This is an organised version of the previous project (AWS complete service). It makes use of the classes structures in node js.

There are three levels of scripts. The highest and the executed one is the task, i.e. TempSensortask and LEDtask. The middle level is the device, i.e. TempSensordevice and LEDdevice. Each script calls the lower one and inherits some functionalities from it. The Thing script is the lowest and is the one that all scripts end up using as every device is a Thing at the end.

More closely, the Thing is the general class that all the devices use, which contain the general functions (connecting to AWS, registering, delta-ing.. etc.). For instance, The TempSensordevice inherits all the functionalities of the Thing, and adds other functions that is only to be used in the Temperature Sensor case such as processing the Enocean telegram to extract the temperature. The TempSensortask does the tasks of the LED as a device (turns on and off).

One thing to note about this exact version of the service is that we extended the LED task from turning the LED on/off by the pi to sending data via Bluetooth to an HC-05 chip and hence to an Arduino to turn the LED on/off. You can go one step back by eliminating the Bluetooth part and decapsulate the onSwitchOn/ onSwitchOn functions out of the Bluetooth.connect in the following code:

bluetooth.connect('00:18:E5:03:67:DA', '1', function(err, connection){

MyLED.onSwitchOn(function(){

LedPin.writeSync(1);

connection.write(new Buffer('1', 'utf-8'), () => {

console.log("wrote");

});

console.log("LED is on");

})

MyLED.onSwitchOff(function(){

LedPin.writeSync(0);

connection.write(new Buffer('0', 'utf-8'), () => {

console.log("wrote");

});

console.log("LED is off");

})

});

Now, for using the Arduino to swith on/off lights, AC, pumps etc. via a relay you need to follow the tutorial in the link below. PAY ATTENTTION to the small details in isolating the circuits such as the connections of the Arduino with the relay (only 5V and input, NO GROUND). <https://www.youtube.com/watch?v=LLFQ8sBWc80>

The following websites might be interesting to look at for in regard with relays:

<https://relaypros.com/Relay/Relay/Wireless>

<https://ncd.io/wireless-relays/>

<https://www.controlanything.com/Relay/Relay/ULTRA_LOW_COST_WIRELESS>

In this folder and the other (XBEE), there are three files for LED tasks. One raw file is for a relay connected straight to the pi (LEDtask). The LEDtaskBT is using blueotooth to send on/off commands to a remote chip that is connected to a microcontroller which controls the relay. The LEDtaskXBEE in the other folder is the same as the previous one except that this is using XBEE.

**Some Code Explanation:**

1. The get() function is used to fetch the state of the shadow at the start, and it must be used after registering the Thing.
2. The .on status function is triggered by either .get() or .update()
3. The .on delta function is triggered if only a change takes place. Notices that even if there’s a delta (in the platform) before running the code, .on delta function won’t be triggered.
4. The two functions (delta and status) return stateObject which is an object representing the shadow. This object contains the state property. In the case of .on status, the state property is the all the thing inside the shadow box in the platform (i.e desired, reported, delta etc.). In the case of the .on delta, the returned state property has only the delta part.
5. The update() function writes the argument as a property to the reported section in the shadow box in the platform.

Notice that LEDtaskSonoff works with LEDdeviceSonoff and they work both ways; commands (from platform to device) and updates (from device to platform.

Changed the update method to add null as a value to desired in the thing.js file. This is because of the weird behaviour that happens following the time when the desired gets the same value as the reported (for instance, when lights on and user switches on via the app) in this case both become on/active1. If just after this happens the user uses the mech switch to switch is off, it stays on (sends reported as off but on cloud desired still on) so the end result is still on, somehow after a couple if (closely timed) clicks with the mech switch, the lights turn off and only reported stays there with value off.

I solved this by sending null as a value for desired when updating the data