Clusters::LevelControl::Attributes::CurrentLevel::Set(endpoint, \*val);

To use the set function to change the value of the level control attribute we have to add:

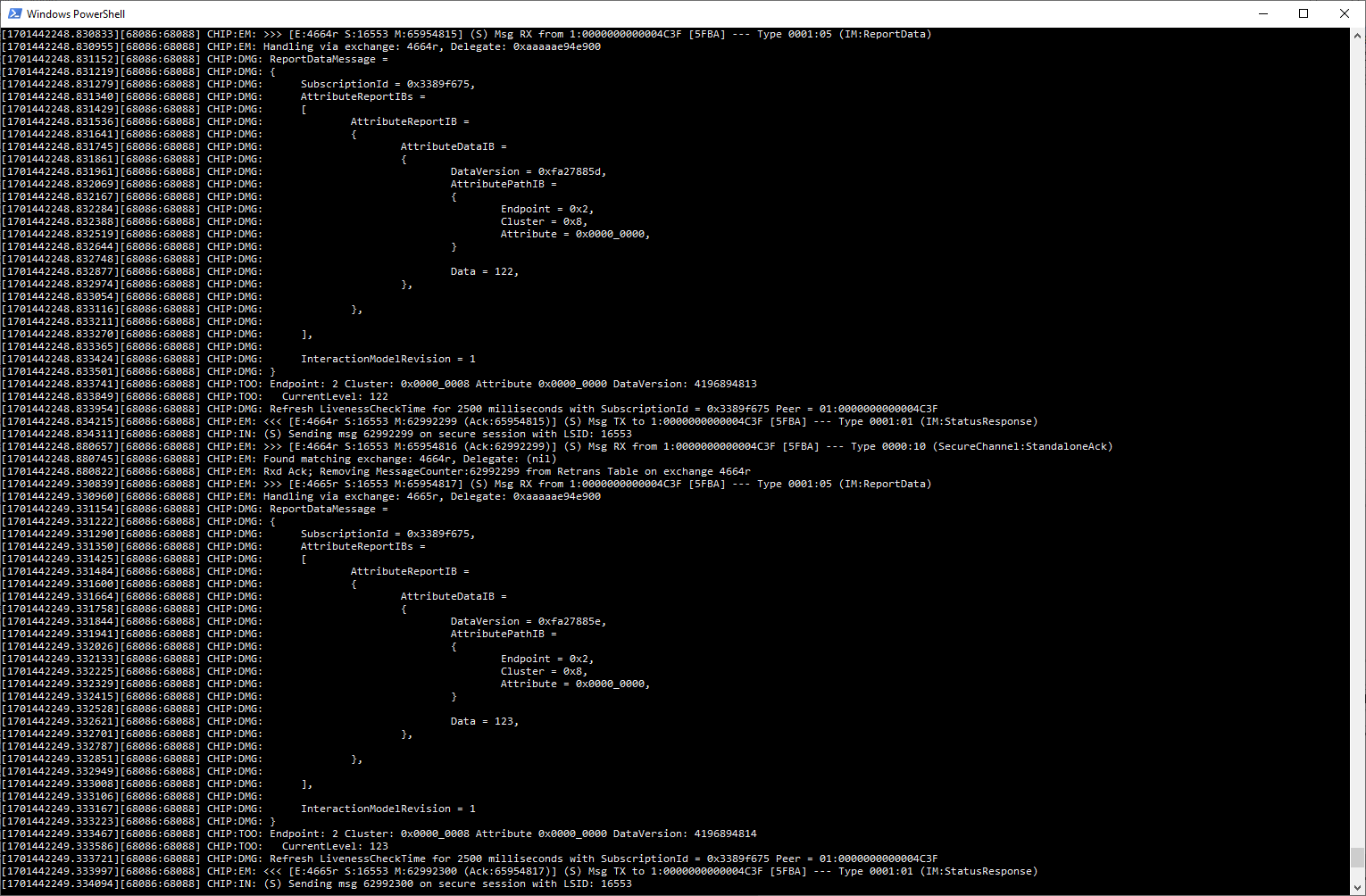
#include *<app-common/zap-generated/attributes/Accessors.h>*

Vertel over de aanpassingen in de AppTask.c

### Capture data using chip-tool

To capture the data we can use the interactive function in the chip-tool and subcribe to the endpoint. Every time the data changes it will be displayed. Use the following commands:

1. Mattertool interfactive start
2. Levelcontrol subscribe current-level (min-interval: I used 0) (max-interval: I used 1) (nodeId) (endpointId)



# Notes

* **How to use the On/Off server with level control to react to the gyroscope**
  + Use a callback function when the gyro reaches a sertain degrees
  + In the callback function include the level-control.h
  + Add app-common/zap-generated/attributes/Accessors.h  
    This is necessary for using the Set function to set the value to the attribute.
  + Add using namespace chip::app::Clusters::LevelControl
  + Add the following line to set the value Attributes::LevelControl::CurrentLevel::Set(endpoint, value) OR when not adding using namespace ….. add chip::app::Clusters::OnOff::Attributes::LevelControl::CurrentLevel::Set(endpoint, value)

# Document Information

## Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Description** |
| 22/09/2023 | BGY | 0.1 | Project overview and wireless controller description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Related Documents

|  |  |
| --- | --- |
| **Ref** | **Description** |
| 22-27349-002\_matter-1-1-core-specification.pdf | Full specification document of the Matter protocol. |
|  |  |
| matter-1-1-application-cluster-specification.pdf | Supported application cluster specification. |
|  |  |
| 22-27351-002\_matter\_1-1-device-library-specification.pdf | All the device types are defined in this document. |
|  |  |