Freescale Semiconductor

How to add an endpoint and a cluster to a ZigBee device in the BeeStack

By: Technical Information Center

Introduction

Does your ZigBee application have only one cluster per device? The answer to this question is usually no so it is very important to know how to add clusters to an application. This document describes in detail how to do this in the BeeStack.

Contents

1. About this document

This document is focused in the BeeStack which is the ZigBee stack produced by Freescale. The solution is created with the BeeKit 3.0.2 with the HCS08 BeeStack codebase 3.1.1 for the MC13234 MCUs but it works with other MCUs and codebases.

A **HA OnOffSwitch** device is taken, it will be modified to add a Thermostat cluster in the same device. Then, it will be tested by creating a network with a **HA OnOffLight** and a **HA TempSensor** devices.

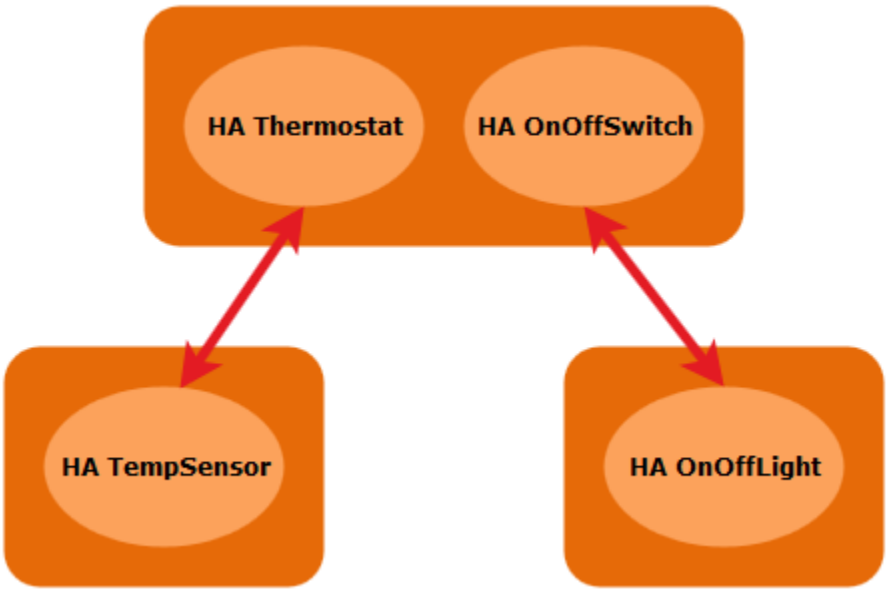

Figure 1. The resulting network.

Let's get down to business…

2. Create a BeeKit solution

The first step is to create the full solution. For this document the MC13234 MCU is used so the HCS08 BeeStack codebase is selected but this procedure applies to other MCUs like Kinetis, if you want to change it, click over the **Select other codebase…** option and mark the desired codebase as **Active** in BeeKit.

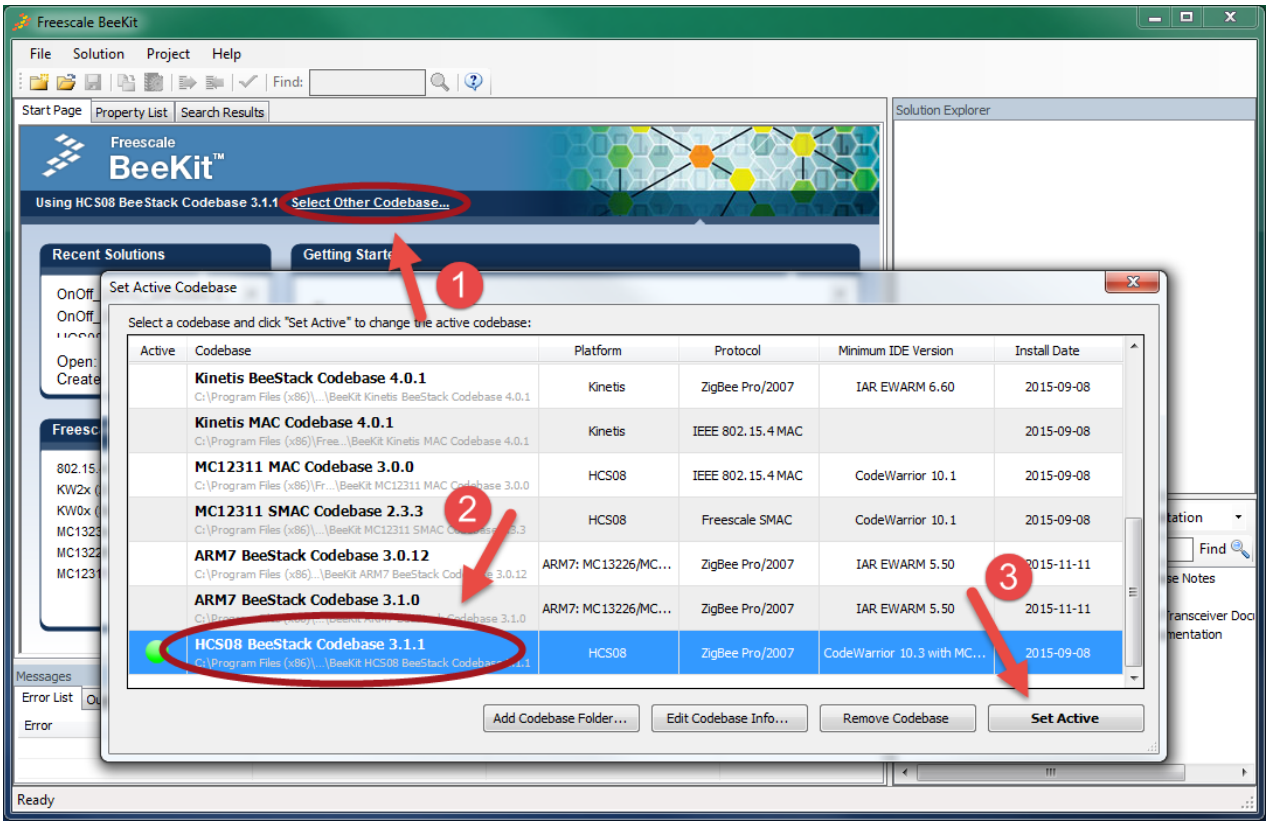

Figure 2. How to change the codebase.

Then create a new project, go to **File > New Project…** to open the wizard, in the **Project types** field select **ZigBee Home Automation Applications** and then in the **Templates** space select **Ha OnOffLight**. Don't forget to give a meaningful name to the solution in the field **Solution name**.

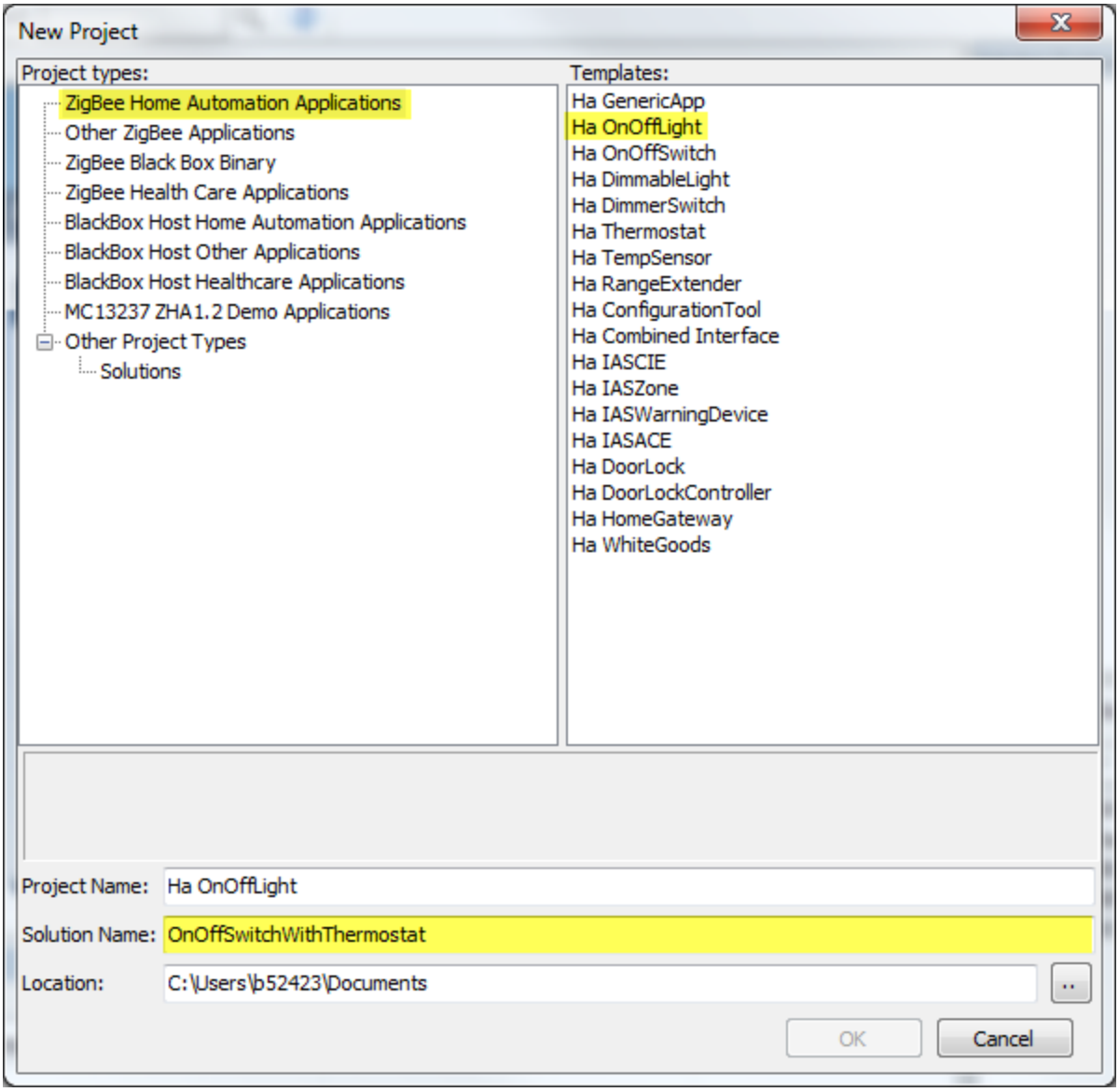

Figure 3. Create the HA solution.

A wizard will be opened, for this document only the MAC address is changed and the other options are the default ones. This is done by changing the default MAC in the wizard.

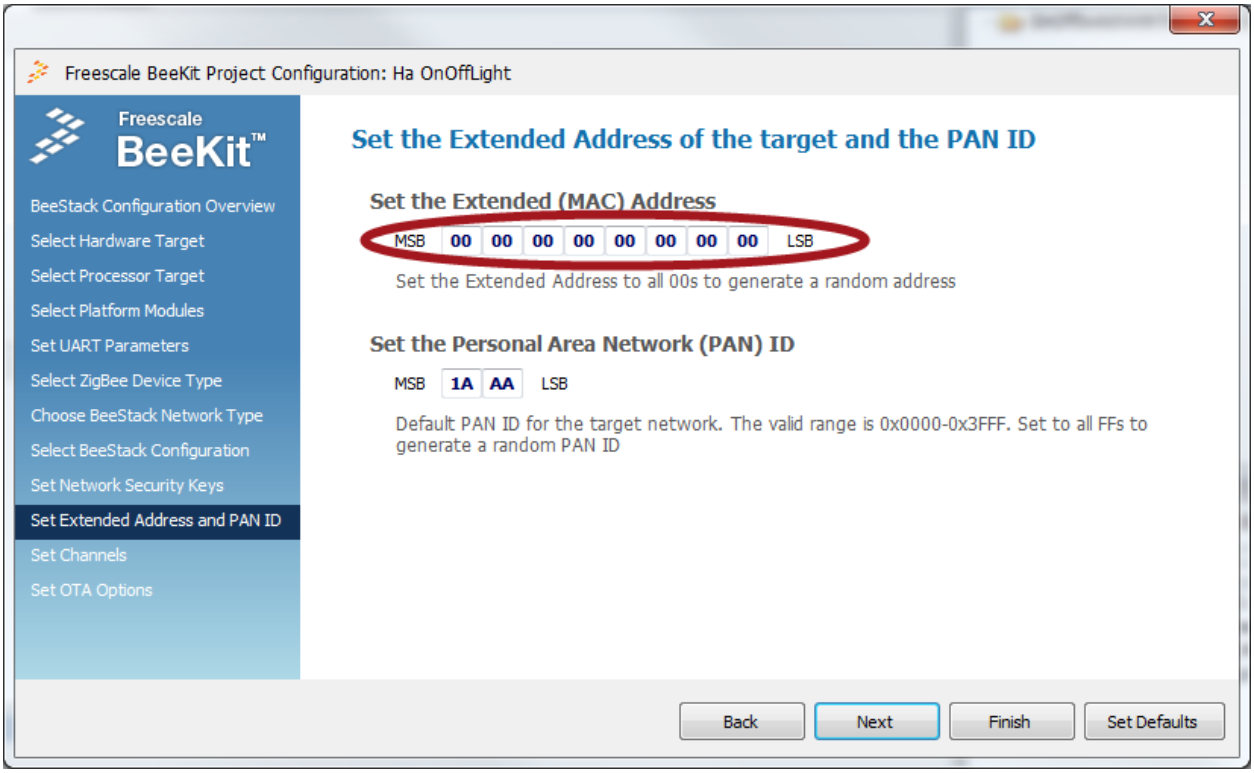

Figure 4. Change the MAC address.

The solution is created with three devices:

- **HA OnOffLight**, which will be the network's coordinator.
- **HA OnOffSwitch**, which will be a router.
- **HA TempSensor**, which will be another router.

It is recommended to use the MC1323x-RCM board in the **HA OnOffSwitch** project to see the Thermostat capabilities in the display, this document uses the MC1323x-REM board.

Open the wizard again to add the other two projects to the solution, by right-clicking over the project's folder in the **Solution explorer** and then click over the **Add…** option.

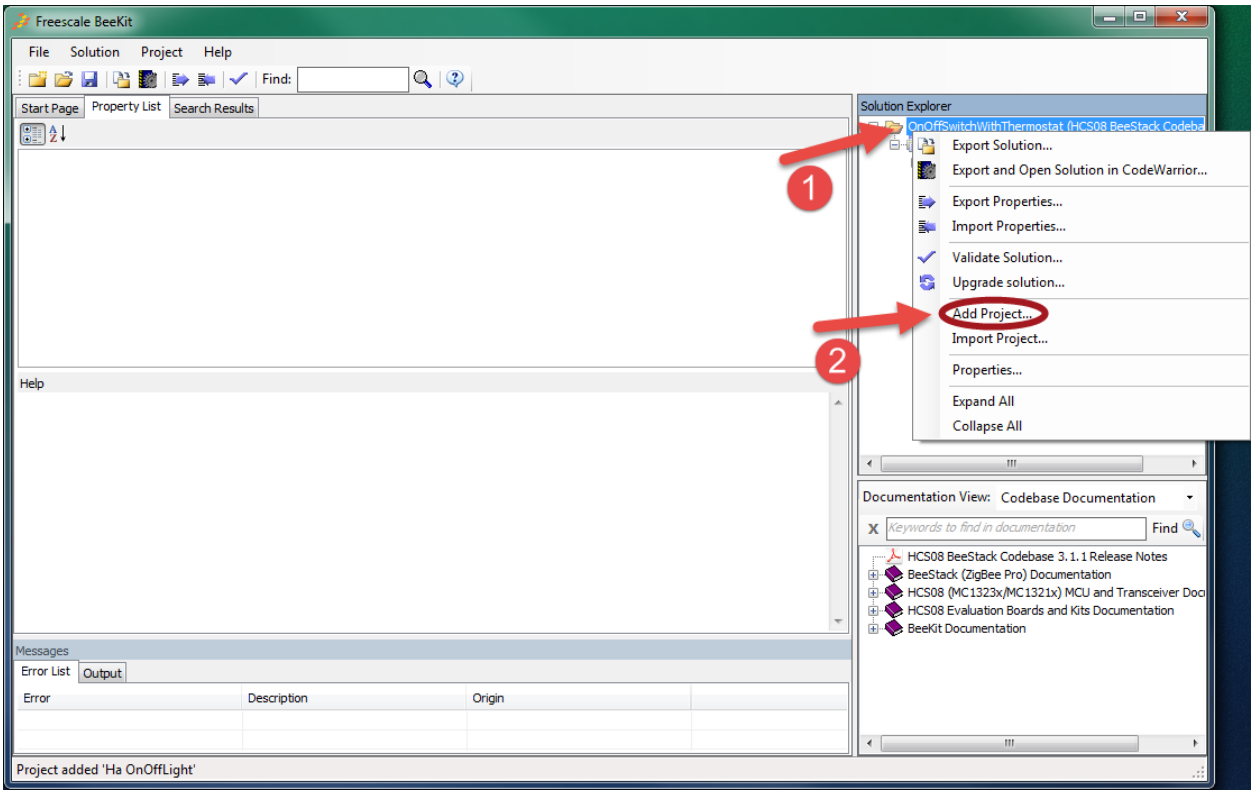

Figure 5. Add a new device to the solution.

3. Add the new endpoint

Once the solution is created, it is time to modify the **HA OnOffSwitch** device to add a new Thermostat endpoint. In the **Solution explorer**, right-click on the **Endpoints** category of the **HA OnOffSwitch** project and then click the **Add Software component…** option.

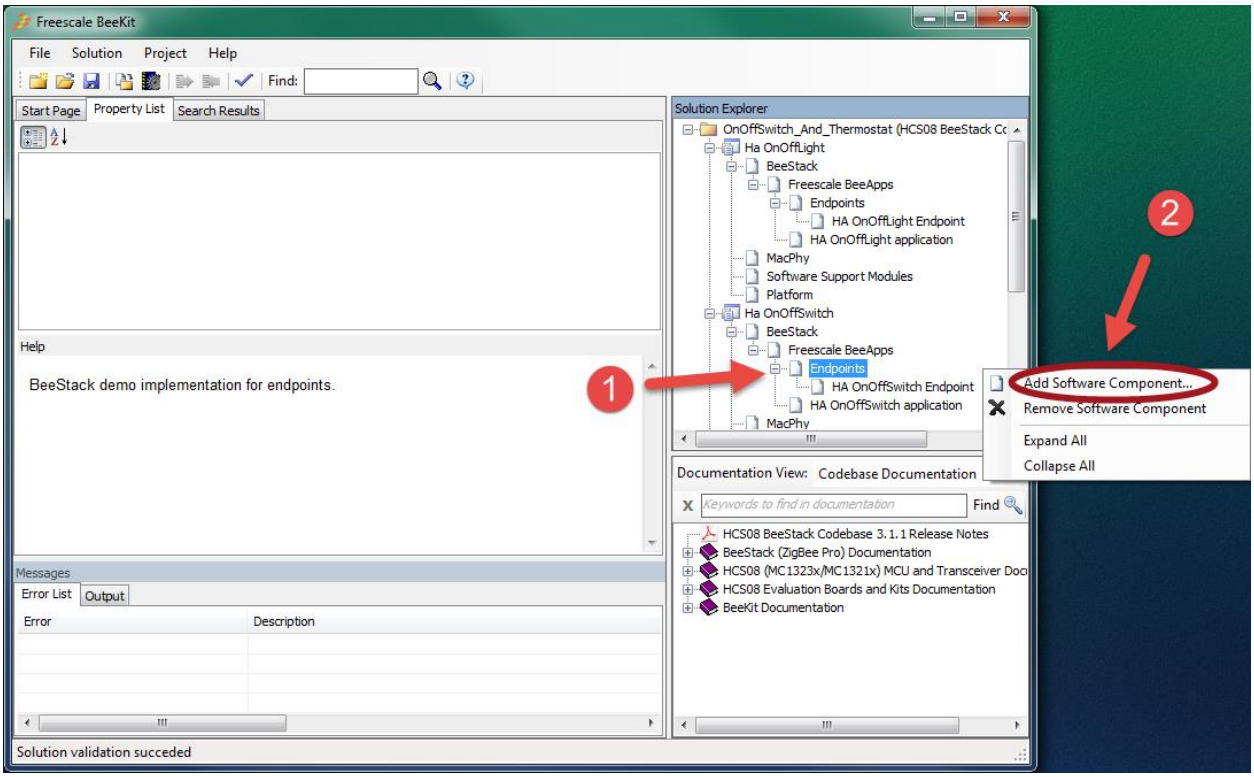

Figure 6. Add a new endpoint.

A new window with a list of endpoints is shown, select the **HA Thermostat Endpoint** option and then click **OK**.

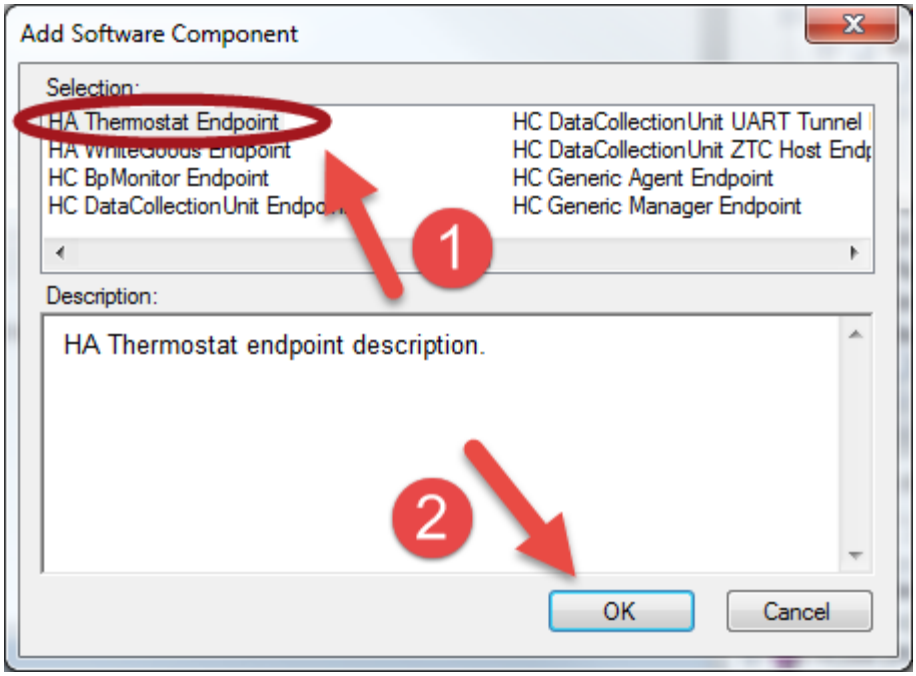

Figure 7. Select the thermostat endpoint.

Since the endpoint numbers must be unique within a ZigBee node, it is needed to change the new endpoint's number. To do this, click over the new **HA Thermostat Endpoint** of the **OnOffSwitch** project, then click on the **simple descriptor** configuration that appears in the **Property List** tab.

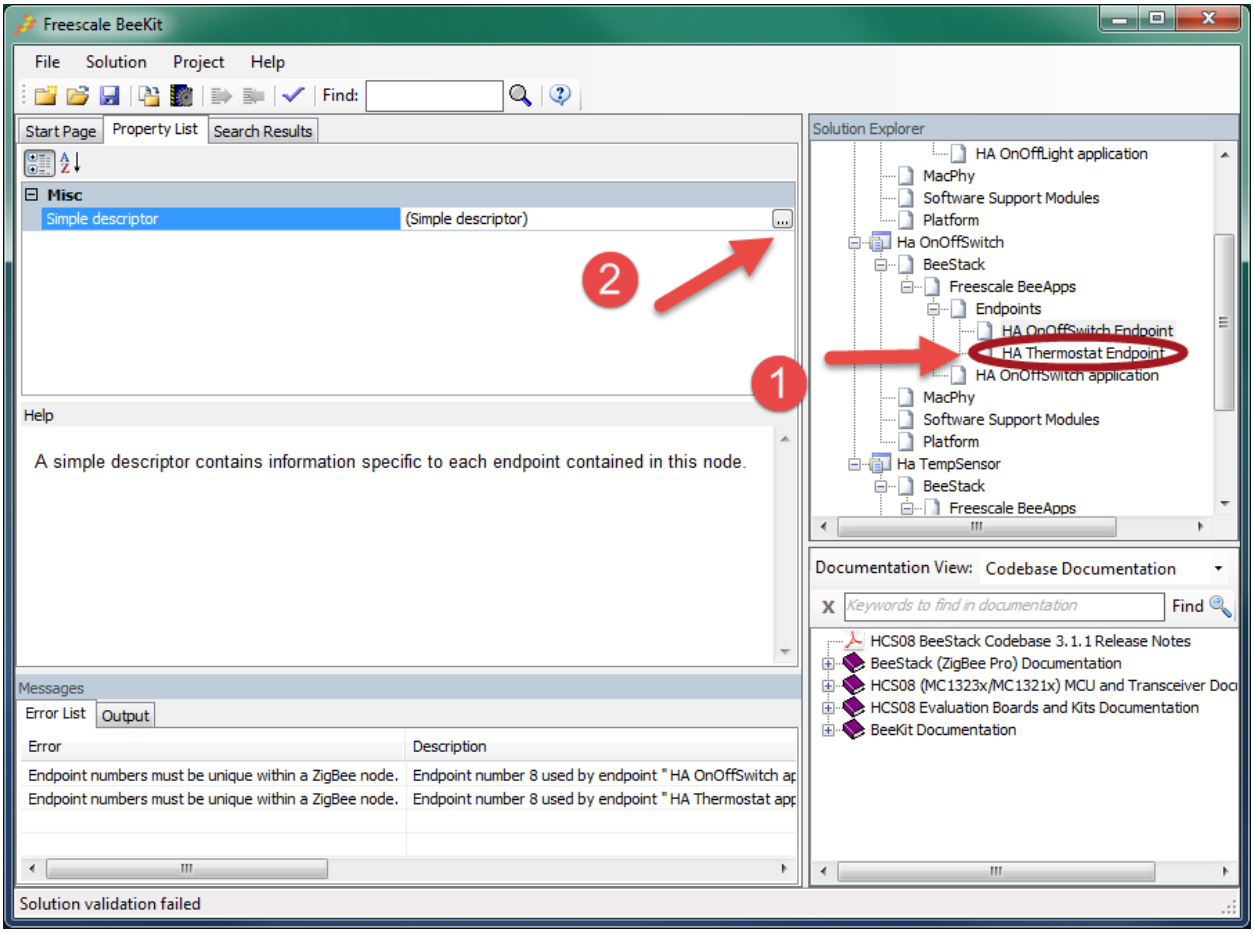

Figure 8. Change the endpoint number.

The **Simple Descriptor** editor window appears, change the **Endpoint number** from 8 to 9 and then click the **OK** button.

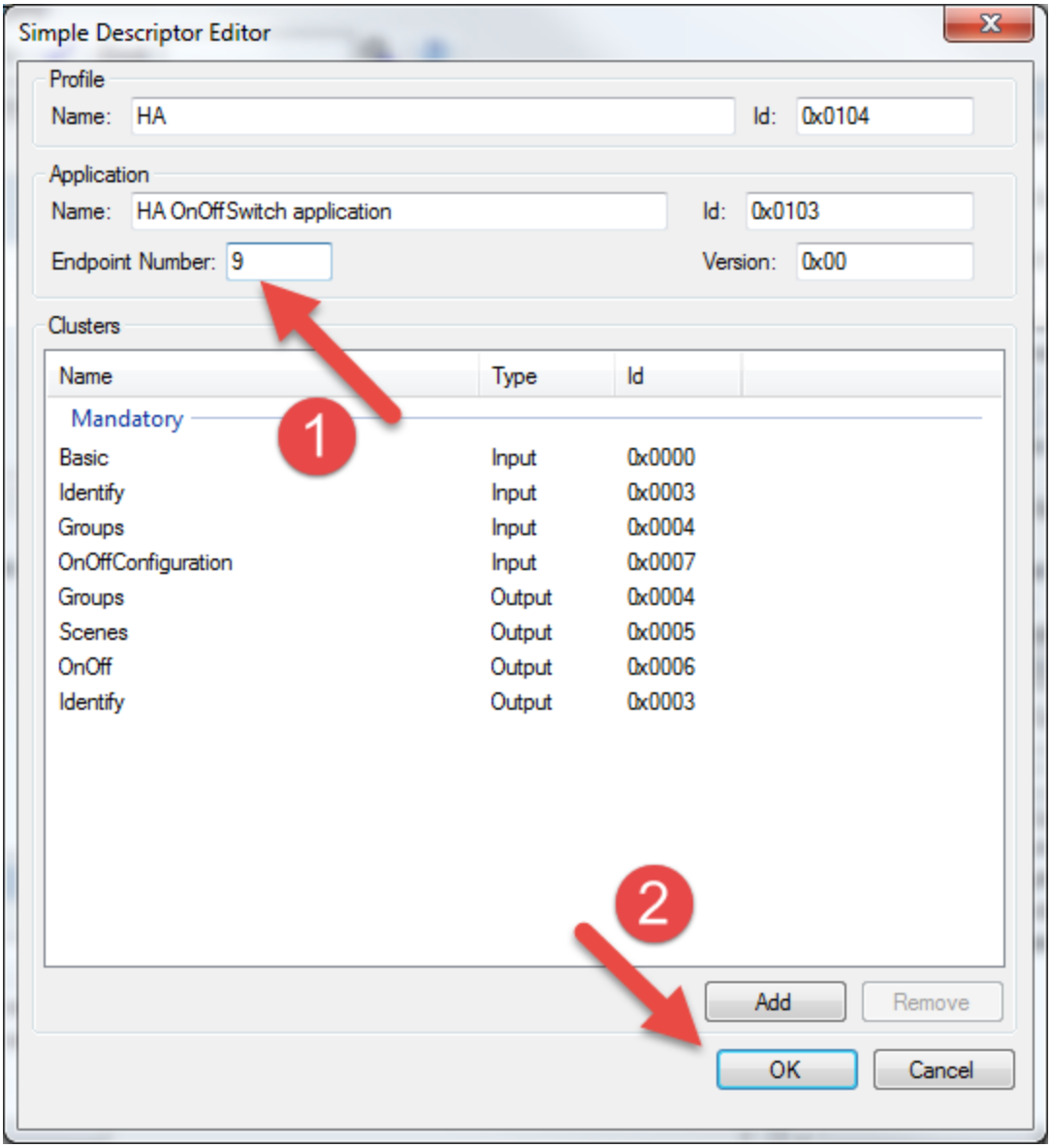

Figure 9. Change the endpoint number.

4. Disable the EZ mode commissioning

It is recommended to disable the EZ mode commissioning, this is done by selecting the **Freescale BeeApps** option in the **Solution explorer** and looking for the options that are shown in the image below. These options must be turned to **False** on **all the devices!** The other options don't need to be changed.

EZ-Mode is a commissioning method that defines network steering and device reset on the node as well as finding and binding for endpoints with target or initiator clusters. For a node that is not already joined to a network, EZ-Mode network steering is the action of creating a network or searching for and joining an open network. For a node that has joined a network, EZ-Mode network steering is the action of opening the network to allow new nodes to join. EZ-Mode finding and binding is the process of automatically establishing application connections, by using the identify cluster, between matching operational clusters on two or more devices.

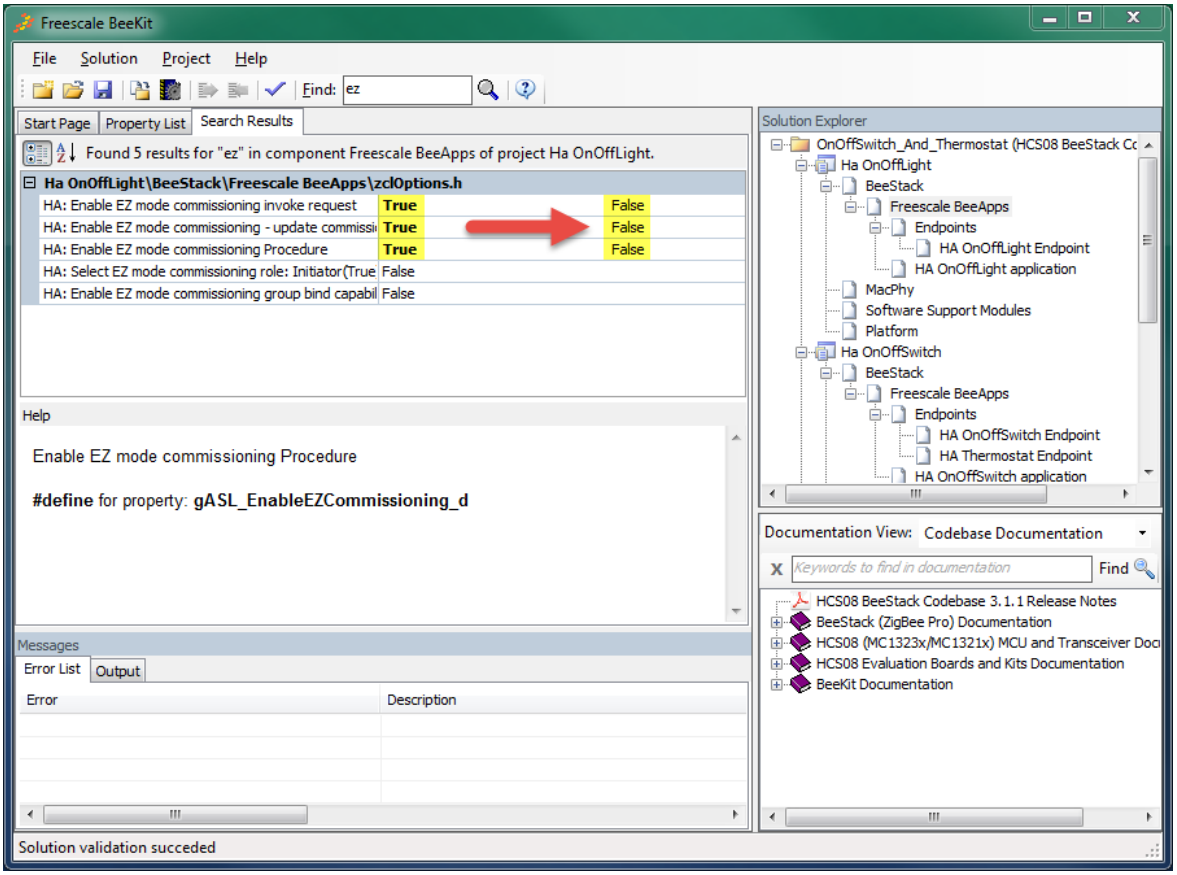

Figure 10. Disable the EZ mode commissioning in BeeKit.

Disabling EZ mode commissioning could also be done in the code by changing these macro definitions in the file **Ha OnOffLight\BeeStack\Freescale BeeApps\zclOptions.h** from **TRUE** to **FALSE**.

```
#ifndef gASL_ZclCmdEzModeInvokeReq_d
#define gASL_ZclCmdEzModeInvokeReq_d                TRUE
#endif

#ifndef gASL_ZclCmdUpdateCommissioningStateReq_d
#define gASL_ZclCmdUpdateCommissioningStateReq_d    TRUE
#endif

/* EZ commissioning */
#ifndef gASL_EnableEZCommissioning_d
#define gASL_EnableEZCommissioning_d                TRUE
#endif
```

```
#ifndef gASL_ZclCmdEzModeInvokeReq_d
#define gASL_ZclCmdEzModeInvokeReq_d                FALSE
#endif

#ifndef gASL_ZclCmdUpdateCommissioningStateReq_d
#define gASL_ZclCmdUpdateCommissioningStateReq_d    FALSE
#endif

#ifndef gASL_EnableEZCommissioning_d
#define gASL_EnableEZCommissioning_d                FALSE
#endif
```

Figure 11. Disable the EZ mode commissioning in the code.

5. Enable the End Device Bind feature

The End Device Bind request is generated from a Local Device wishing to be binded with a remote device. It is usually generated by some user action like a button press. The End Device Bind response is generated by the ZigBee Coordinator in response to a request and contains its status.

It is enabled by turning the **ZDP: Enable End_Device_Bind_req** and **ZDP: Enable End_Device_Bind_rsp** from **False** to **True**. This must be done on **all the devices!**

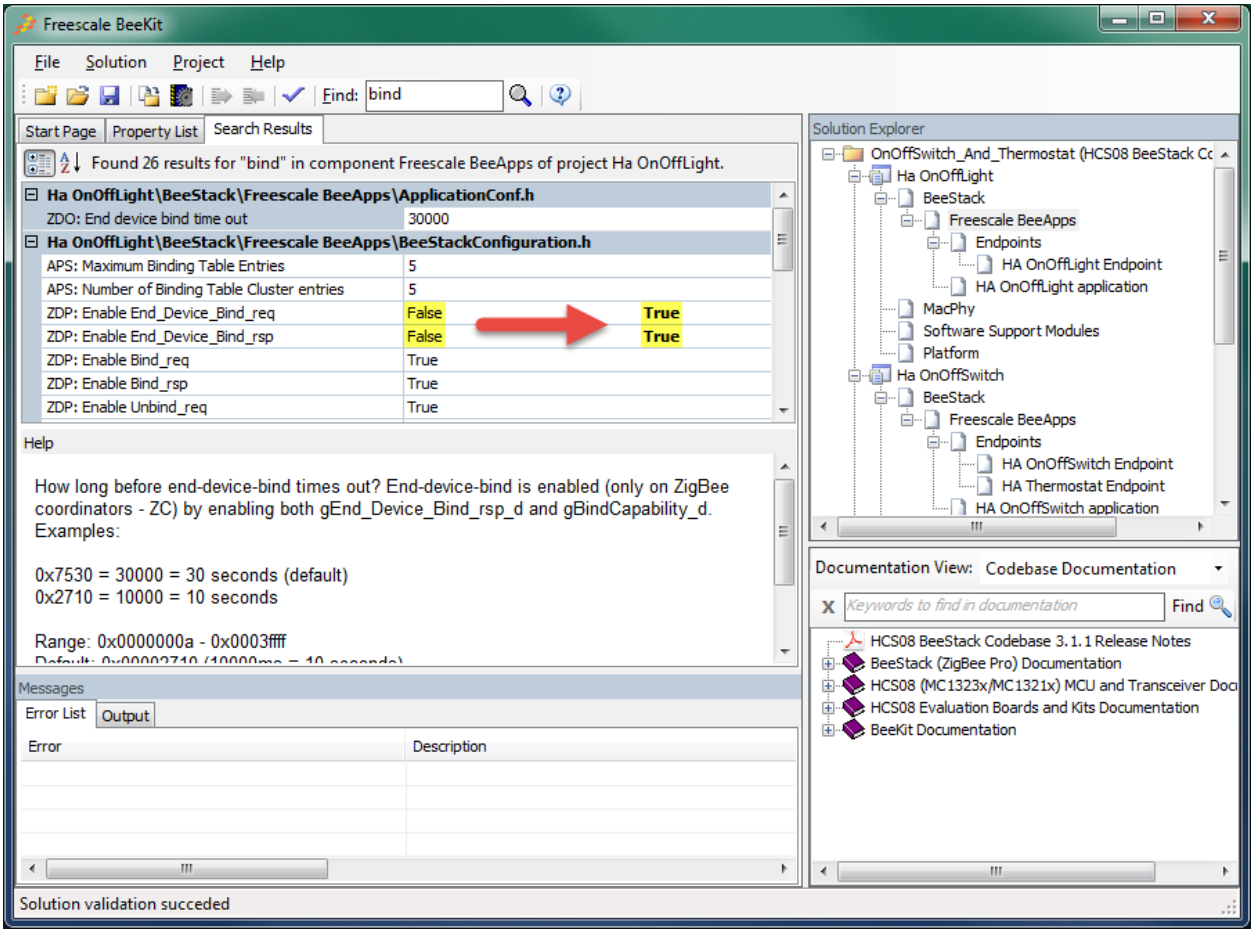

Figure 12. Enable the End Device Bind in BeeKit.

This could also be done in the code by changing these macro definitions in the file **Ha OnOffLight\BeeStack\Freescale BeeApps\BeeStackConfiguration.h** from **FALSE** to **TRUE**.

```
/*
End Device Bind, Bind, Unbind and Bind Management Client Services Primitives
*/
/*End Device Bind Request Enabled*/
#ifndef gEnd_Device_Bind_req_d
#define gEnd_Device_Bind_req_d      FALSE
#endif

#ifndef gEnd_Device_Bind_rsp_d
#define gEnd_Device_Bind_rsp_d      FALSE
#endif
```

```
#ifndef gEnd_Device_Bind_req_d
#define gEnd_Device_Bind_req_d      TRUE
#endif

#ifndef gEnd_Device_Bind_rsp_d
#define gEnd_Device_Bind_rsp_d      TRUE
#endif
```

Figure 13. Enable the End Device Bind in the code.

6. Export the project to IAR or CodeWarrior

Depending on the MCUs and the installed IDEs, these projects can be exported to IAR or CodeWarrior. In this case, these are exported to CodeWarrior.

To do this, it is recommended to **Validate the solution** by pressing the **Check** button as it is shown in the image below and if there is not an error proceed to **Export and open the solution** in the desired IDE.

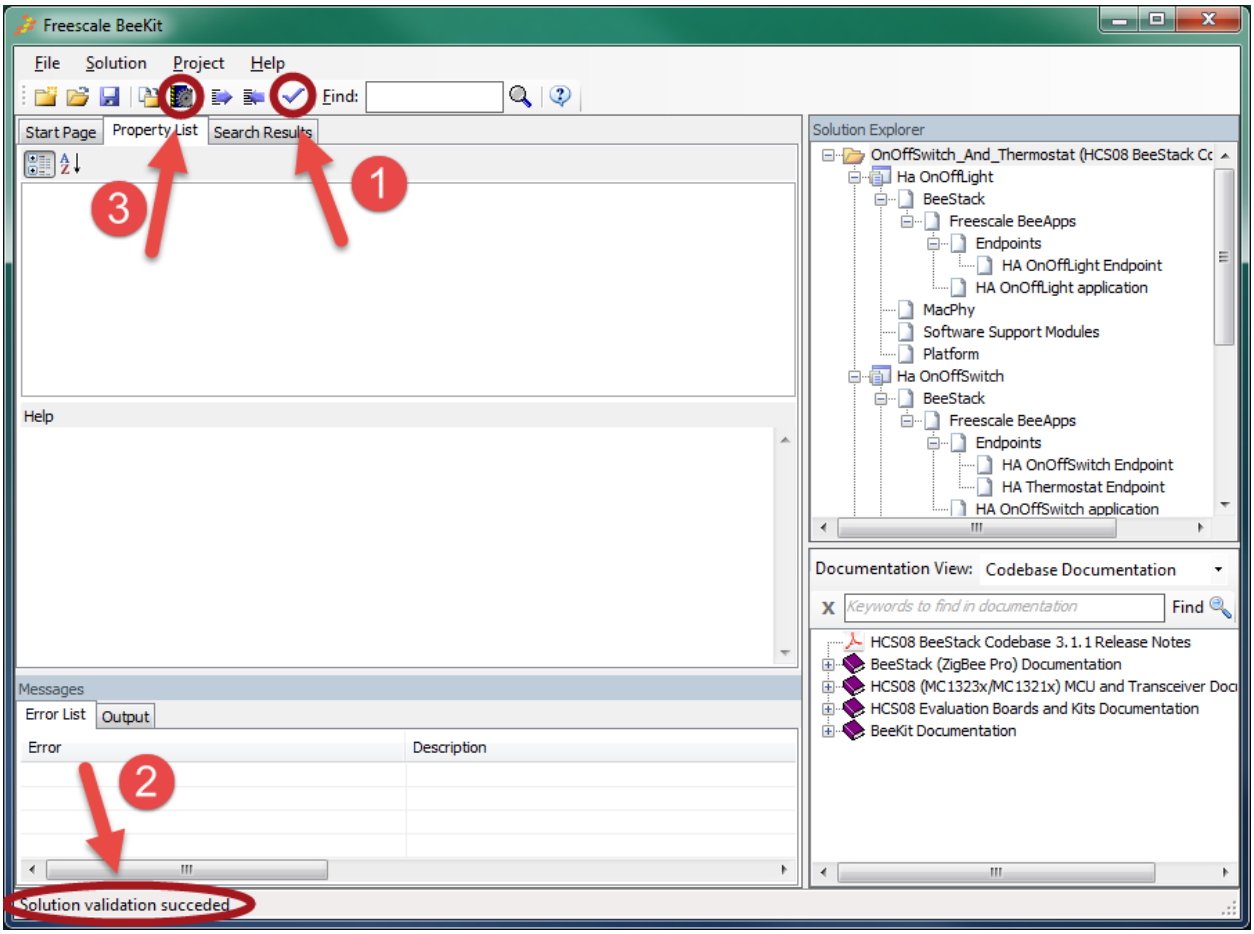

Figure 14. Export the solution.

This will open a new window where the IDE is chosen and then the OK button is pressed to open the solution.

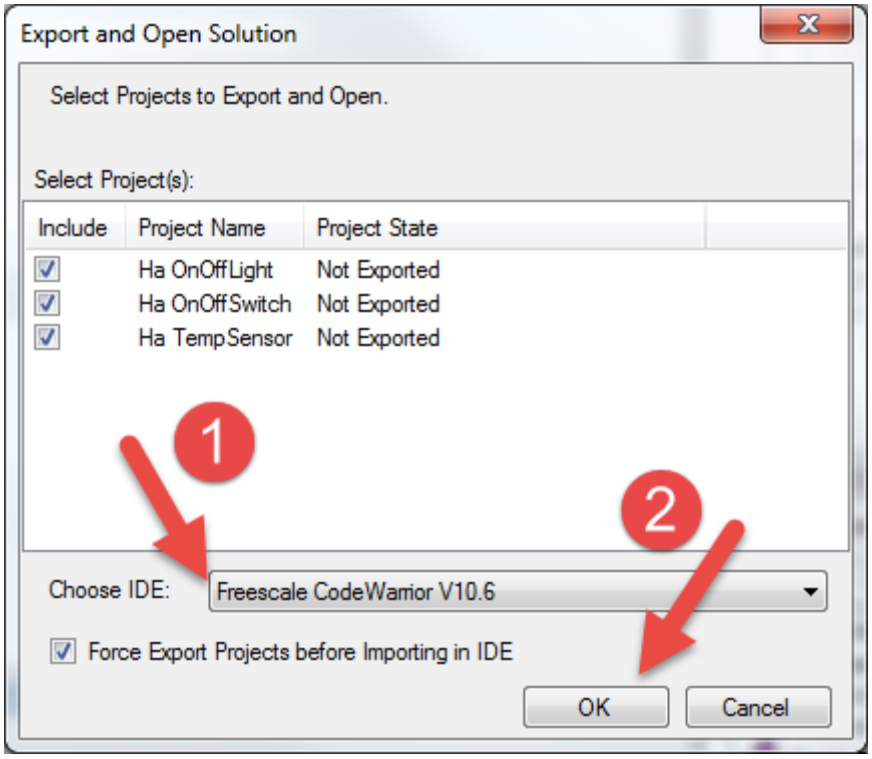

Figure 15. Export the solution.

7. Declare the new endpoint

Now that the solution is exported, it is time to handle the two endpoints in the application. The first step to do this is to declare a new `zbEndPoint_t` variable in the file **Freescale BeeApps\BeeApp.c** to handle the new endpoint.

```
zbEndPoint_t appEndPoint_Thermostat;
```

Depending on your code or application, you may want to declare an **appEndPoint** array with all the endpoints instead of a variable for each one.

8. Add the Temperature Report Callback

Add the **AppThermostatTemperatureReport** callback to the **BeeApps\BeeApp.c** file. This is taken from the **HA Thermostat** demo which we want to merge into the **HA OnOffSwitch**.

Paste the function somewhere in the **BeeApp.c** file, don't forget to add its prototype.

```
void AppThermostatTemperatureReport(zbApsdeDataIndication_t *pIndication,
afDeviceDef_t *pDevice);

/*************************************************************************
  AppThermostatTemperatureReport

  Callback to report the temperature from the sensor to the thermostat.
 *************************************************************************/
void AppThermostatTemperatureReport
(
          zbApsdeDataIndication_t *pIndication,
          afDeviceDef_t *pDevice
)
{
   int16_t Temperature;
   zclCmdReportAttr_t * pReport;
   zclFrame_t *pFrame;
   zbClusterId_t aClusterId;
   pFrame = (void *)pIndication->pAsdu;
   if (pFrame->command == gZclCmdReportAttr_c && IsEqual2BytesInt(pIndication-
>aClusterId, gZclClusterTemperature_c))
   {
          pReport = ZCL_GetPayload(pIndication);
          Temperature = TwoBytesToUint16(pReport->aData);
          Set2Bytes(aClusterId, gZclClusterThermostat_c);

          /* Writes the Temperature from the temperature sensor */
          (void)ZCL_SetAttribute(appEndPoint_Thermostat, aClusterId,
gZclAttrThermostat_LocalTemperatureId_c, gZclServerAttr_c,&Temperature);
          ASL_DisplayTemperature(gASL_LocalTemperature_c, Temperature,
gZclDisplayMode_TempCelsius_c, gASL_HCUOff_c);
   }
   (void)pDevice;
}
```

Please note that the first parameter required by the function **ZCL_SetAttribute** makes reference to the new endpoint so if you are copying this function from the original demo don't forget to change it.

9. Register the thermostat callback

Once the thermostat callback is in the code, it is time to be registered. It is done in the **BeeAppInit** function inside the **BeeApps\BeeApp.c** file through the **ZCL_Register** function.

```
/* Register the thermostat callback */
ZCL_Register(AppThermostatTemperatureReport);
```

10. Get the thermostat endpoint number

The new variable **appEndPoint_Thermostat** should contain the endpoint's number, this is done in the **BeeAppInit** function too. It is given from the **endPointList** array which contains the endpoints' information.

```
/* where to send switch commands from */
appEndPoint = endPointList[0].pEndpointDesc->pSimpleDesc->endPoint;
appEndPoint_Thermostat = endPointList[1].pEndpointDesc->pSimpleDesc->endPoint;
```

The resulting **BeeAppInit** function looks as follows.

```
/*****************************************************************************
 * BeeAppInit
 *
 * Initializes the application
 *
 * Initialization function for the App Task. This is called during
 * initialization and should contain any application specific initialization
 * (ie. hardware initialization/setup, table initialization, power up
 * notification.
 *****************************************************************************/
void BeeAppInit
(
              void
)
{
       index_t i;
       bool_t updateDeviceData = TRUE;

       /* register the application endpoint(s), so we receive callbacks */
       for(i=0; i<gNum_EndPoints_c; ++i) {
              (void)AF_RegisterEndPoint(endPointList[i].pEndpointDesc);
       }

       /* where to send switch commands from */
```

```
        appEndPoint = endPointList[0].pEndpointDesc->pSimpleDesc->endPoint;
        appEndPoint_Thermostat = endPointList[1].pEndpointDesc->pSimpleDesc->endPoint;

        /* initialize common user interface */
        ASL_InitUserInterface("HaOnOffSwitch");

        /* init application timers */
        gAppGenericTimerId = TMR_AllocateTimer();

        /* Register the thermostat callback */
        ZCL_Register(AppThermostatTemperatureReport);

#if gASL_EnableEZCommissioning_d
        /* EZ commissioning Init */
        EZCommissioning_Init();

        /* Check EZ commissioning State */
        if(NvRestoreDataSet(gNvDataSet_App_ID_c))
        {
                if (gZclIdentifyCommissioningState &
gZclCommissioningState_NetworkState_d)
                {
                        ZDO_Start(gStartSilentRejoinWithNvm_c);
                        updateDeviceData = FALSE;
                }
        }
#endif

        if(updateDeviceData)
        {
                BeeApp_FactoryFresh();
        }

}
```

11. Add the new variable into the ASL user interface environment

The ASL User Interface provides a bridge between the human and the ZigBee device. Here, the binding and the application are done, that's why the new endpoint needs to be reachable in this environment. The new endpoint is added by declaring the endpoint in the **BeeApps\ASL\APS_UserInterface.h** file.

```
        extern zbEndPoint_t appEndPoint_Thermostat;
```

12. Add binding support from the thermostat endpoint

The binding is usually done through a button press. By default, the **HA OnOffLight** and the **HA OnOffSwitch** are binded after press the SW3 button on both boards so it is required to use another button to make the bind between the **HA Thermostat** (inside **HA OnOffSwitch**) and the **HA TempSensor**. After inspect the **ASL_HandleKeys** function in the **HA OnOffSwitch** device, it is noticed that the SW2 (**PERMIT_JOIN_SW** case) can be used to this purpose so it will be used.

The binding is done by calling the two functions listed below. Note that the third parameter requested by the **ASL_EndDeviceBindRequest** requires the element 1 from the **endPointList** array which is related to the thermostat endpoint.

```
ASL_UpdateDevice(appEndPoint_Thermostat, gBind_Device_c);

ASL_EndDeviceBindRequest(NULL, aDestAddress, endPointList[1].pEndpointDesc->
pSimpleDesc);
```

The entire **PERMIT_JOIN_SW** case looks as follows.

```
        case PERMIT_JOIN_SW:        /* SW3 on MC1322x-LPN, SW2 on other boards */
              if (appState != mStateIdle_c) {
#if gCoordinatorCapability_d || gRouterCapability_d || gComboDeviceCapability_d
#if gComboDeviceCapability_d
                    if (NlmeGetRequest(gDevType_c) == gEndDevice_c)
                    {
                          break;
                    }
#endif
                    if(PermitJoinStatusFlag)
                          PermitJoinStatusFlag = PermitJoinOff;
                    else
                          PermitJoinStatusFlag = PermitJoinOn;
                    ASL_UpdateDevice(DummyEndPoint,gPermitJoinToggle_c);
                    APP_ZDP_PermitJoinRequest(PermitJoinStatusFlag);
#endif
              }

              ASL_UpdateDevice(appEndPoint_Thermostat, gBind_Device_c);
              ASL_EndDeviceBindRequest(NULL, aDestAddress,
        endPointList[1].pEndpointDesc->pSimpleDesc);

              break;
```

13. Program the devices and test the network

We are almost done, now is time to build and program the boards as usual. These applications, like all the demo applications, start in the configuration mode, where key presses cause the network formation and setup. When that is done, another key press takes the application to run mode, where it can do whatever the application is designed to do.

The suggested steps to run this network are:

a. Press **SW1** in the **HA OnOffLight**, which is the network coordinator, to start the network.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
1	10	14:52:33.184648	26.136416	14	ZigBee	MAC	Beacon Request		0xFFFF	225
2	10	14:52:33.423544	0.238896	14	ZigBee	MAC	Beacon Request		0xFFFF	226
3	10	14:52:33.661144	0.237600	14	ZigBee	MAC	Beacon Request		0xFFFF	227

Figure 16. Once the network is created, the coordinator sends beacon requests.

b. Press **SW1** in the other two boards to join the network.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
5	10	14:52:58.555944	9.678512	14	ZigBee	MAC	Beacon Request		0xFFFF	225
6	28	14:52:58.560872	0.004928	14	ZigBee	NWK	Beacon	0x0000		30
7	10	14:52:58.794584	0.233712	14	ZigBee	MAC	Beacon Request		0xFFFF	226
8	28	14:52:58.798216	0.003632	14	ZigBee	NWK	Beacon	0x0000		31
9	10	14:52:59.032184	0.233968	14	ZigBee	MAC	Beacon Request		0xFFFF	227
10	28	14:52:59.036488	0.004304	14	ZigBee	NWK	Beacon	0x0000		32
11	21	14:52:59.173992	0.137504	14	ZigBee	MAC	Association Request	02:00…	0x0000	228
12	5	14:52:59.175080	0.001088	14	ZigBee	MAC	Acknowledgement			228
13	18	14:52:59.670632	0.495552	14	ZigBee	MAC	Data Request	02:00…	0x0000	229
14	5	14:52:59.671624	0.000992	14	ZigBee	MAC	Acknowledgement			229
15	27	14:52:59.673688	0.002064	14	ZigBee	MAC	Association Response	00:50…	02:00…	229
16	5	14:52:59.674968	0.001280	14	ZigBee	MAC	Acknowledgement			229
🔒 17	65	14:52:59.689784	0.014816	14	ZigBee	APS	Transport Key	0x0000	0x352C	230
18	5	14:52:59.692280	0.002496	14	ZigBee	MAC	Acknowledgement			230
🔒 19	50	14:52:59.743800	0.051520	14	ZigBee	NWK	Link Status	0x352C	0xFFFF	230
🔒 20	57	14:52:59.750712	0.006912	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	231
🔒 21	57	14:52:59.770712	0.020000	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	231
🔒 22	57	14:52:59.898904	0.128192	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	232
🔒 23	57	14:52:59.930216	0.031312	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	232
🔒 24	57	14:53:00.047800	0.117584	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	233
🔒 25	57	14:53:00.080456	0.032656	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	233
🔒 26	50	14:53:03.939336	3.858880	14	ZigBee	NWK	Link Status	0x0000	0xFFFF	234
27	10	14:53:04.930840	0.991504	14	ZigBee	MAC	Beacon Request		0xFFFF	224
28	28	14:53:04.934808	0.003968	14	ZigBee	NWK	Beacon	0x0000		33
29	28	14:53:04.938024	0.003216	14	ZigBee	NWK	Beacon	0x352C		30
30	10	14:53:05.169320	0.231296	14	ZigBee	MAC	Beacon Request		0xFFFF	225
31	28	14:53:05.171336	0.002016	14	ZigBee	NWK	Beacon	0x0000		34
32	28	14:53:05.177448	0.006112	14	ZigBee	NWK	Beacon	0x352C		31
33	10	14:53:05.406904	0.229456	14	ZigBee	MAC	Beacon Request		0xFFFF	226
34	28	14:53:05.409272	0.002368	14	ZigBee	NWK	Beacon	0x0000		35
35	28	14:53:05.410904	0.001632	14	ZigBee	NWK	Beacon	0x352C		32
36	5	14:53:05.551048	0.140144	14	ZigBee	MAC	Acknowledgement			227
37	18	14:53:06.044696	0.493648	14	ZigBee	MAC	Data Request	01:00…	0x0000	228
38	5	14:53:06.045688	0.000992	14	ZigBee	MAC	Acknowledgement			228
39	27	14:53:06.050664	0.004976	14	ZigBee	MAC	Association Response	00:50…	01:00…	235
40	5	14:53:06.051944	0.001280	14	ZigBee	MAC	Acknowledgement			235
🔒 41	65	14:53:06.066408	0.014464	14	ZigBee	APS	Transport Key	0x0000	0x62BF	236
42	5	14:53:06.068904	0.002496	14	ZigBee	MAC	Acknowledgement			236
🔒 43	50	14:53:06.118824	0.049920	14	ZigBee	NWK	Link Status	0x62BF	0xFFFF	229
🔒 44	57	14:53:06.128136	0.009312	14	ZigBee	ZDP	Device Announce	0x62BF	0xFFFF	230
🔒 45	57	14:53:06.147640	0.019504	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	237
🔒 46	57	14:53:06.163736	0.016096	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	234
🔒 47	57	14:53:06.262024	0.098288	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	238
🔒 48	57	14:53:06.300184	0.038160	14	ZigBee	ZDP	Device Announce	0x62BF	0xFFFF	231
🔒 49	57	14:53:06.342344	0.042160	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	235
🔒 50	57	14:53:06.392424	0.050080	14	ZigBee	ZDP	Device Announce	0x0000	0xFFFF	239
🔒 51	57	14:53:06.445720	0.053296	14	ZigBee	ZDP	Device Announce	0x62BF	0xFFFF	232
🔒 52	57	14:53:06.522264	0.076544	14	ZigBee	ZDP	Device Announce	0x352C	0xFFFF	236

Figure 17. Sniffer capture of when the routers join the network.

c. Press **SW3** in the **HA OnOffLight** and in the **HA OnOffSwitch** to bind the OnOff endpoints.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
61	53	14:53:51.258760	2.120000	14	ZigBee	NWK	Link Status	0x62BF	0xFFFF	235
62	77	14:53:58.530808	7.272048	14	ZigBee	ZDP	End Device Bind Request	0x62BF	0x0000	236
63	5	14:53:58.533688	0.002880	14	ZigBee	MAC	Acknowledgement			236
64	53	14:53:59.923048	1.389360	14	ZigBee	NWK	Link Status	0x352C	0xFFFF	240
65	67	14:54:00.415080	0.492032	14	ZigBee	ZDP	Unbind Request	0x0000	0x62BF	243
66	5	14:54:00.417640	0.002560	14	ZigBee	MAC	Acknowledgement			243
67	47	14:54:00.429768	0.012128	14	ZigBee	ZDP	Unbind Response	0x62BF	0x0000	237
68	5	14:54:00.431688	0.001920	14	ZigBee	MAC	Acknowledgement			237
69	67	14:54:00.513256	0.081568	14	ZigBee	ZDP	Bind Request	0x0000	0x62BF	244
70	5	14:54:00.515816	0.002560	14	ZigBee	MAC	Acknowledgement			244
71	47	14:54:00.528488	0.012672	14	ZigBee	ZDP	Bind Response	0x62BF	0x0000	238
72	5	14:54:00.530408	0.001920	14	ZigBee	MAC	Acknowledgement			238
73	67	14:54:00.608696	0.078288	14	ZigBee	ZDP	Unbind Request	0x0000	0x62BF	245
74	5	14:54:00.611256	0.002560	14	ZigBee	MAC	Acknowledgement			245
75	47	14:54:00.622152	0.010896	14	ZigBee	ZDP	Unbind Response	0x62BF	0x0000	239
76	5	14:54:00.624072	0.001920	14	ZigBee	MAC	Acknowledgement			239
77	67	14:54:00.709896	0.085824	14	ZigBee	ZDP	Bind Request	0x0000	0x62BF	246
78	5	14:54:00.712456	0.002560	14	ZigBee	MAC	Acknowledgement			246
79	5	14:54:00.725896	0.013440	14	ZigBee	MAC	Acknowledgement			240
80	67	14:54:00.807512	0.081616	14	ZigBee	ZDP	Unbind Request	0x0000	0x62BF	247
81	5	14:54:00.810072	0.002560	14	ZigBee	MAC	Acknowledgement			247
82	47	14:54:00.821080	0.011008	14	ZigBee	ZDP	Unbind Response	0x62BF	0x0000	241
83	5	14:54:00.823000	0.001920	14	ZigBee	MAC	Acknowledgement			241
84	67	14:54:00.906184	0.083184	14	ZigBee	ZDP	Bind Request	0x0000	0x62BF	248
85	5	14:54:00.908744	0.002560	14	ZigBee	MAC	Acknowledgement			248
86	47	14:54:00.919048	0.010304	14	ZigBee	ZDP	Bind Response	0x62BF	0x0000	242
87	5	14:54:00.920968	0.001920	14	ZigBee	MAC	Acknowledgement			242
88	67	14:54:01.004472	0.083504	14	ZigBee	ZDP	Unbind Request	0x0000	0x62BF	249
89	5	14:54:01.007016	0.002544	14	ZigBee	MAC	Acknowledgement			249
90	47	14:54:01.016424	0.009408	14	ZigBee	ZDP	Unbind Response	0x62BF	0x0000	243
91	5	14:54:01.018328	0.001904	14	ZigBee	MAC	Acknowledgement			243
92	67	14:54:01.100200	0.081872	14	ZigBee	ZDP	Bind Request	0x0000	0x62BF	250
93	5	14:54:01.102744	0.002544	14	ZigBee	MAC	Acknowledgement			250
94	47	14:54:01.114984	0.012240	14	ZigBee	ZDP	Bind Response	0x62BF	0x0000	244
95	5	14:54:01.116904	0.001920	14	ZigBee	MAC	Acknowledgement			244
96	47	14:54:01.199864	0.082960	14	ZigBee	ZDP	End Device Bind Response	0x0000	0x62BF	251
97	5	14:54:01.201784	0.001920	14	ZigBee	MAC	Acknowledgement			251

Figure 18. Sniffer capture for the OnOff binding.

d. Press **SW3** in the **HA TempSensor** and the **SW2** in the **HA OnOffSwitch** (which is a **HA Thermostat**, too) to bind the thermostat and the temperature sensor endpoints.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
103	81	14:54:21.392744	0.006704	14	ZigBee	ZDP	End Device Bind Request	0x62BF	0x0000	247
104	5	14:54:21.395736	0.002992	14	ZigBee	MAC	Acknowledgement			247
105	71	14:54:23.360648	1.964912	14	ZigBee	ZDP	End Device Bind Request	0x352C	0x0000	242
106	5	14:54:23.363336	0.002688	14	ZigBee	MAC	Acknowledgement			242
107	67	14:54:23.471016	0.107680	14	ZigBee	ZDP	Unbind Request	0x0000	0x62BF	254
108	5	14:54:23.473576	0.002560	14	ZigBee	MAC	Acknowledgement			254
109	47	14:54:23.485560	0.011984	14	ZigBee	ZDP	Unbind Response	0x62BF	0x0000	248
110	5	14:54:23.487480	0.001920	14	ZigBee	MAC	Acknowledgement			248
111	67	14:54:23.569656	0.082176	14	ZigBee	ZDP	Bind Request	0x0000	0x62BF	255
112	5	14:54:23.572216	0.002560	14	ZigBee	MAC	Acknowledgement			255
113	47	14:54:23.584520	0.012304	14	ZigBee	ZDP	Bind Response	0x62BF	0x0000	249
114	5	14:54:23.586440	0.001920	14	ZigBee	MAC	Acknowledgement			249
115	47	14:54:23.667064	0.080624	14	ZigBee	ZDP	End Device Bind Response	0x0000	0x62BF	0
116	5	14:54:23.668984	0.001920	14	ZigBee	MAC	Acknowledgement			0
117	67	14:54:23.680840	0.011856	14	ZigBee	ZDP	Bind Request	0x62BF	0x352C	250
118	5	14:54:23.683400	0.002560	14	ZigBee	MAC	Acknowledgement			250
119	47	14:54:23.696040	0.012640	14	ZigBee	ZDP	Bind Response	0x352C	0x62BF	243
120	5	14:54:23.697960	0.001920	14	ZigBee	MAC	Acknowledgement			243
121	67	14:54:23.764952	0.066992	14	ZigBee	ZDP	Unbind Request	0x0000	0x352C	1
122	5	14:54:23.767512	0.002560	14	ZigBee	MAC	Acknowledgement			1
123	47	14:54:23.779112	0.011600	14	ZigBee	ZDP	Unbind Response	0x352C	0x0000	244
124	5	14:54:23.781032	0.001920	14	ZigBee	MAC	Acknowledgement			244
125	67	14:54:23.864536	0.083504	14	ZigBee	ZDP	Bind Request	0x0000	0x352C	2
126	5	14:54:23.867096	0.002560	14	ZigBee	MAC	Acknowledgement			2
127	47	14:54:23.879080	0.011984	14	ZigBee	ZDP	Bind Response	0x352C	0x0000	245
128	5	14:54:23.881000	0.001920	14	ZigBee	MAC	Acknowledgement			245
129	47	14:54:23.959976	0.078976	14	ZigBee	ZDP	End Device Bind Response	0x0000	0x352C	3
130	5	14:54:23.961896	0.001920	14	ZigBee	MAC	Acknowledgement			3

Figure 19. Sniffer capture for the temperature binding.

e. Keep **SW1** pressed for one second on all the boards to turn into the run mode.

f. Press **SW1** in the **HA OnOffSwitch** to toggle the remote light in the **HA OnOffLight**.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
134	53	14:54:45.107992	8.656640	14	ZigBee	NWK	Link Status	0x352C	0xFFFF	247
135	48	14:54:45.272216	0.164224	14	ZigBee	ZCL	On/Off: Toggle	0x62BF	0x0000	252
136	5	14:54:45.274168	0.001952	14	ZigBee	MAC	Acknowledgement			252
137	48	14:54:48.520056	3.245888	14	ZigBee	ZCL	On/Off: Toggle	0x62BF	0x0000	253
138	5	14:54:48.522008	0.001952	14	ZigBee	MAC	Acknowledgement			253
139	53	14:54:49.394552	0.872544	14	ZigBee	NWK	Link Status	0x0000	0xFFFF	5
140	48	14:54:49.521112	0.126560	14	ZigBee	ZCL	On/Off: Toggle	0x62BF	0x0000	254
141	5	14:54:49.523064	0.001952	14	ZigBee	MAC	Acknowledgement			254
142	48	14:54:50.647160	1.124096	14	ZigBee	ZCL	On/Off: Toggle	0x62BF	0x0000	255
143	5	14:54:50.649112	0.001952	14	ZigBee	MAC	Acknowledgement			255

Figure 20. Sniffer capture for On/Off toggles.

g. Keep **SW2** pressed for one second in the **HA TempSensor** to start to send periodical temperature reports to the **HA Thermostat**.

	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	MAC Src.	MAC Dst.	MAC Seq.
146	53	14:55:04.465544	4.292960	14	ZigBee	NWK	Link Status	0x0000	0xFFFF	6
147	53	14:55:06.343992	1.878448	14	ZigBee	ZCL	Temperature Measurement: Report Attributes	0x352C	0x62BF	249
148	5	14:55:06.346088	0.002096	14	ZigBee	MAC	Acknowledgement			249
149	53	14:55:06.578024	0.231936	14	ZigBee	NWK	Link Status	0x62BF	0xFFFF	1
150	53	14:55:13.839112	7.261088	14	ZigBee	ZCL	Temperature Measurement: Report Attributes	0x352C	0x62BF	250
151	5	14:55:13.841224	0.002112	14	ZigBee	MAC	Acknowledgement			250
152	53	14:55:15.233384	1.392160	14	ZigBee	NWK	Link Status	0x352C	0xFFFF	251
153	53	14:55:15.841704	0.608320	14	ZigBee	ZCL	Temperature Measurement: Report Attributes	0x352C	0x62BF	252
154	5	14:55:15.843816	0.002112	14	ZigBee	MAC	Acknowledgement			252
155	53	14:55:17.844040	2.000224	14	ZigBee	ZCL	Temperature Measurement: Report Attributes	0x352C	0x62BF	253
156	5	14:55:17.846152	0.002112	14	ZigBee	MAC	Acknowledgement			253

Figure 21. Sniffer capture for periodical temperature measurement report attributes.

h. Press **SW1** to decrease and **SW2** to increment the simulated temperature in the **HA TempSensor**. It is seen in the **HA OnOffSwitch** that the LEDs change their state depending on the **HA TempSensor** temperature.

- LED2 flashing - Temperature is below 5°C
- All LEDs off - Temperature is between -5°C and 10°C
- LED2 On, LED3 and LED4 off - Temperature is between 10°C and 20°C
- LED2 and LED3 on, LED4 off - Temperature is between 20°C and 30°C

- LED2, LED3 and LED4 on - Temperature is between 30°C and 40°C
- LED2, LED3 and LED4 flashing - Temperature is above 40°C

14. Conclusion

This guide showed how easy is to add a new cluster to add new functionalities to a ZigBee device in the BeeStack. This procedure is a good starting point for the Applications Engineer to design more complex devices.