



SmartThings
Developers

SmartThings Device Connectivity

Integrating Devices to the Cloud and getting setup for Apps

Vinay Rao, Samsung SmartThings
Tom Manley, Samsung SmartThings

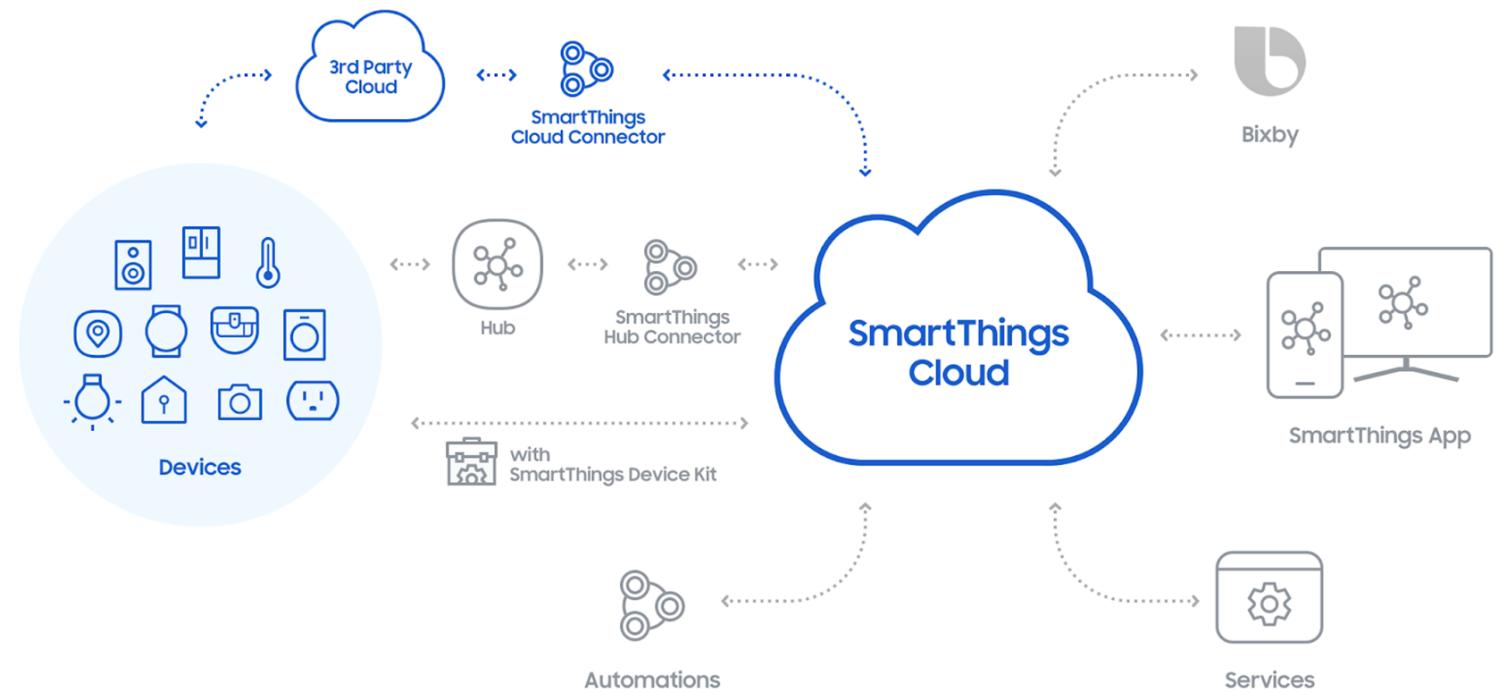
IoT Fuse
April 23, 2019

Setup

- Install the SmartThings app for Android or iOS (not the Classic App)
- Sign in and create a Samsung account if you don't already have one
- We'll be going around the room sharing a SmartThings location with you
- Get the slides from <https://github.com/tpmanley/conferences/tree/master/iotfuse-2019>

SmartThings Ecosystem

- Open, device compatible
- Intelligent, SmartApps
- Secure
- Developer Support

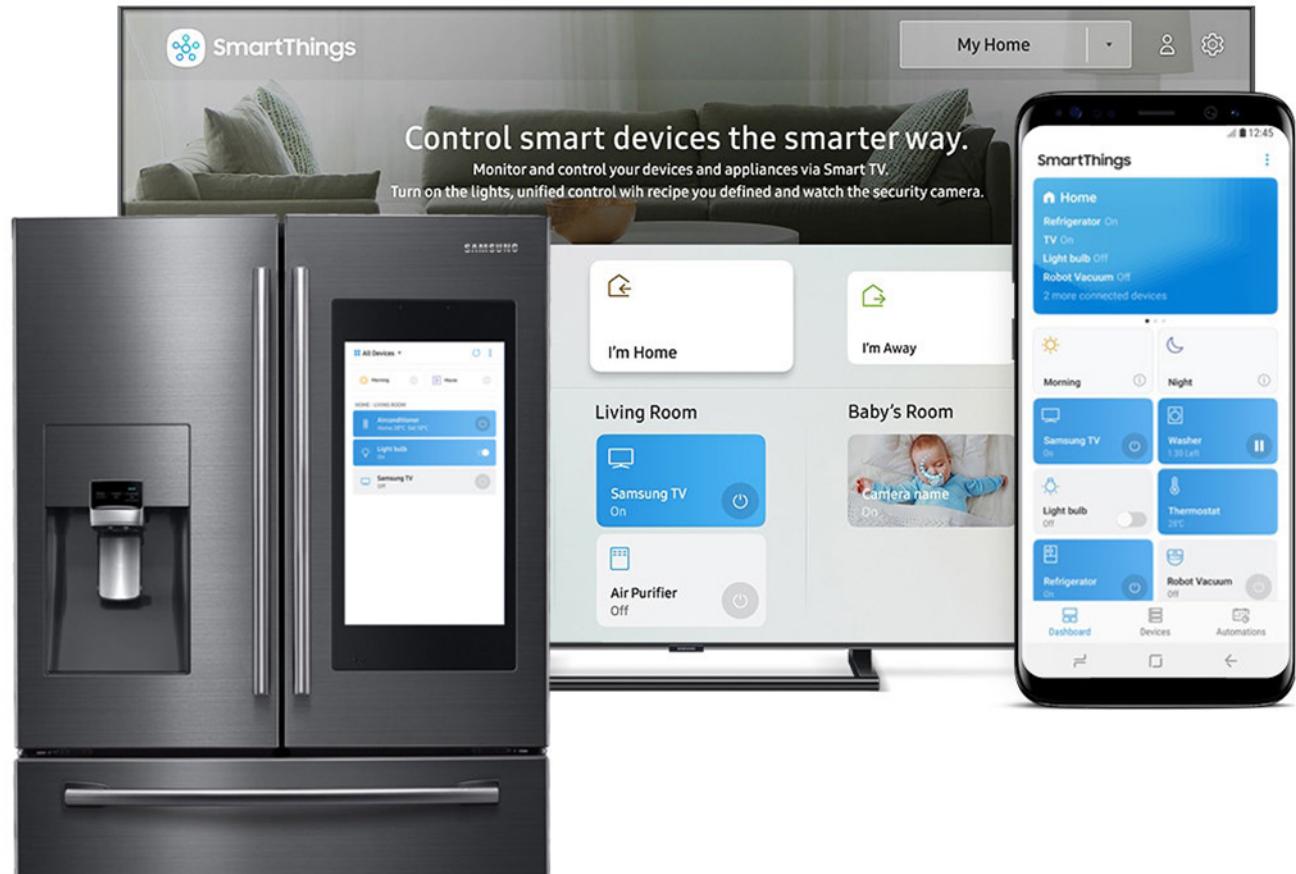


Why Connect To SmartThings?

Access and control your devices from millions of places

Leverage ecosystem of over 4000 different IoT devices to create smart home automations

Open up bundling options in hundreds of markets worldwide



ST Schema Integration

- Schema-based device discovery eliminates the requirement to define device profiles for each device variant
- Built-in UI and simplified event model reduces the amount of code needed in the connector
- Supports the full range of SmartThings platform capabilities, no compromise in device functionality

ST Schema Connector Form

Hosting and Credentials Catalog Info

Where are you hosting your connector app?

AWS Lambda

Host the app in AWS Lambda with your AWS account, and grant SmartThings permission to execute the Lambda.

WebHook Endpoint

Host the app on your public server. The server needs to support https with SSL certificates.

Target URL



Device Cloud Credentials

Client ID

The client ID you generated from your cloud

Client Secret

The client secret generated from your cloud

Authorization URI

The OAuth2 authorization url for your cloud

Refresh Token URL

The OAuth2 token url from your cloud

Partner OAuth scope (Optional)

The scopes required for your cloud. Separate scopes by "Comma" or "Tab"

ST Schema Connector

Simplified Event Model

Discovery

Return a description of all devices

State Refresh

Return current state of each device

Command

Handle commands from the SmartThings platform, returning updated device state

```
JS index.js  x
1 "use strict";
2
3 const { lambda } = require('st-schema');
4 const { lib } = require('st-schema');
5 const requestConfig = require("./requestConfig.json");
6 const Library = new lib(requestConfig);
7
8 +async function discoveryRequest(request, response) {
15 }
16
17 +async function commandRequest(request, response) {
56 }
57
58 +async function stateRefreshRequest(request, response) {
72 }
73
74 +module.exports.handler = lambda({
78 });
79
80 +function getDeviceHandler(partnerCapabilities) {
86 }
```

Less Coding. No need to define UI pages, handle OAuth token exchange, or explicitly manage the creation and deletion of devices on the SmartThings platform

Getting Published

Two Options to Suit Your Product

SmartThings-Certified

SmartThings functional certification & security review

Premium listing in SmartThings app
Use WWST logo for packaging & website

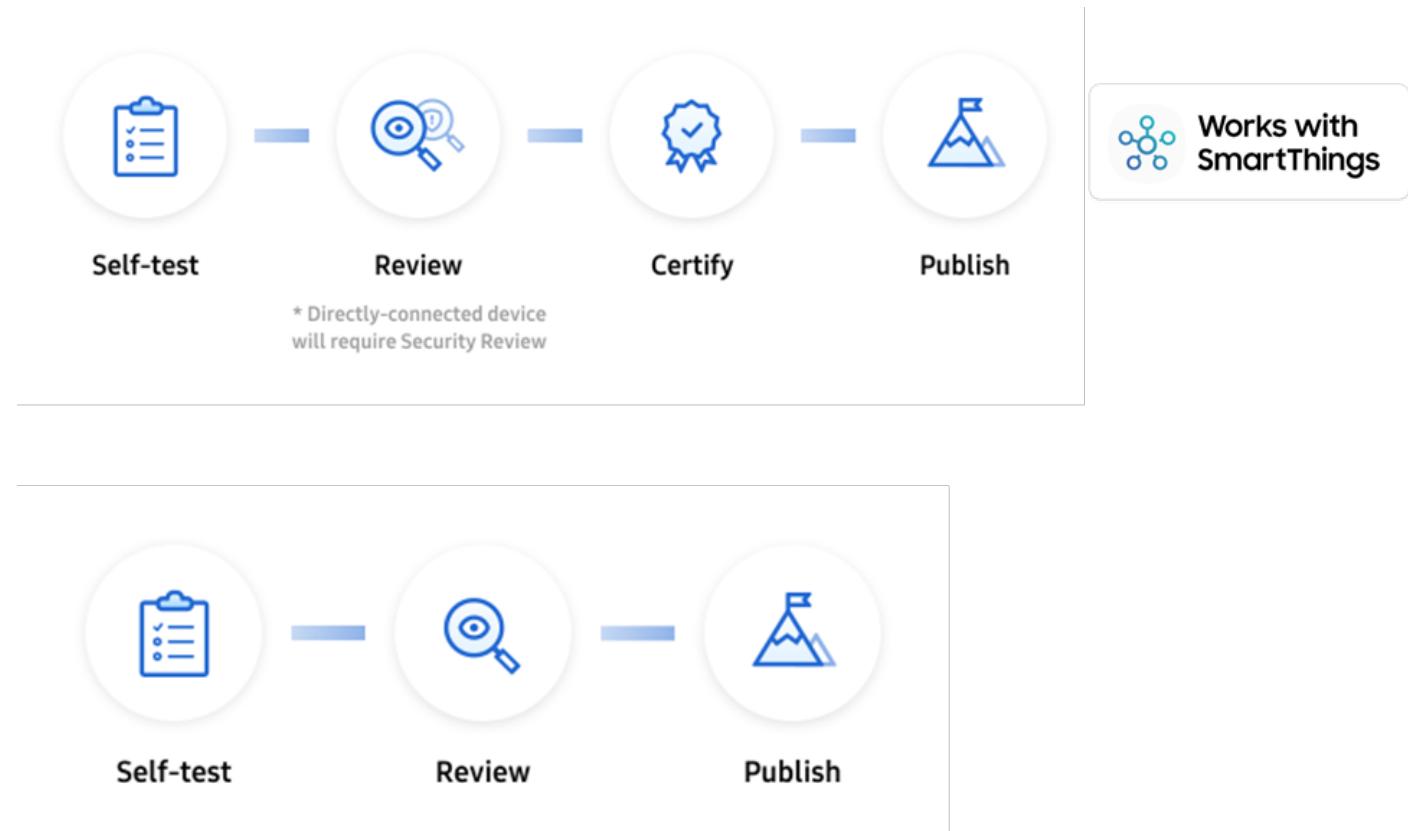
You must provide device state change callbacks

SmartThings-Compatible

Use guidelines to self-test and verify

Fast path to listing in SmartThings app

Available now for cloud-connected devices





SmartThings
Developers

CODE WALKTHROUGH

glitch.com/~iot-fuse-code

Agenda

- Brief introduction to Hub Connected Devices
- Walk through the process of on-boarding and testing a “new” contact sensor
- Modify the DTH to expose some new capabilities
- Use the sensor to control the lights

Hub Connected Device Integration

- Includes Zigbee, Z-Wave and Wi-Fi devices with a local API
- Currently supported in SmartThings via Device Type Handlers (DTHs)
- DTHs are Groovy code that convert protocol specific messages into SmartThings capabilities
- There are DTHs for many types of devices and you can also modify or create new ones

Zigbee DTH Example

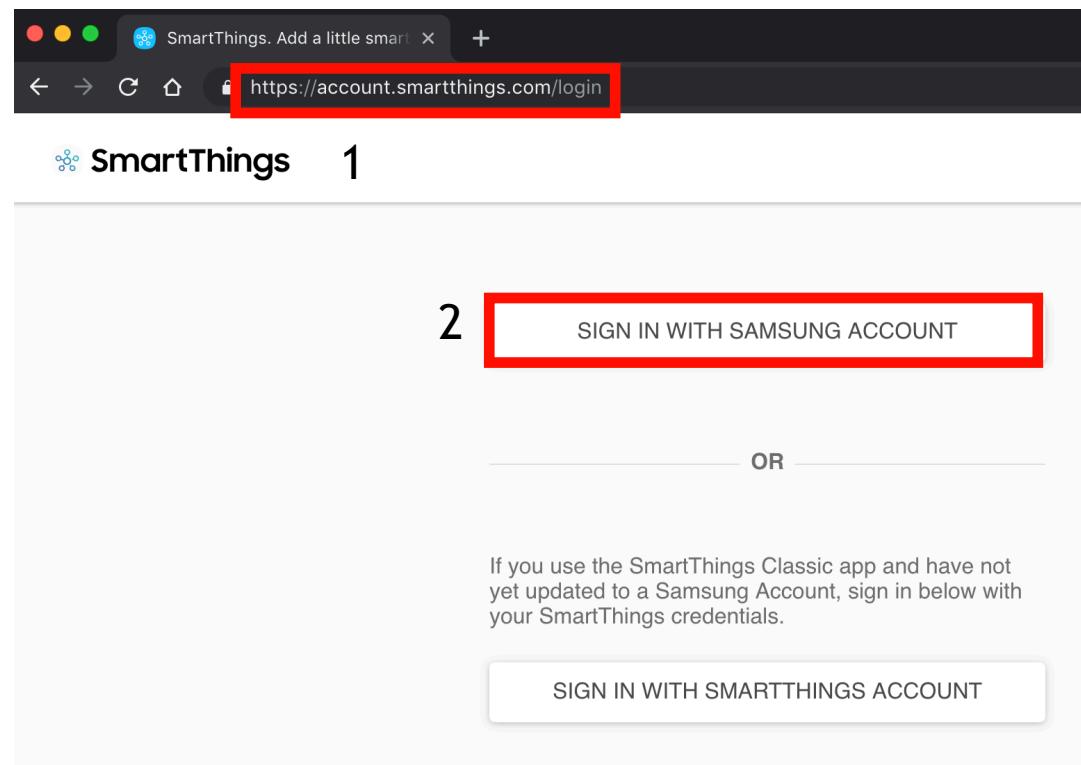
Source Code

```
metadata {  
    definition (name: "ZigBee Dimmer") {  
        capability "Switch"  
        capability "Switch Level"  
    }  
}  
  
def parse(String description) {  
    // Converts device status into an event  
    def event = zigbee.getEvent(description)  
    sendEvent(event)  
}  
  
def off() {  
    zigbee.off() // Sends the "Off" command  
}  
  
def on() {  
    zigbee.on() // Sends the "On" command  
}  
  
def setLevel(value) {  
    zigbee.setLevel(value) // Sends a command to change the brightness  
}
```

Maps the Zigbee On/Off and Level Control Clusters
to SmartThings Switch and Switch Level capabilities

Add Support for New Device

- Soon you will be able to add hub connected devices via the Developer Workspace at <https://smartthings.developer.samsung.com>
- For now login to <https://account.smartthings.com/>



Add Support for New Device

- Email: demo.smarthings@gmail.com
- Password: 6^xVad9tBs9f



Samsung account

Email or phone number

demo.smarthings@gmail.com



Password

.....

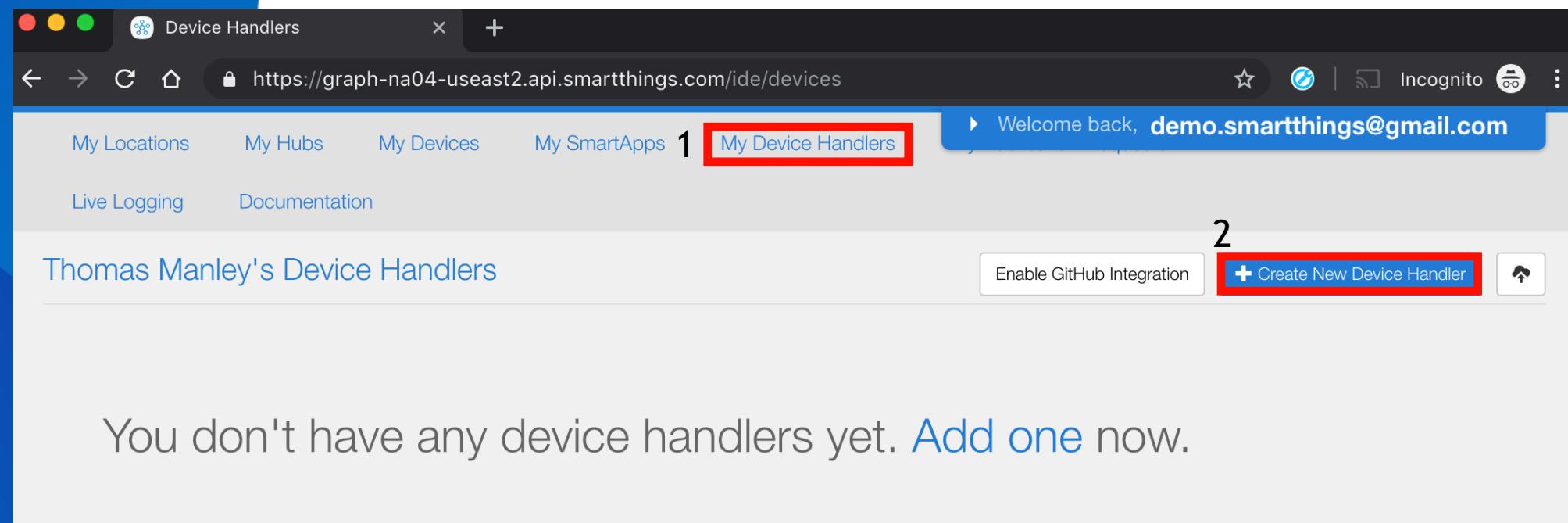


Remember my ID

SIGN IN

Add Support for New Device

- Navigate to <https://graph-na04-useast2.api.smarthings.com/>
- Create New Device Handler

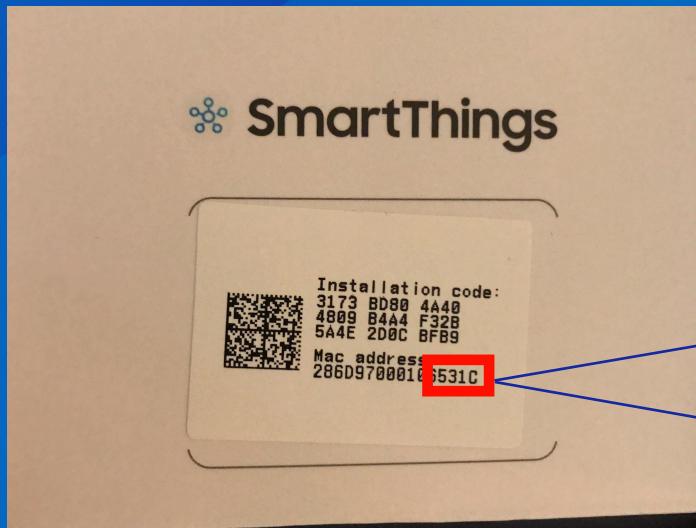


Add Support for New Device

The screenshot shows the 'Create New Device Handler' page in the SmartThings IDE. The URL in the browser is <https://graph-na04-useast2.api.smartthings.com/ide/device/create>. The page has a navigation bar with links for My Locations, My Hubs, My Devices, My SmartApps, My Device Handlers, and My Projects. Below the navigation bar is a 'Documentation' link. The main content area is titled 'Create New Device Handler'. It contains a paragraph of text explaining the purpose of the page, mentioning Zigbee Home Automation (HA), Zigbee Light-Link (ZLL), and Z-Wave protocols. It also links to the SmartThings Developer Workspace and provides information on certification and pairing. Three numbered steps are overlaid on the interface:

1. A button labeled 'From Template' is highlighted with a red box.
2. A dropdown menu labeled 'Protocol' is open, showing a list of options. The option 'Zigbee Home Automation (HA) or 3.0' is highlighted with a red box.
3. A sub-menu titled 'Select Protocol' is displayed, listing 'Zigbee Home Automation (HA) or 3.0', 'Zigbee Light Link (ZLL)', 'Z-Wave', and 'LAN/Other'.

Add Support for New Device



Use the last 4 digits of the Mac address in the Model Name and Device Join Name

From Template From Code From Example

Protocol
Zigbee Home Automation (HA) or :
Template Capabilities

1 Battery
 Color Control
 Color Temperature
 Contact Sensor
 Lock
 Lock Codes
 Motion Sensor
 Power Meter
 Power Source
 Smoke Detector
 Switch
 Switch Level
 Temperature Measurement
 Thermostat Cooling Setpoint
 Thermostat Fan Mode
 Thermostat Heating Setpoint
 Thermostat Mode
 Thermostat Operating State
 Water Sensor

Other Capabilities

Matching Device Handler Found!

2 SmartSense Open/Closed Sensor

Fill out and submit this form with information about your device to have it pair and b using this existing device handler

Device Handler Update form

3 **Zigbee Manufacturer Name**
Samjin

The name specified in the ManufacturerName attribute (0x0004) in the Basic Cluster (0x0000). e.g.

4 **Zigbee Model Name**
multi-531C

The name specified in the ModelIdentifier attribute (0x0005) in the Basic Cluster (0x0000). e.g. ACI

5 **Zigbee Device ID (optional)**
Enter device ID

The four digit hex code that specifies the device in the selected application profile, as defined in the

6 **Device Join Name**
multi-531C

The device name that you want to show when it pairs in the SmartThings App. for example ACME

7 **Update Existing Device Handler**

Add Support for New Device

DO NOT NAVIGATE AWAY FROM THIS TAB/WINDOW

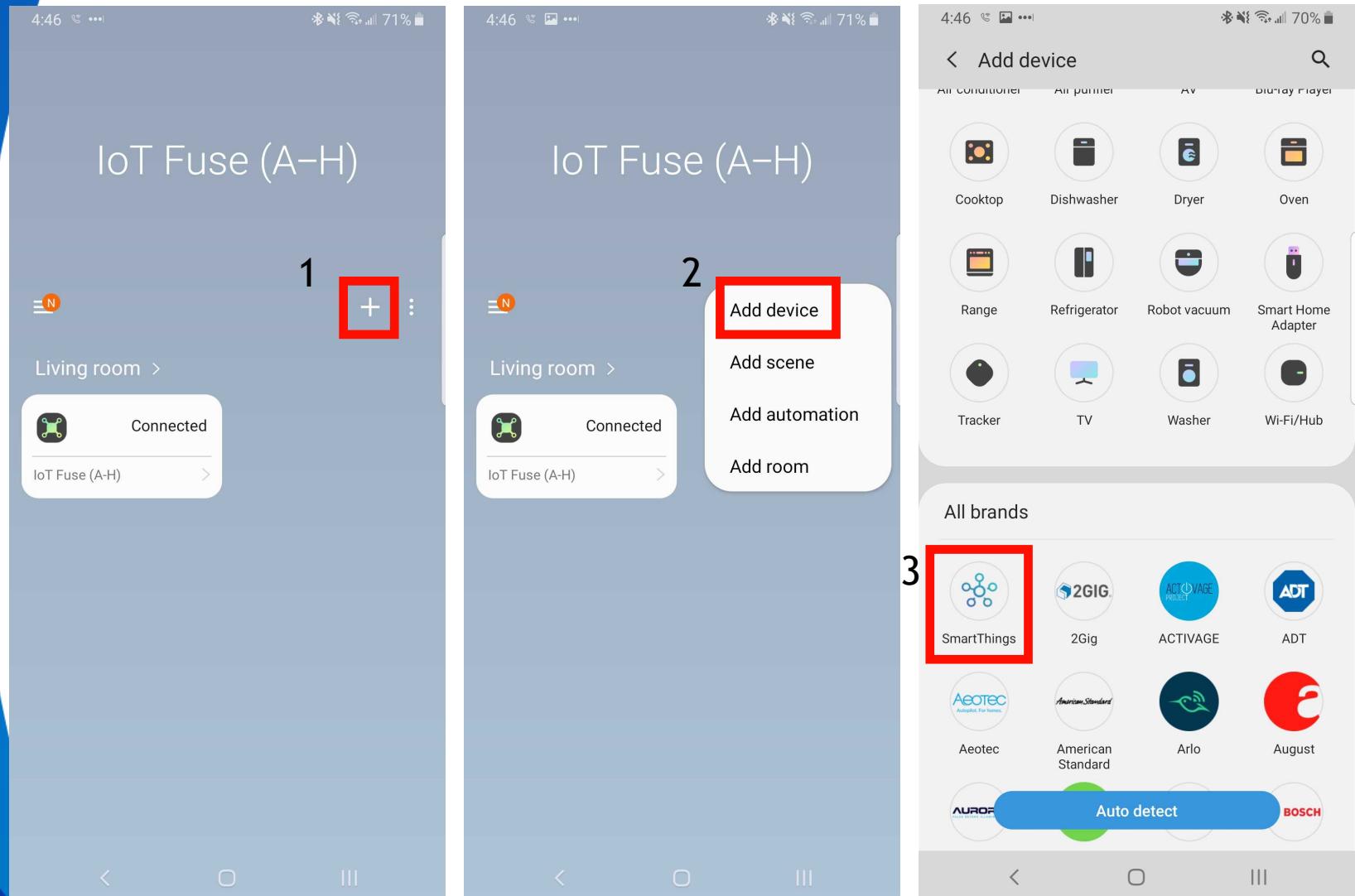
SmartSense Open/Closed Sensor

2 3

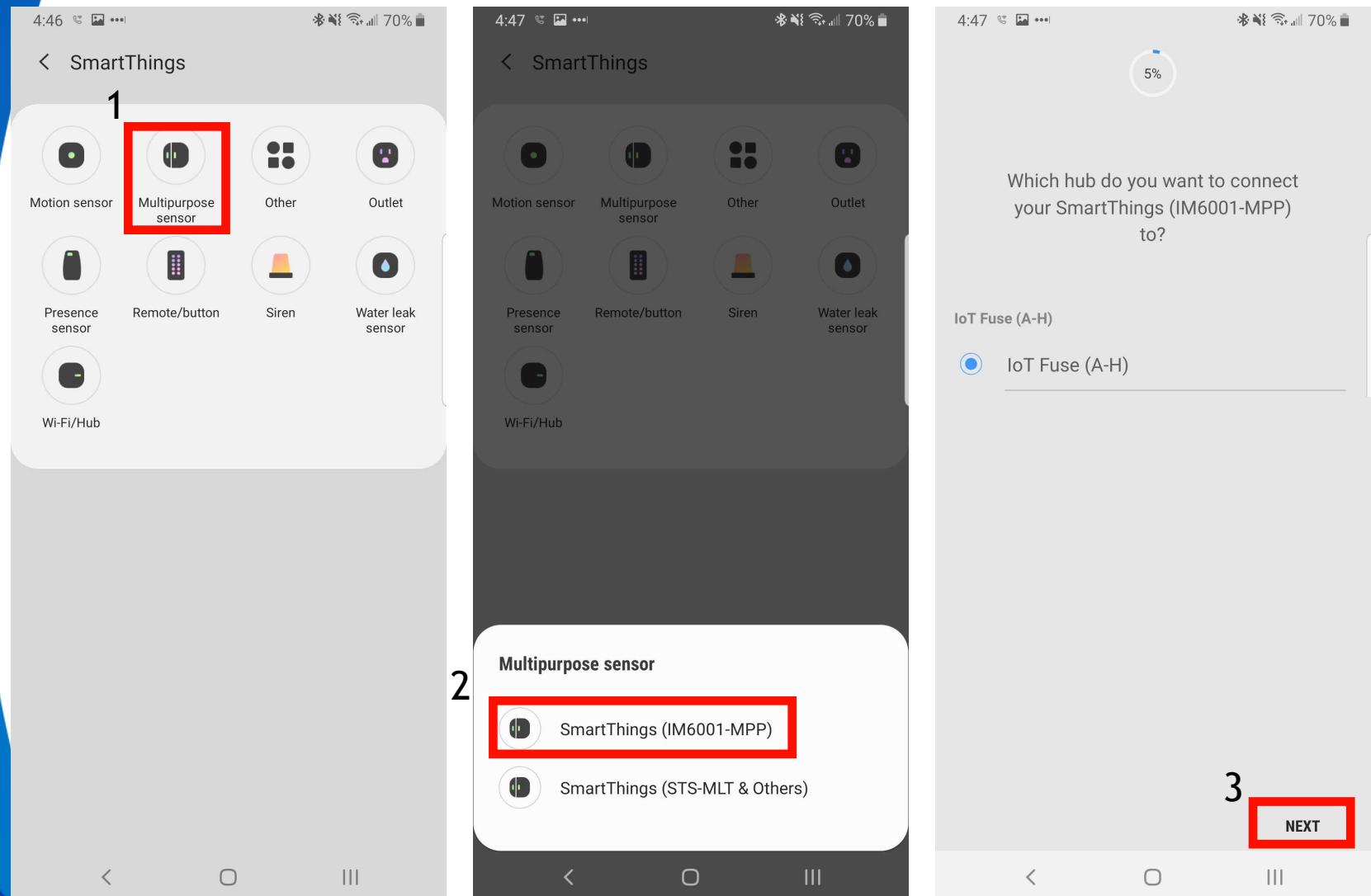
Save Publish IDE Settings Device Type Settings Simulator

```
26     capability "Health Check"
27     capability "Sensor"
28
29     command "enrollResponse"
30
31
32     fingerprint inClusters: "0000,0001,0003,0402,0500,0020,0B05", outClusters: "0019", manufacturer: "CentraLite"
33     fingerprint inClusters: "0000,0001,0003,0402,0500,0020,0B05", outClusters: "0019", manufacturer: "CentraLite"
34     fingerprint inClusters: "0000,0001,0003,0020,0402,0500,0B05", outClusters: "0019", manufacturer: "CentraLite"
35     fingerprint inClusters: "0000,0001,0003,0020,0402,0500,0B05", outClusters: "0019", manufacturer: "CentraLite"
36     fingerprint inClusters: "0000,0001,0003,0020,0402,0500,0B05", outClusters: "0019", manufacturer: "CentraLite"
37     fingerprint inClusters: "0000,0001,0003,0402,0500,0020,0B05", outClusters: "0019", manufacturer: "Visonic",
38     fingerprint inClusters: "0000,0001,0003,0020,0402,0500,0B05", outClusters: "0019", manufacturer: "Ecolink",
39     fingerprint inClusters: "0000,0001,0003,0020,0402,0500,0B05", profileId: "0104", manufacturer: "Samjin", model: "multi-531C", deviceJoinName: "multi-531C"
40   }
41
42 }
```

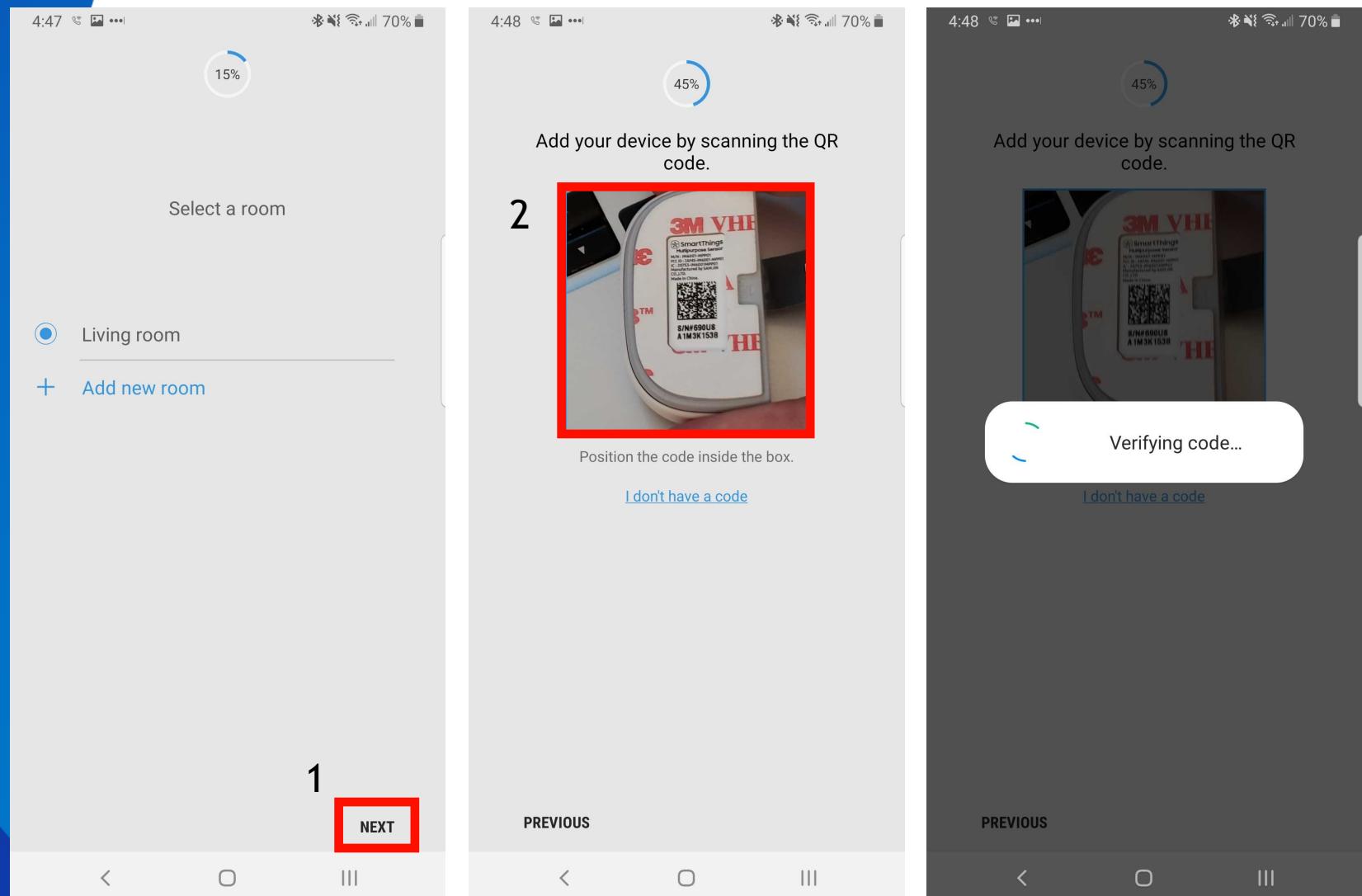
Join the Device (Android)



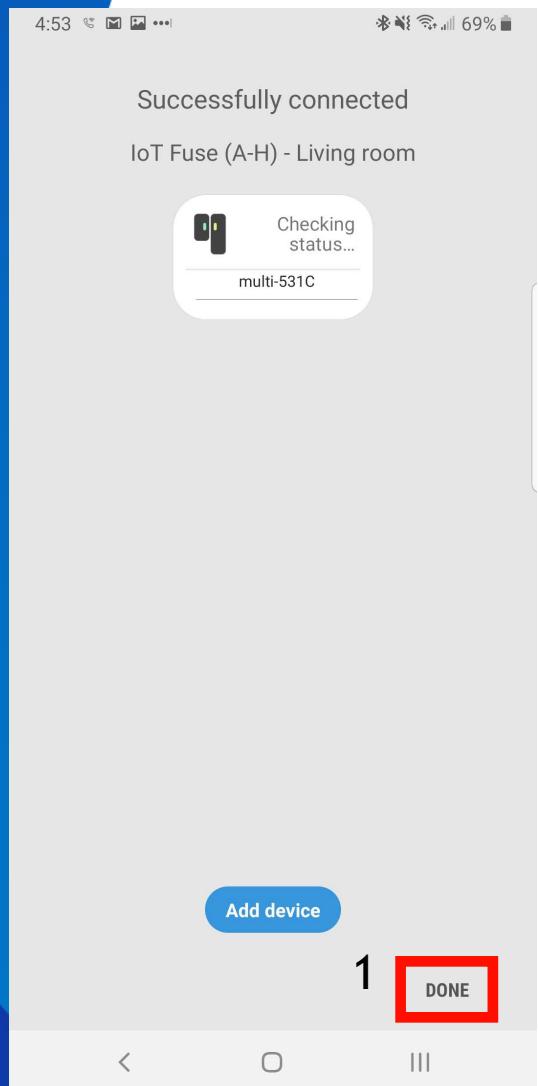
Join the Device (Android)



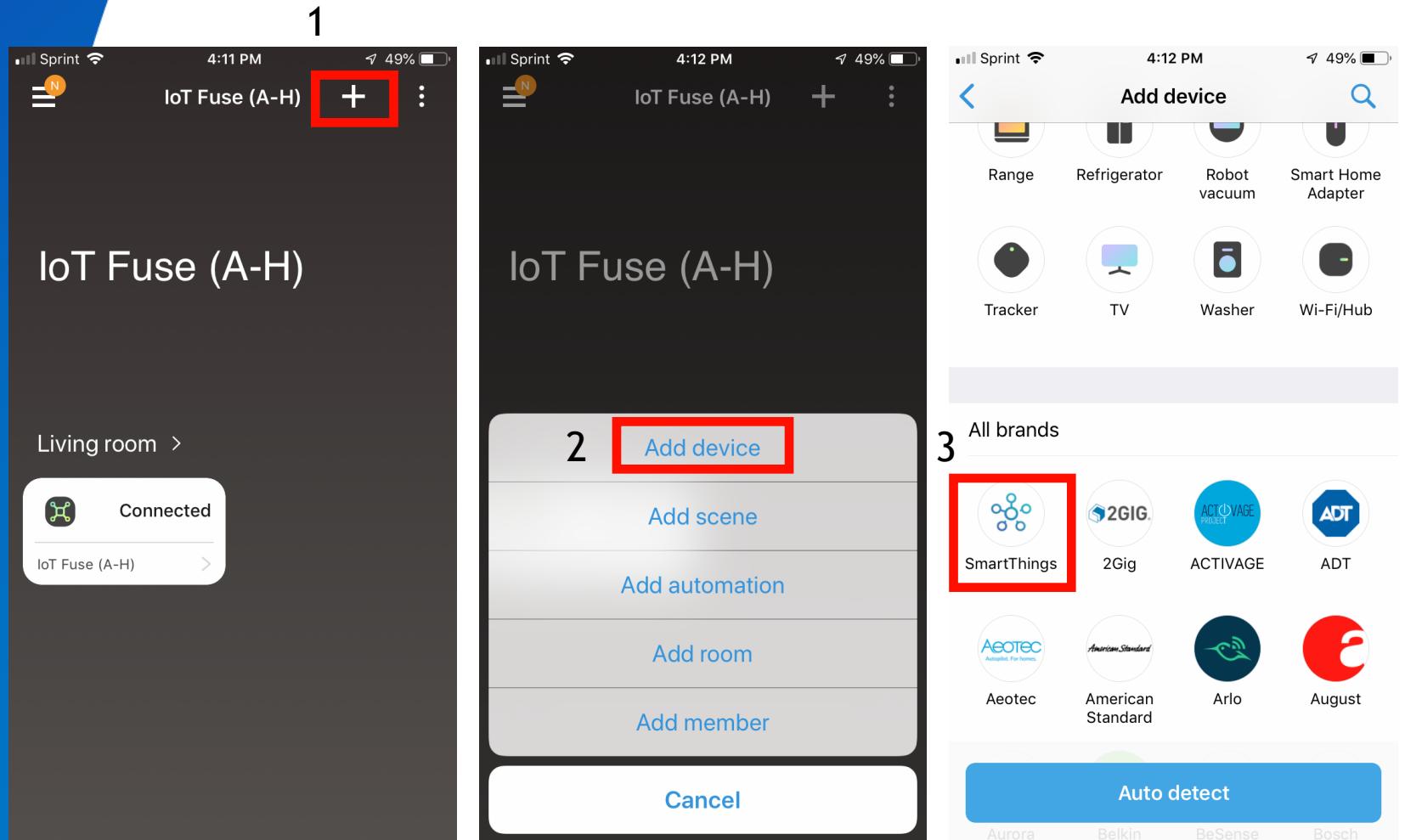
Join the Device (Android)



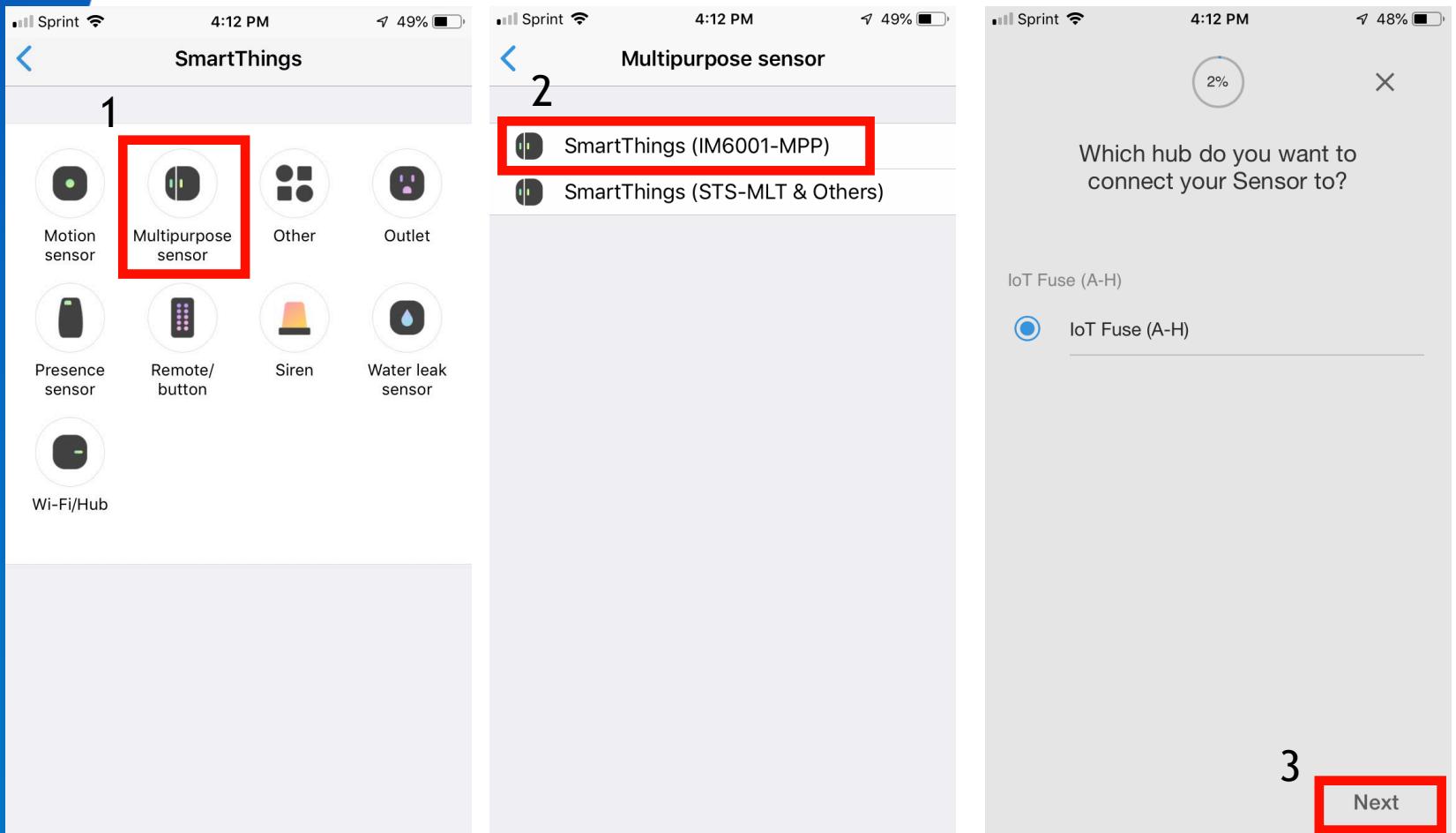
Join the Device (Android)



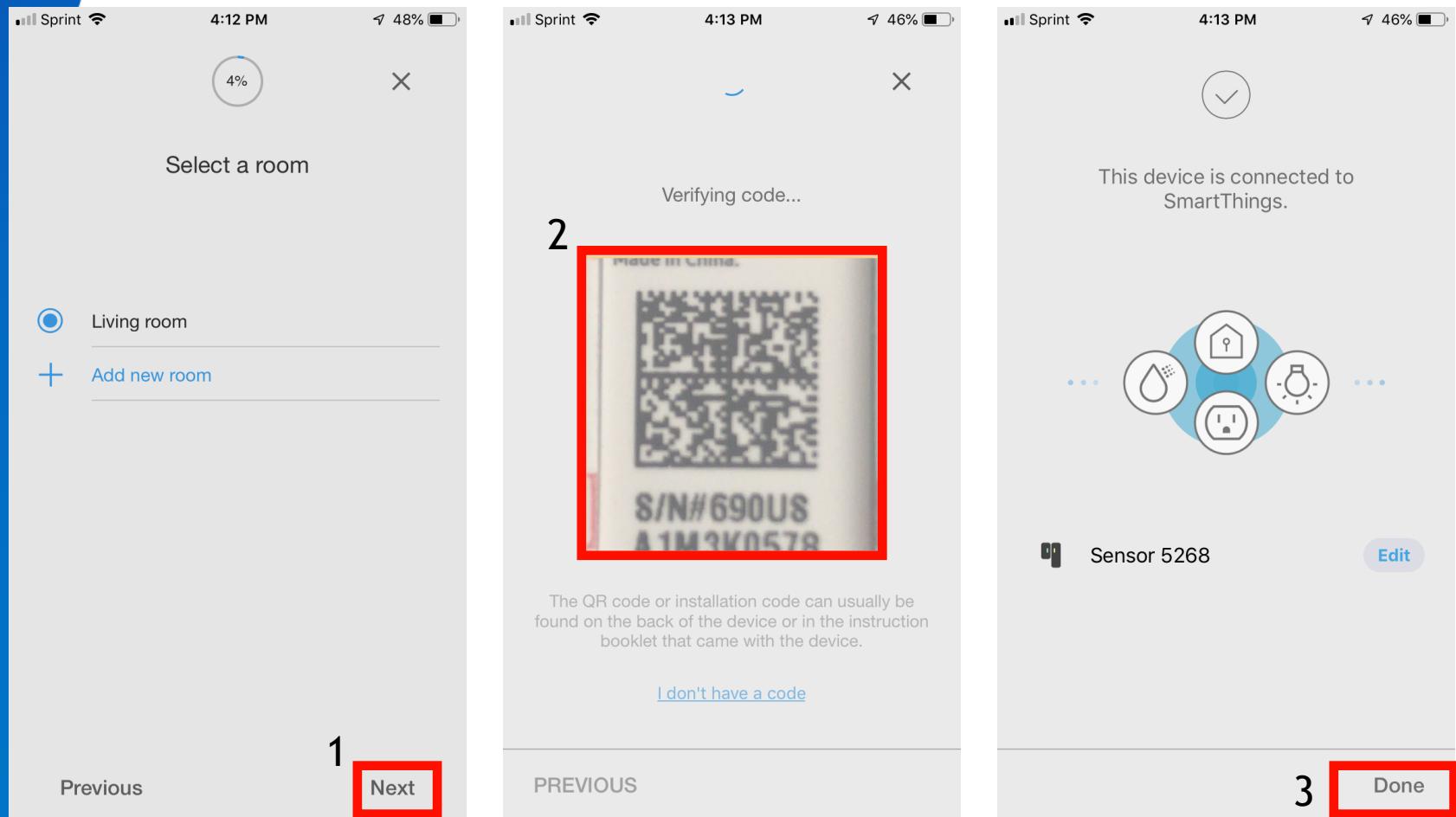
Join the Device (iOS)



Join the Device (iOS)



Join the Device (iOS)



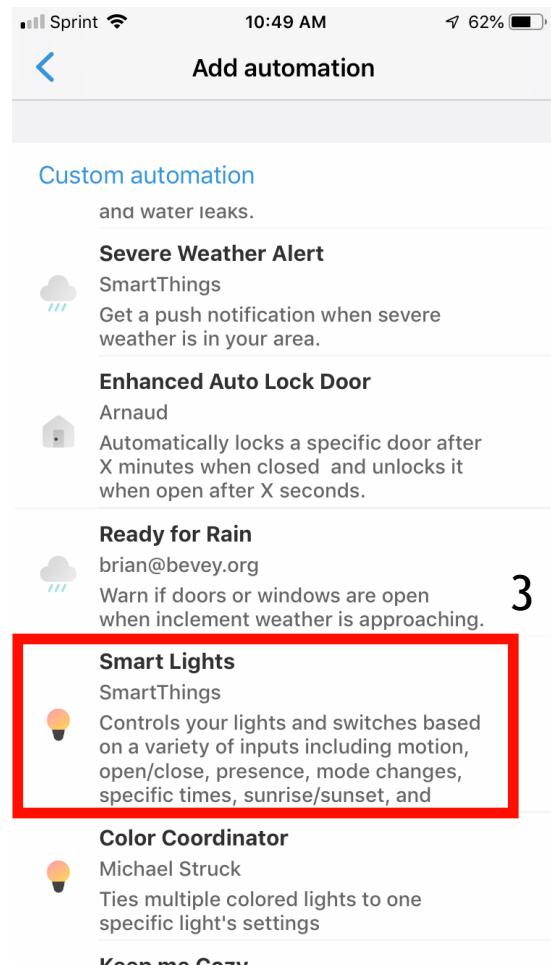
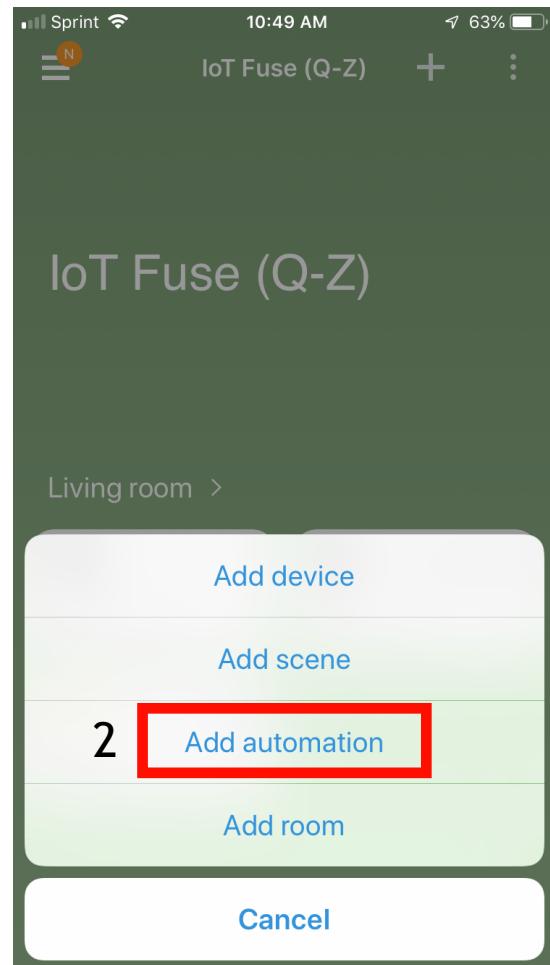
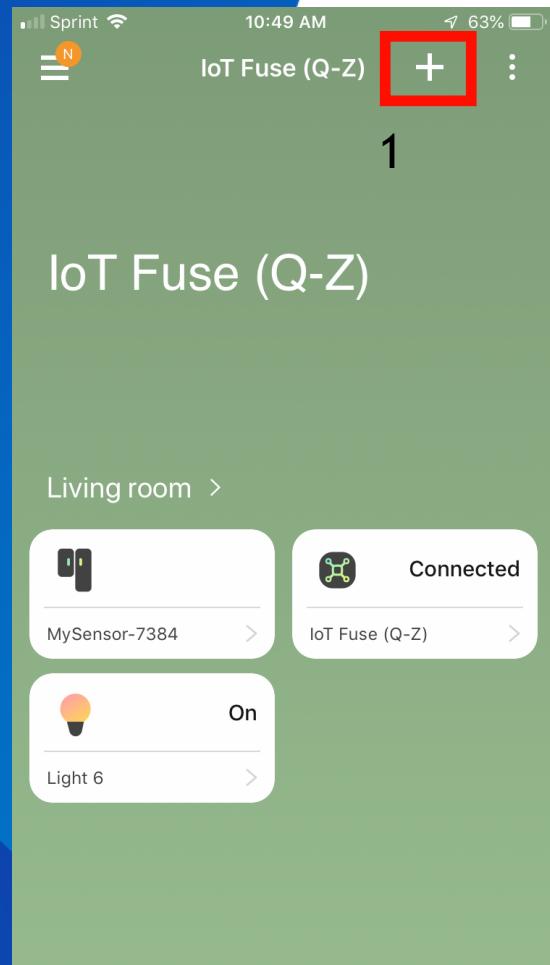
A graphic element consisting of several blue diagonal stripes of varying shades, starting from white at the top left and transitioning to dark blue at the bottom right.

**Play with Test
the Device**

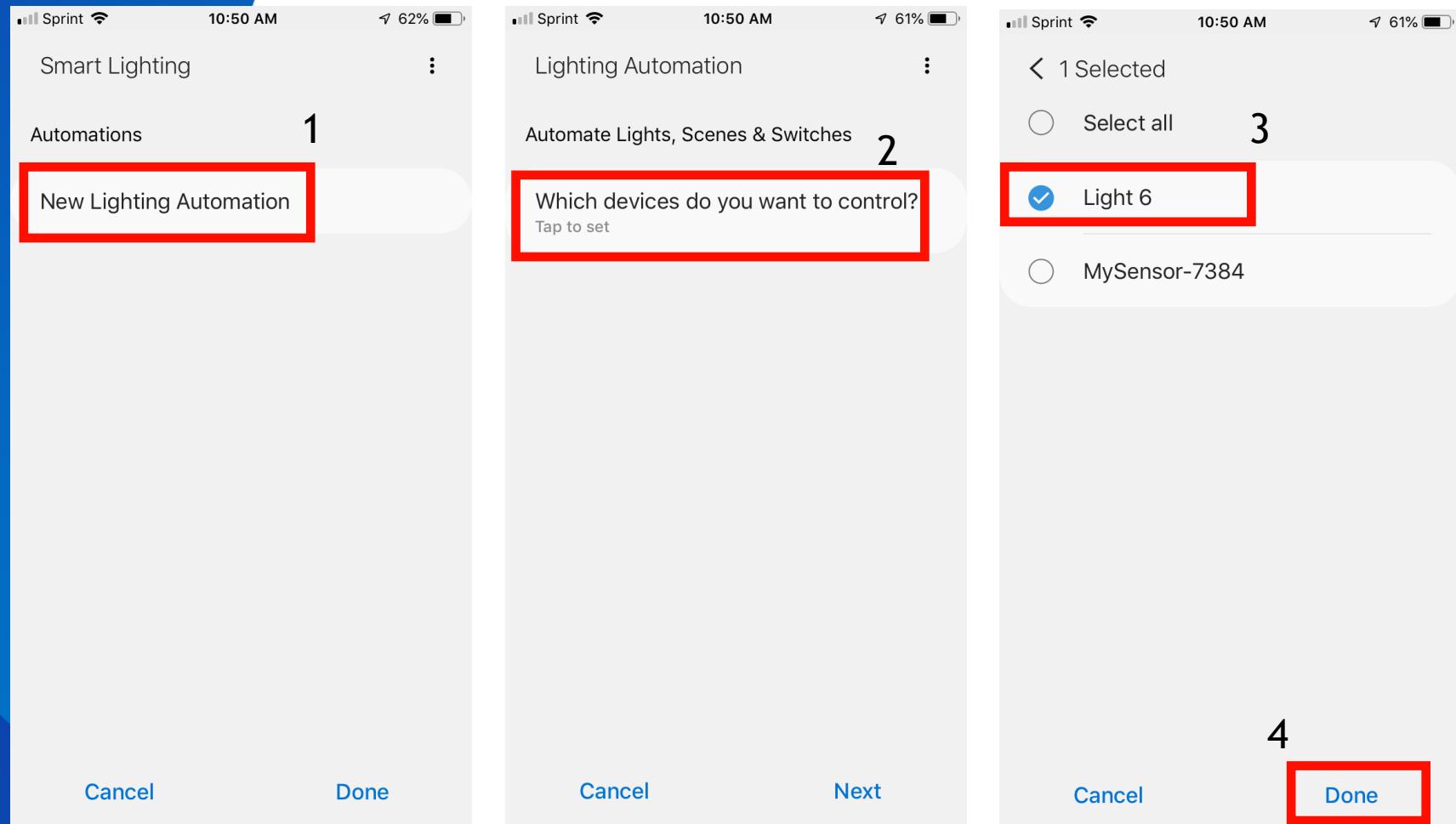
Implement a Custom Device Type Handler

- Grab the code from our conference repo: <https://raw.githubusercontent.com/tpmanley/conferences/master/iotfuse-2019/smartsense-multi-dimmer.groovy>
- Edit the Device Type Handler you created earlier and replace all the code with that from the above repository
- Modify the new Device Type Handler to add custom functionality to the device

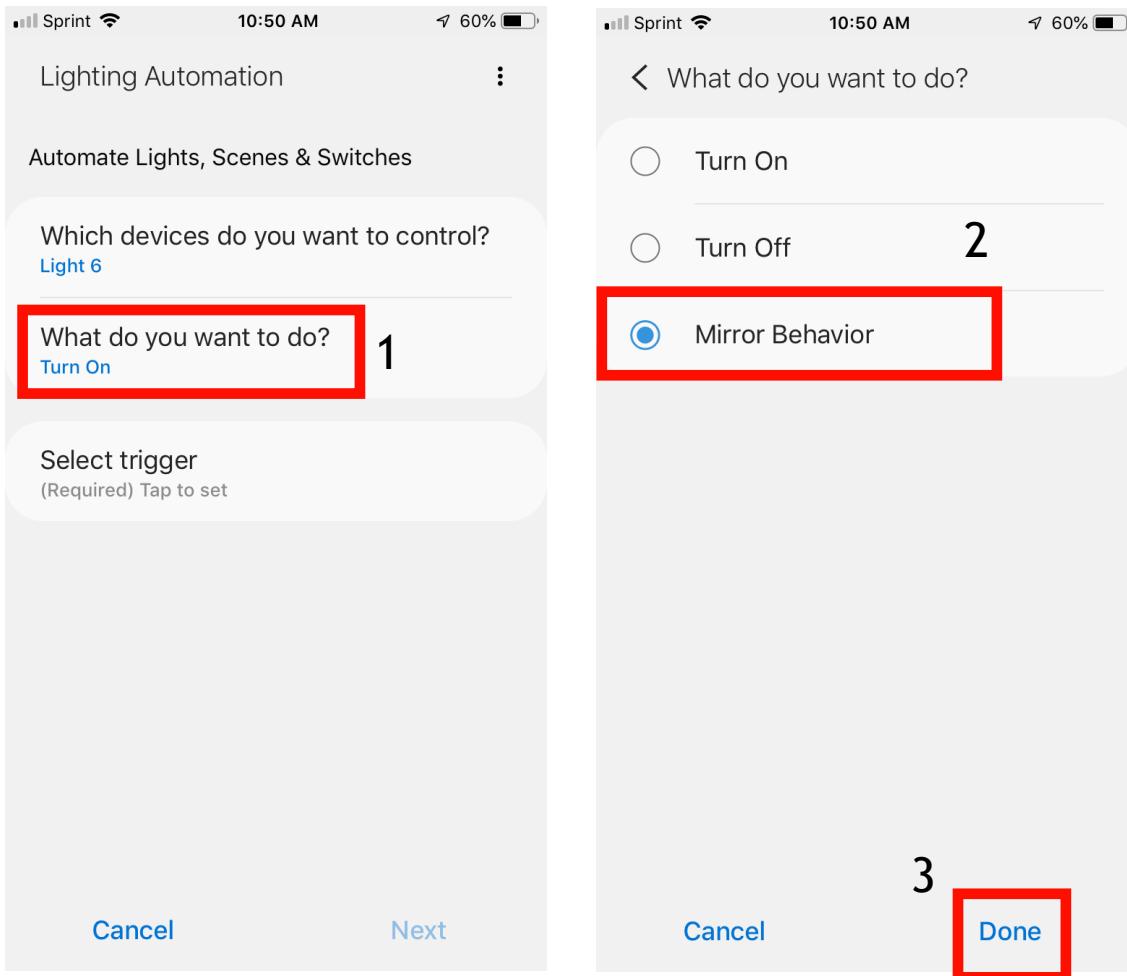
Setup Bulb Automation



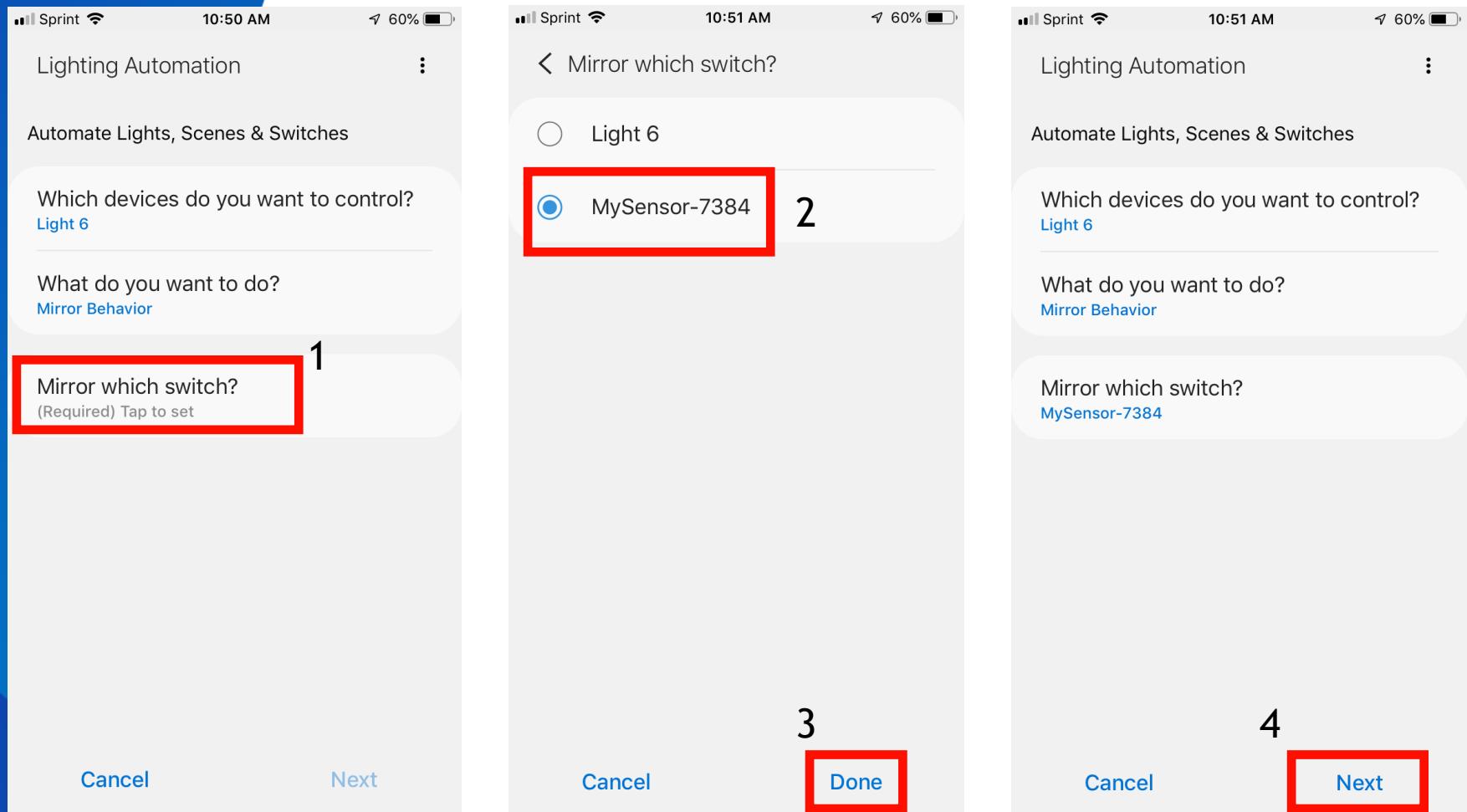
Setup Bulb Automation



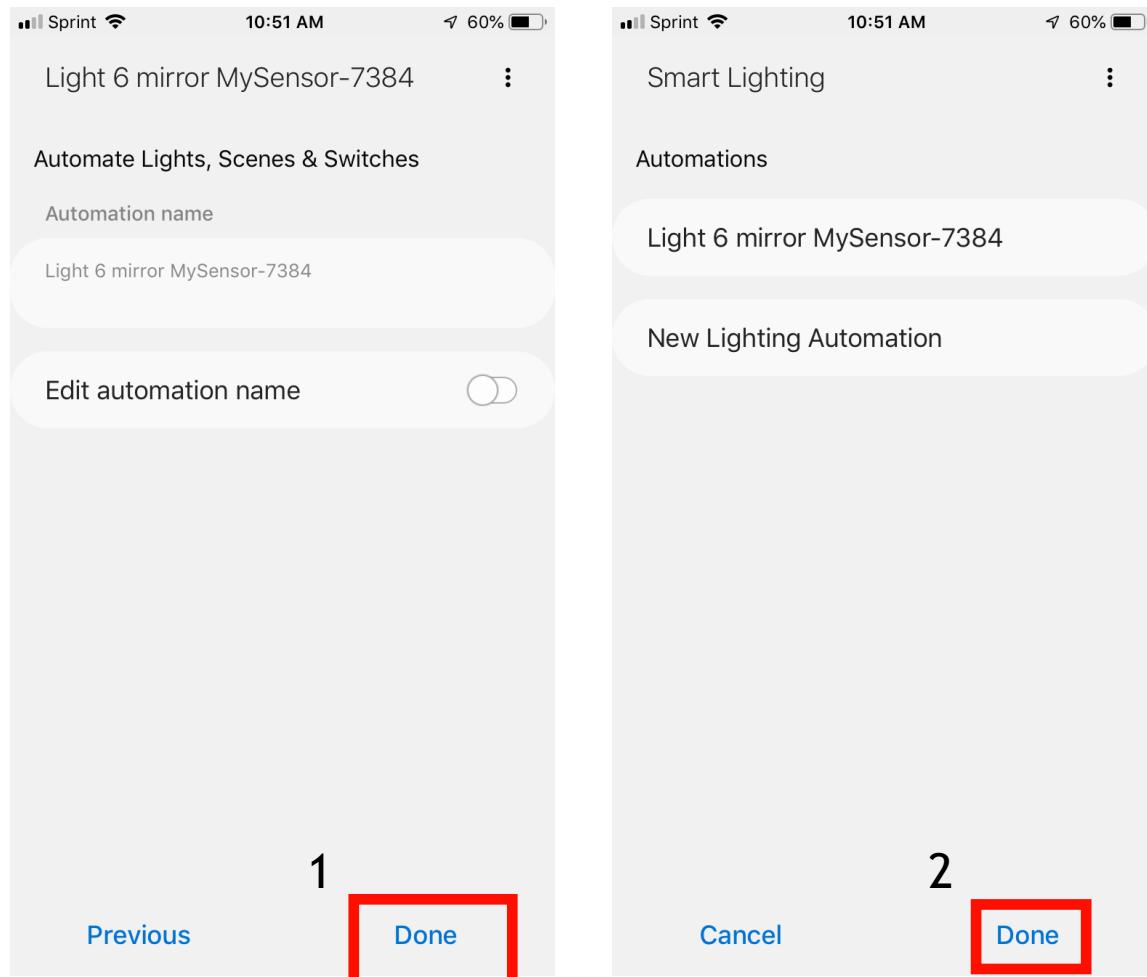
Setup Bulb Automation



Setup Bulb Automation



Setup Bulb Automation





SmartThings
Developers

Thank You
(...and we are hiring!)

glitch.com/~iot-fuse-code
github.com/tpmanley/conferences