Particle

PARTICLE 102 - INTRODUCING PARTICLE PRIMITIVES, THE DEVICE CLOUD, MESH, BLE, AND NFC

MANAGING DEVICES FROM THE CONSOLE

WORKING WITH PARTICLE PRIMITIVES

INTRODUCING PARTICLE GEN3 & MESH

MESH PUBLISH & SUBSCRIBE

BLUETOOTH & NFC

MANAGING DEVICES FROM THE CONSOLE

WORKING WITH PARTICLE PRIMITIVES

INTRODUCING PARTICLE GEN3 & MESH

MESH PUBLISH & SUBSCRIBE

BLUETOOTH & NFC

CONSOLE.PARTICLE.IO

Particle

NEW
Intelligent OTA
Learn More

Docs | Contact Sales | Support | brandon@particle.io

Devices

Number of devices: 68

ID	Type	Name	Last Handshake
e00fce689982644ef5f2c864	B Series	neuron-one	8/28/19 at 4:15pm
e00fce68f0d1a741897238a5	Argon	tf-lite-tester	8/28/19 at 12:20pm
e00fce688a86c51239f46f8e	B Series	neuron-two	8/28/19 at 12:03pm
e00fce68998df7b0c4c53388	Argon	dock-demo	8/27/19 at 9:44pm
e00fce681bf6727481217a19	Argon	emotion-mesh-gateway	8/6/19 at 5:06pm
e00fce68c956724a12365172	Xenon	emotion-pixel-04	8/6/19 at 3:42pm
e00fce683f3e7d3386ef6cc7	Xenon	emotion-pixel-01	8/6/19 at 3:41pm
e00fce6836e622751cf2e6f3	Xenon	emotion-pixel-03	8/6/19 at 11:40am
e00fce68f4e883be2fb41a6b	Argon	tc-beacon-03	8/5/19 at 4:06pm
e00fce68bc7da5f8a6a90871	Xenon	dock-demo-node	8/5/19 at 2:33pm

1 2 3 4 5 6 7 »

CONSOLE.PARTICLE.IO

Particle

NEW
Intelligent OTA
Learn More

Docs | Contact Sales | Support | brandon@particle.io

Devices

Number of devices: 68

ID	Type	Name	Last Handshake
e00fce689982644ef5f2c864	B Series	neuron-one	8/28/19 at 4:15pm
e00fce68f0d1a741897238a5	Argon	tf-lite-tester	8/28/19 at 12:20pm
[illegible]	B Series	neuron-two	8/28/19 at 12:03pm
[illegible]	Argon	dock-demo	8/27/19 at 9:44pm
[illegible]	Argon	emotion-mesh-gateway	8/6/19 at 5:06pm
[illegible]	Xenon	emotion-pixel-04	8/6/19 at 3:42pm
e00fce683f3e7d3386ef6cc7	Xenon	emotion-pixel-01	8/6/19 at 3:41pm
e00fce6836e622751cf2e6f3	Xenon	emotion-pixel-03	8/6/19 at 11:40am
e00fce68f4e883be2fb41a6b	Argon	tc-beacon-03	8/5/19 at 4:06pm
e00fce68bc7da5f8a6a90871	Xenon	dock-demo-node	8/5/19 at 2:33pm

1 2 3 4 5 6 7 »

All your devices

CONSOLE.PARTICLE.IO

Particle

NEW
Intelligent OTA
Learn More

Docs | Contact Sales | Support | brandon@particle.io

Devices

Number of devices: 68

ID	Type	Name	Last Handshake
e00fce689982644ef5f2c864	B Series	neuron-one	8/28/19 at 4:15pm
e00fce68f0d1a741897238a5	Argon	tf-lite-tester	8/28/19 at 12:20pm
e00fce688a86c51239f46f8e	B Series	neuron-two	8/28/19 at 12:03pm
	Argon	dock-demo	8/27/19 at 9:44pm
	Argon	emotion-mesh-gateway	8/6/19 at 5:06pm
	Xenon	emotion-pixel-04	8/6/19 at 3:42pm
e00fce683f3e7d3386ef6cc7	Xenon	emotion-pixel-01	8/6/19 at 3:41pm
e00fce6836e622751cf2e6f3	Xenon	emotion-pixel-03	8/6/19 at 11:40am
e00fce68f4e883be2fb41a6b	Argon	tc-beacon-03	8/5/19 at 4:06pm
e00fce68bc7da5f8a6a90871	Xenon	dock-demo-node	8/5/19 at 2:33pm

Their type and name

1 2 3 4 5 6 7 »

CONSOLE.PARTICLE.IO

Particle

NEW
Intelligent OTA
Learn More

Docs | Contact Sales | Support | brandon@particle.io

Devices

Number of devices: 68

ID	Type	Name	Last Handshake
e00fce689982644ef5f2c864	B Series	neuron-one	8/28/19 at 4:15pm
e00fce68f0d1a741897238a5	Argon	tf-lite-tester	8/28/19 at 12:20pm
e00fce688a86c51239f46f8e	B Series	neuron-two	8/28/19 at 12:03pm
	Argon	dock-demo	8/27/19 at 9:44pm
	Argon	emotion-mesh-gateway	8/6/19 at 5:06pm
	Xenon	emotion-pixel-04	8/6/19 at 3:42pm
e00fce683f3e7d3386ef6cc7	Xenon	emotion-pixel-01	8/6/19 at 3:41pm
e00fce6836e622751cf2e6f3	Xenon	emotion-pixel-03	8/6/19 at 11:40am
e00fce68f4e883be2fb41a6b	Argon	tc-beacon-03	8/5/19 at 4:06pm
e00fce68bc7da5f8a6a90871	Xenon	dock-demo-node	8/5/19 at 2:33pm

1 2 3 4 5 6 7 »

And the last time they appeared online

REAL-TIME EVENT LOGS

Particle

Devices > View Device

PING | EDIT

ID: 3b001e00024736333934363 8

Name: trash-panda

Device OS: 0.8.0-rc.8

Type: Photon

Serial Number: PHHMAB819ZY6QXD

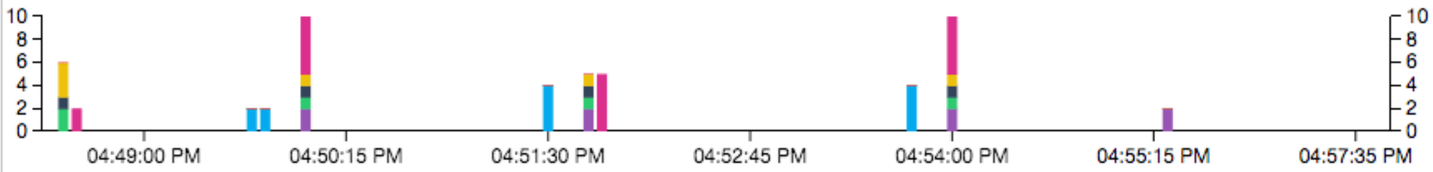
Last Handshake: Jul 19th 2018, 4:56 pm

Notes

Click the edit button to keep notes on this device, like 'Deployed to customer site'.

EVENT LOGS | DIAGNOSTICS NEW

spark/flash/status | device went offline | device came online | spark/device/last_reset | spark/device/diagnostics/update | spark/device/app-hash



PAUSE | SEE IN TERMINAL | PUBLISH EVENT

EVENT NAME	DATA	PUBLISHED AT
spark/device/diagnostics/update	{"device":{"system":	*July 19th at 4:56:39 pm*
spark/device/last_reset	power_down	*July 19th at 4:56:39 pm*
device came online		*July 19th at 4:56:39 pm*

DEVICE VITALS NEW

Jul 19th, 2018, 04:22PM

- **Strong** Wi-Fi signal
- **0** disconnect events
- **146ms** round-trip time
- **0** rate-limited publishes
- **35kB** of 81kB RAM used

Download History | Run diagnostics

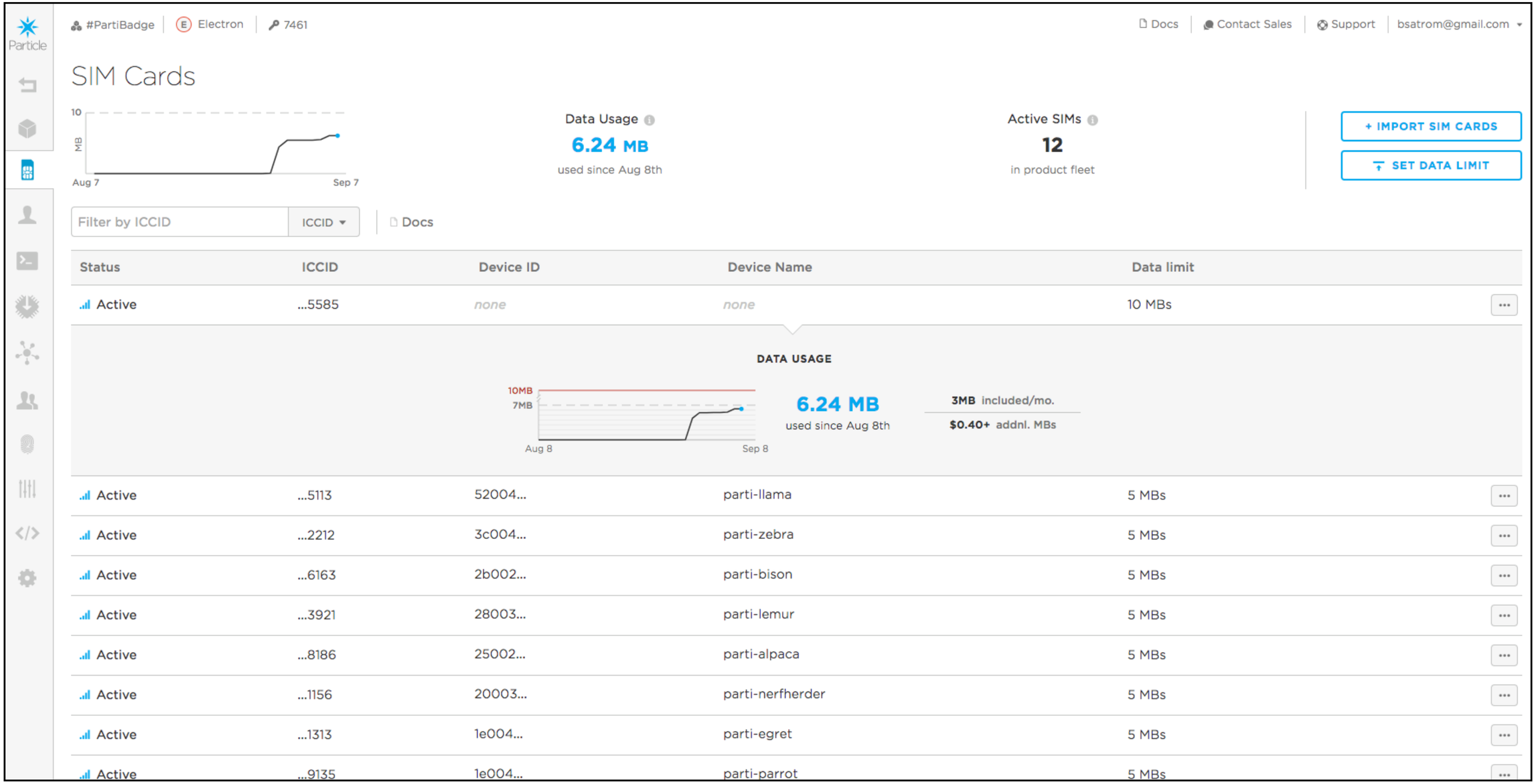
FUNCTIONS

f updateFName = 1

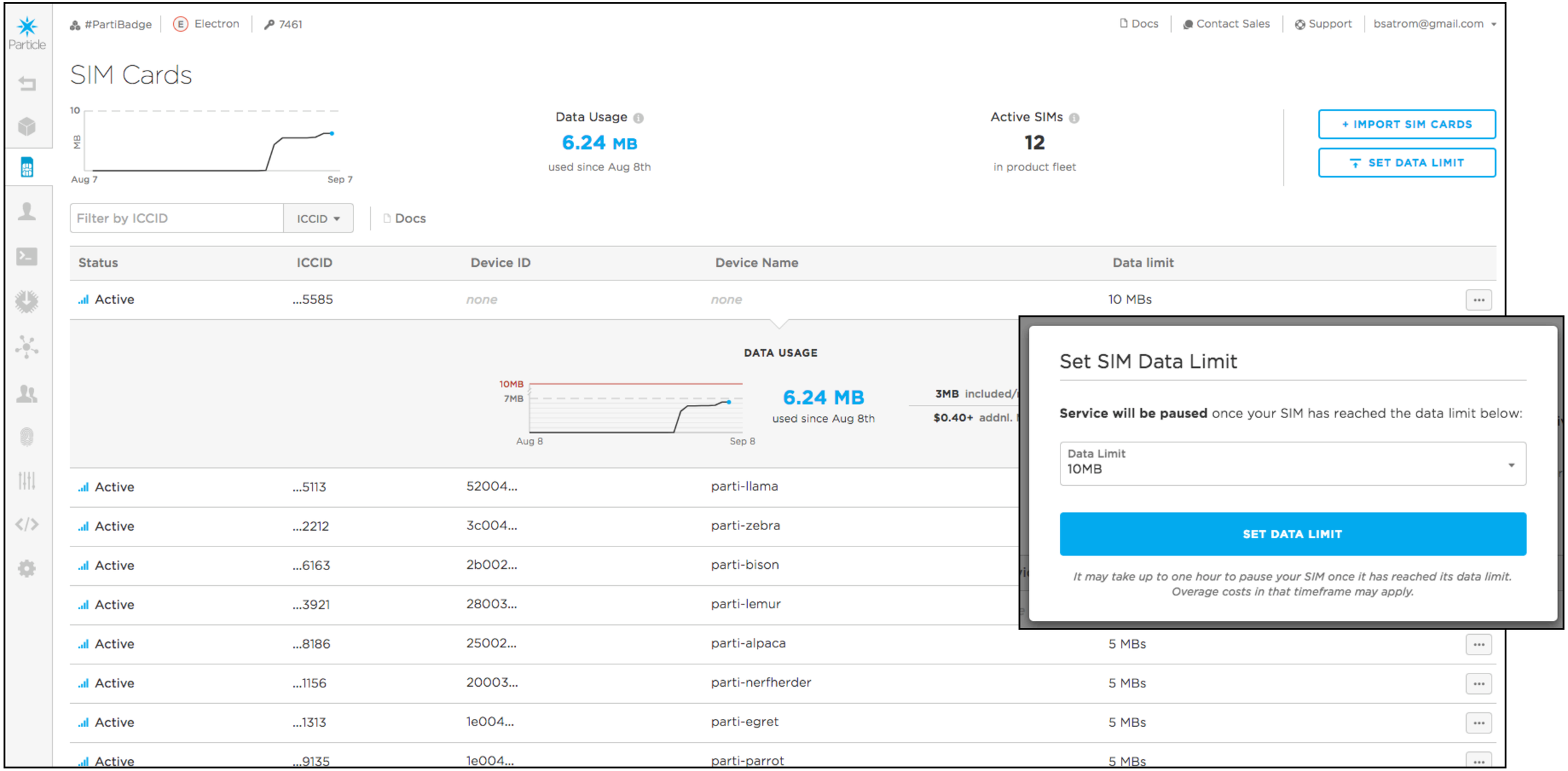
Brandon | CALL

f updateLName = 1

SIM MANAGEMENT



SIM MANAGEMENT



FLEET & FIRMWARE MANAGEMENT WITH PRODUCTS

Particle

#PartiBadge | Electron | 7461

Docs | Contact Sales | Support | brandon@particle.io

Devices

Filter by Device ID | Device ID

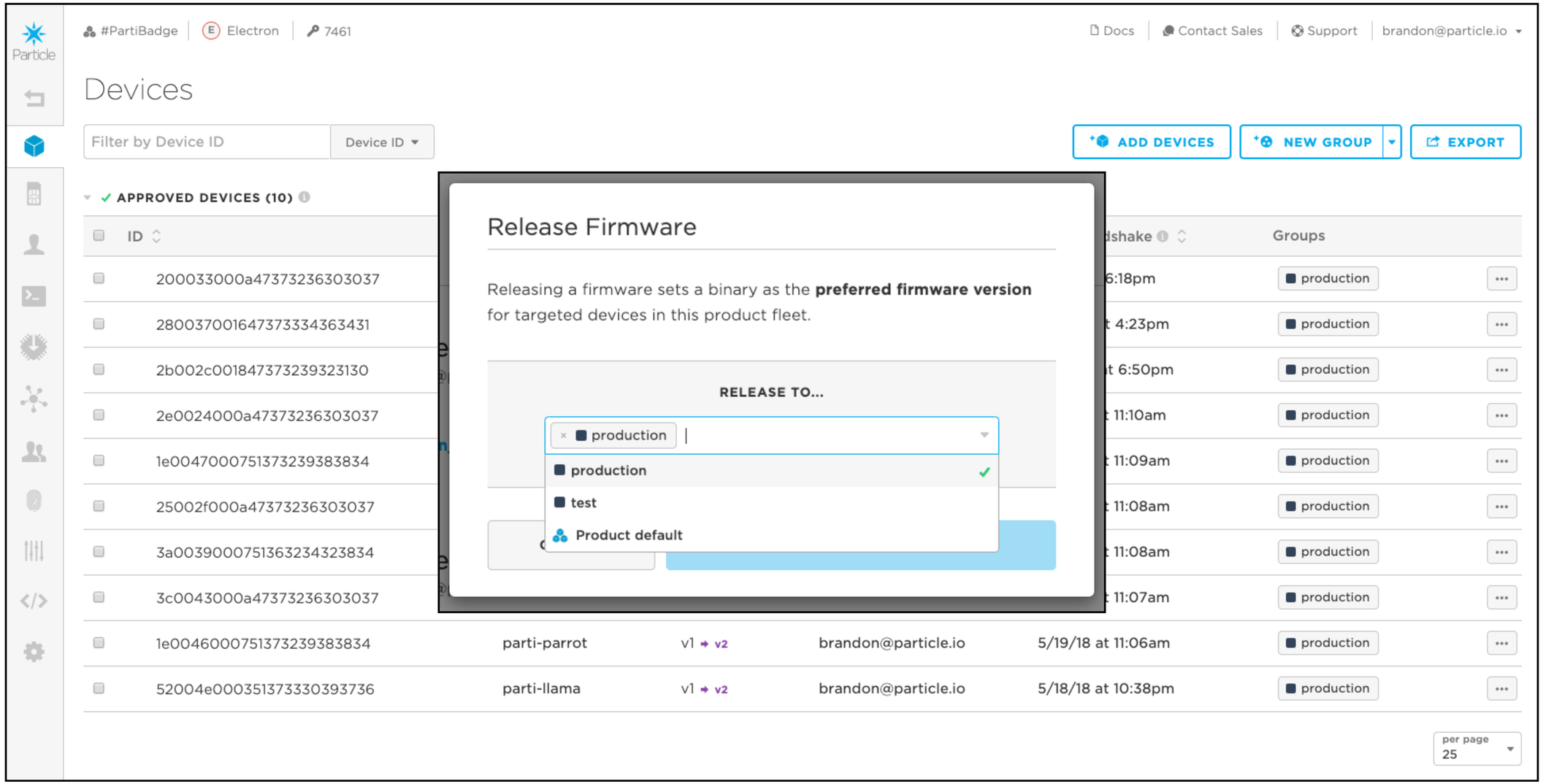
ADD DEVICES | NEW GROUP | EXPORT

APPROVED DEVICES (10)

ID	Name	Firmware	Owner	Last Handshake	Groups
200033000a47373236303037	parti-nerfherder	→ v2	brandon@particle.io	8/1/18 at 6:18pm	production
280037001647373334363431	parti-lemur	v2	brandon@particle.io	7/29/18 at 4:23pm	production
2b002c001847373239323130	parti-bison	v2 🔒	brandon@particle.io	5/20/18 at 6:50pm	production
2e0024000a47373236303037	parti-monkey	v1 → v2	brandon@particle.io	5/19/18 at 11:10am	production
1e0047000751373239383834	parti-egret	v1 → v2	brandon@particle.io	5/19/18 at 11:09am	production
25002f000a47373236303037	parti-alpaca	v1 → v2	brandon@particle.io	5/19/18 at 11:08am	production
3a0039000751363234323834	parti-badger	v1 → v2	brandon@particle.io	5/19/18 at 11:08am	production
3c0043000a47373236303037	parti-zebra	v1 → v2	brandon@particle.io	5/19/18 at 11:07am	production
1e0046000751373239383834	parti-parrot	v1 → v2	brandon@particle.io	5/19/18 at 11:06am	production
52004e000351373330393736	parti-llama	v1 → v2	brandon@particle.io	5/18/18 at 10:38pm	production

per page 25

FLEET & FIRMWARE MANAGEMENT WITH PRODUCTS



REMOTE DIAGNOSTICS

EVENT LOGS DIAGNOSTICS NEW

RUN TESTS

EVERYTHING LOOKS GOOD!

All diagnostic tests have passed. This device is healthy.

Device Vitals

Healthy

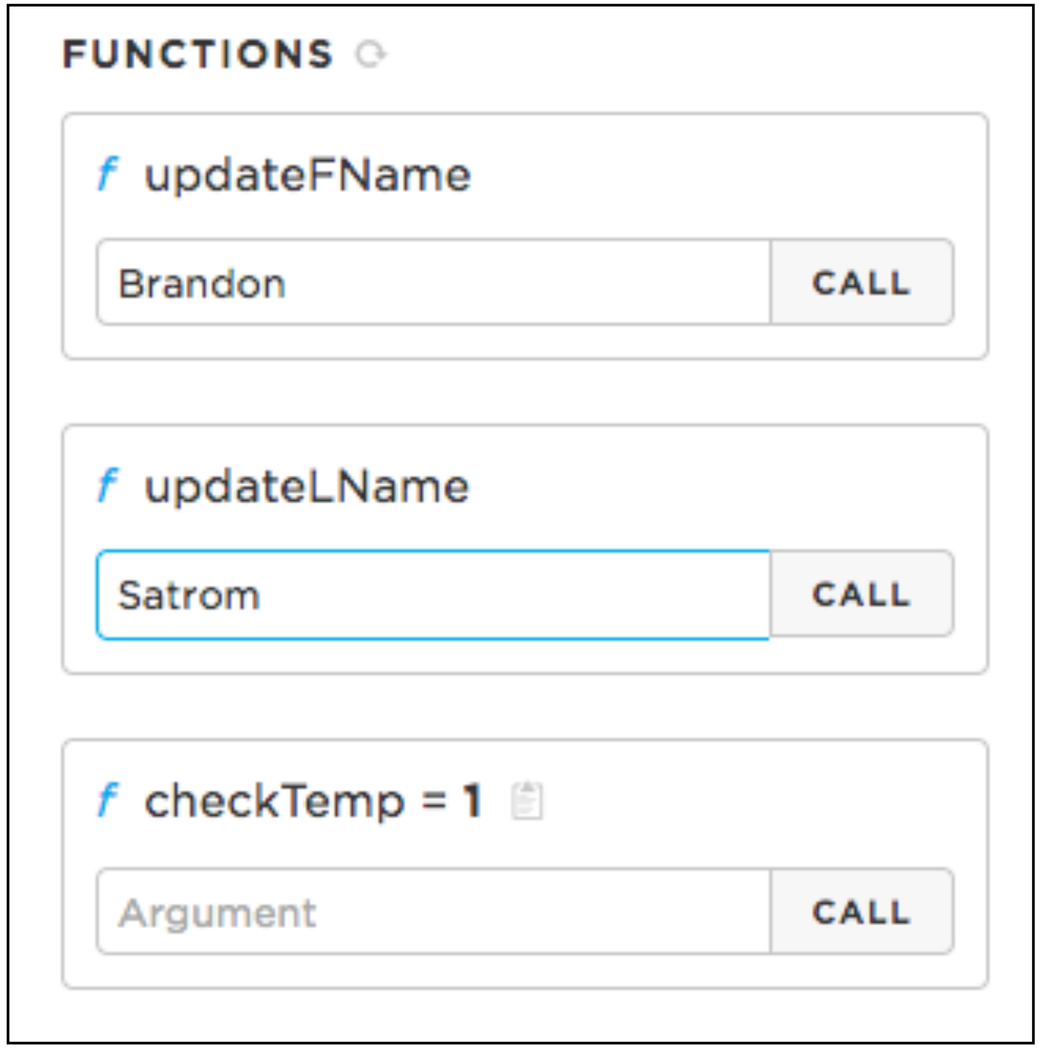
- **Strong** Wi-Fi signal
- **0** disconnect events
- **57ms** round-trip time
- **0** rate-limited publishes
- **33kB** of 81kB RAM used

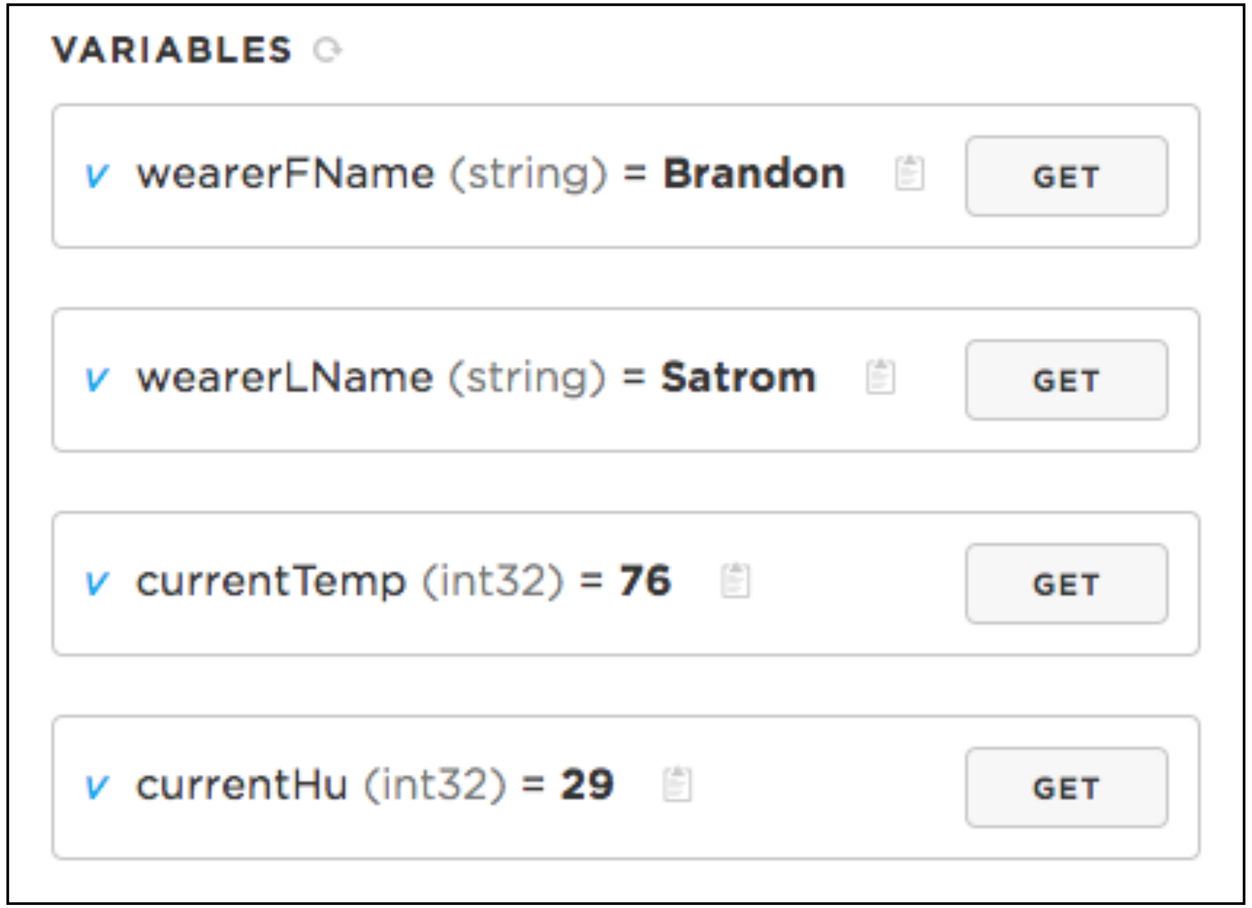
Device Cloud

Healthy

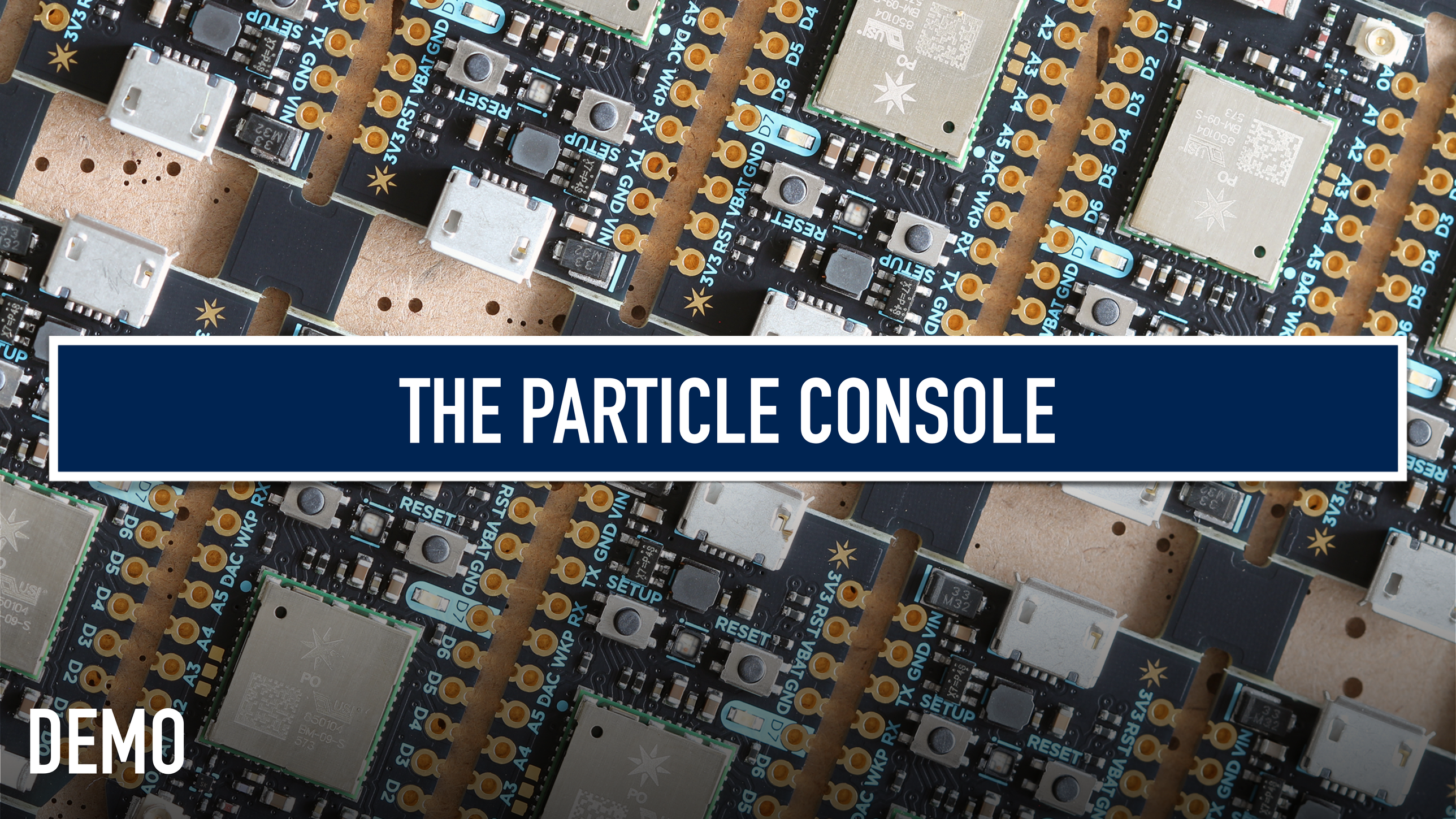
- API
- Device Service
- Webhooks

VIEWING CLOUD VARIABLES AND CALLING CLOUD FUNCTIONS





THE PARTICLE CONSOLE



DEMO

MANAGING DEVICES FROM THE CONSOLE

WORKING WITH PARTICLE PRIMITIVES

INTRODUCING PARTICLE GEN3 & MESH

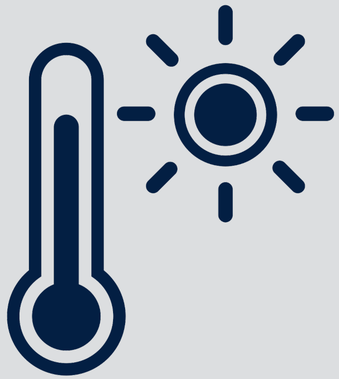
MESH PUBLISH & SUBSCRIBE

BLUETOOTH & NFC

PARTICLE CLOUD FUNCTIONS

Call a function, remotely

Particle.function()

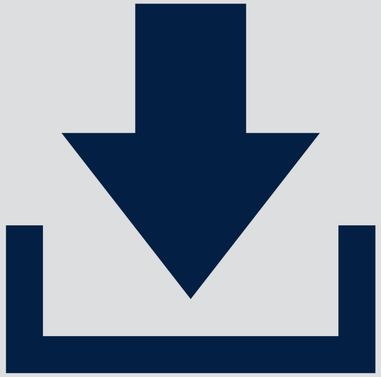


Fetch a variable, remotely

Particle.variable()

Send an event to the cloud

Particle.publish()



Listen for events

Particle.subscribe()

PARTICLE.VARIABLE()

What it does:

Expose a firmware variable to the cloud

Why it's cool:

- Can be fetched via the Device Cloud API
- Viewable from the Device Console

Usage notes:

- 20 variables max.
- 12 character limit per variable name

```
int analogvalue = 0;
double tempC = 0;

void setup()
{
  // variable name max length is 12 characters long
  Particle.variable("analogvalue", analogvalue);
  Particle.variable("temp", tempC);

  // Setup for Sensor on A0
  pinMode(A0, INPUT);
}

void loop()
{
  // Read the analog value of the sensor
  analogvalue = analogRead(A0);

  //Convert the reading into degrees Celsius
  tempC = (((analogvalue * 3.3)/4095) - 0.5) * 100;
  delay(200);
}
```

PARTICLE.VARIABLE()

What it does:

Expose a firmware variable to the cloud

Why it's cool:

- Can be fetched via the Device Cloud API
- Viewable from the Device Console

Usage notes:

- 20 variables max.
- 12 character limit per variable name

```
int analogvalue = 0;
double tempC = 0;

void setup()
{
```

```
# EXAMPLE REQUEST IN TERMINAL
# Device ID is 0123456789abcdef
# Your access token is 123412341234
curl "https://api.particle.io/v1/devices/0123456789abcdef/
analogvalue?access_token=123412341234"
curl "https://api.particle.io/v1/devices/0123456789abcdef/
temp?access_token=123412341234"

# In return you'll get something like this:
960
27.44322344322344
```

```
  //Convert the reading into degrees Celsius
  tempC = (((analogvalue * 3.3)/4095) - 0.5) * 100;
  delay(200);
}
```

PARTICLE.FUNCTION()

What it does:

Expose a firmware function to the cloud

Why it's cool:

- Can be called via the Device Cloud API
- Callable from the Device Console

Usage notes:

- 15 functions max.
- 12 character limit per function name

```
int togglePump(String command);

void setup()
{
  // register the cloud function
  Particle.function("togglePump", togglePump);
}

// this function automagically gets called upon a matching
POST request
int togglePump(String command)
{
  if (command == "on")
  {
    activateWaterPump();
  }
  else
  {
    deactivatePump();
  }

  return 1;
}
```

PARTICLE.FUNCTION()

What it does:

Expose a firmware function to the cloud

Why it's cool:

- Can be called via the Device Cloud API
- Callable from the Device Console

Usage notes:

- 15 functions max.
- 12 character limit per function name

```
int togglePump(String command);

void setup()
{
```

```
# API Call
# GET /v1/devices/{DEVICE_ID}/{VARIABLE}

# EXAMPLE REQUEST IN TERMINAL
# Device ID is 0123456789abcdef
# Your access token is 123412341234
curl "https://api.particle.io/v1/devices/0123456789abcdef/
analogvalue?access_token=123412341234"
curl "https://api.particle.io/v1/devices/0123456789abcdef/
temp?access_token=123412341234"

# In return you'll get something like this:
960
27.44322344322344
```

```
  }

  return 1;
}
```

PARTICLE.PUBLISH()

What it does:

Publish an event that will be forwarded to all registered listeners.

Why it's cool:

- Enables device-to-device communication
- Viewable from the Device Console

Usage notes:

- 63 characters max for event names
- Events are public by default, but can be marked as private.

```
double tempC = 0;

void setup()
{
  Particle.variable("temp", tempC);

  pinMode(A0, INPUT);
}

void loop()
{
  analogvalue = analogRead(A0);
  tempC = (((analogvalue * 3.3) / 4095) - 0.5) * 100;

  if (tempC > 120)
  {
    Particle.publish("temp/critical", tempC);
  }
  else if (tempC > 80)
  {
    Particle.publish("temp/warning", tempC);
  }
}
```

PARTICLE.PUBLISH()

What it does:

Publish an event that will be forwarded to all registered listeners.

Why it's cool:

- Enables device-to-device communication
- Viewable from the Device Console

Usage notes:

- 63 characters max for event names
- Events are public by default, but can be marked as private.

```
double tempC = 0;

void setup()
{
```

```
# API Call
# GET /v1/events/{EVENT_NAME}

# EXAMPLE REQUEST
curl -H "Authorization: Bearer {ACCESS_TOKEN_GOES_HERE}" \
https://api.particle.io/v1/events/temp/critical

# Will return a stream that echoes text when your event is
published
event: temp/critical
data:
{"data":"125","ttl":"60","published_at":"2018-05-28T19:20:34
.638Z",
  "deviceid":"0123456789abcdef"}
```

```
    Particle.publish("temp/warning", tempC);
  }
}
```

PARTICLE.SUBSCRIBE()

What it does:

Subscribe to events published by devices.

Why it's cool:

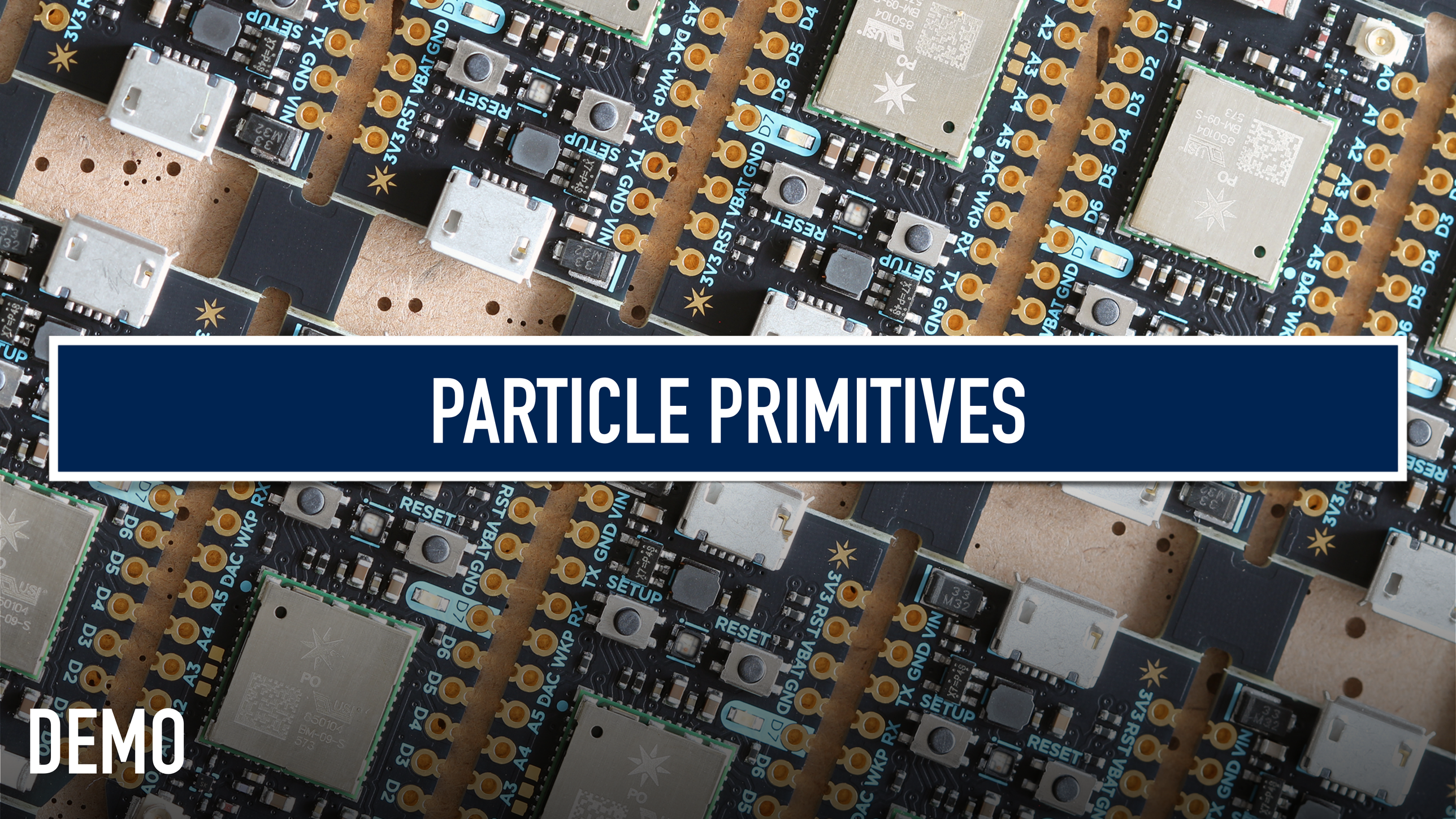
- Enables device-to-device communication
- Non-IoT devices can also trigger events

Usage notes:

- 4 subscribe handlers per device, max
- Subscriptions work like prefix filters, meaning you can capture multiple publish events via clever naming.

```
void setup()
{
  // Subscribes to temp/warning AND temp/critical
  Particle.subscribe("temp", handleTemp);
}

void handleTemp(const char *event, const char *data)
{
  double temp = extractTemp(data);

  if (temp > 120)
  {
    deactivatePump();
  }
  else if (temp > 80)
  {
    reducePumpSpeed();
  }
}
```

PARTICLE PRIMITIVES



DEMO

MANAGING DEVICES FROM THE CONSOLE

WORKING WITH PARTICLE PRIMITIVES

INTRODUCING PARTICLE GEN3 & MESH

MESH PUBLISH & SUBSCRIBE

BLUETOOTH & NFC

Argon

- Wi-Fi + BLE +Mesh
- Wi-Fi endpoint or mesh gateway
- Starts at $25

Boron

- LTE-M1 + BLE + Mesh
- Cellular endpoint or mesh gateway
- Starts at $49

Xenon

- BLE + Mesh
- Mesh endpoint
- Starts at $15

Mesh enabled, next generation

- Feather form factor
- OpenThread-based Mesh

Nordic nRF52840 SoC

- ARM Cortex-M4F 32-bit
- 1MB flash, 256KB RAM
- IEEE 802.15.4-2006: 250
- Bluetooth 5: 2 Mbps, 1 Mbps, 500 Kbps, 125 Kbps
- ARM TrustZone Cryptographic security module
- NFC-A tag

Argon

- Wi-Fi + BLE +Mesh
- Wi-Fi endpoint or mesh gateway
- Starts at $25

ESP32 Wi-Fi coprocessor

- On-board 4MB flash for ESP32
- 802.11 b/g/n support
- 802.11 n (2.4 GHz), up to 150 Mbps

Device Features

- On-board add'l 2MB SPI flash
- 20 mixed signal GPIO (6 x Analog, 8 x PWM), UART, I2C, SPI
- Integrated Li-Po charging and battery connector
- JTAG (SWD) Connector

Boron

- LTE-M1 + BLE + Mesh
- Cellular endpoint or mesh gateway
- Starts at $49

u-blox SARA R410 LTE Modem

- LTE CAT M1/ NB1 module with global hardware support (MVNO support for US only)
- 3GPP Release 13 LTE Cat M1

Device Features

- On-board add'l 2MB SPI flash
- 20 mixed signal GPIO (6 x Analog, 8 x PWM), UART, I2C, SPI
- Integrated Li-Po charging and battery connector
- JTAG (SWD) Connector

Xenon

- BLE + Mesh
- Mesh endpoint
- Starts at $15

Mesh networking with OpenThread

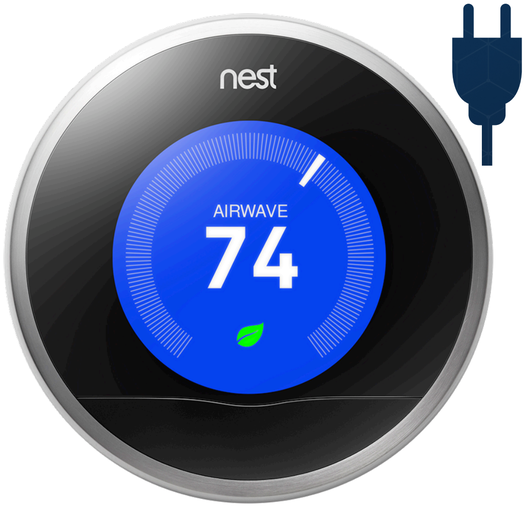
- IEEE 802.15.4-2006: 250
- Bluetooth 5: 2 Mbps, 1 Mbps, 500 Kbps, 125 Kbps

NEST: FROM THERMOSTATS TO SMOKE DETECTORS

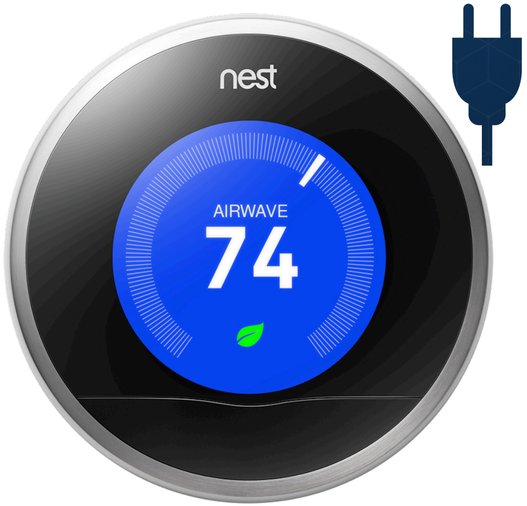
NEST: FROM THERMOSTATS TO SMOKE DETECTORS

NEST: FROM THERMOSTATS TO SMOKE DETECTORS

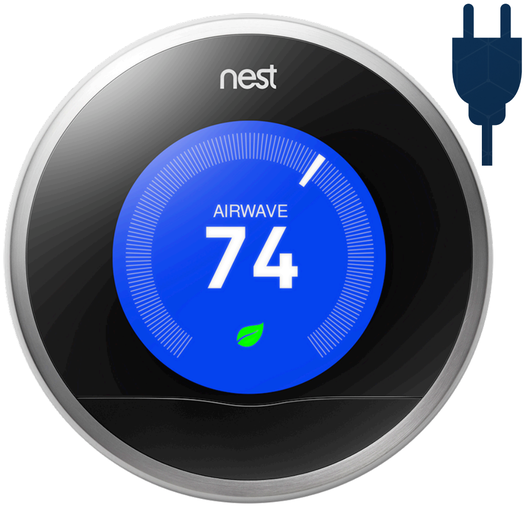
NEST: FROM THERMOSTATS TO SMOKE DETECTORS



NEST: FROM THERMOSTATS TO SMOKE DETECTORS



NEST: FROM THERMOSTATS TO SMOKE DETECTORS



THE THREAD GROUP & CONTRIBUTING OPENTHREAD

OPENTHREAD
released by Nest

THREAD GROUP

SIEMENS
Ingenuity for life

D-Link

NORDIC SEMICONDUCTOR

amazon
Lab126

Qualcomm

BOSCH

arm

LG
Life's Good

ANALOG DEVICES

NXP

TEXAS INSTRUMENTS

SAMSUNG

Particle

eero

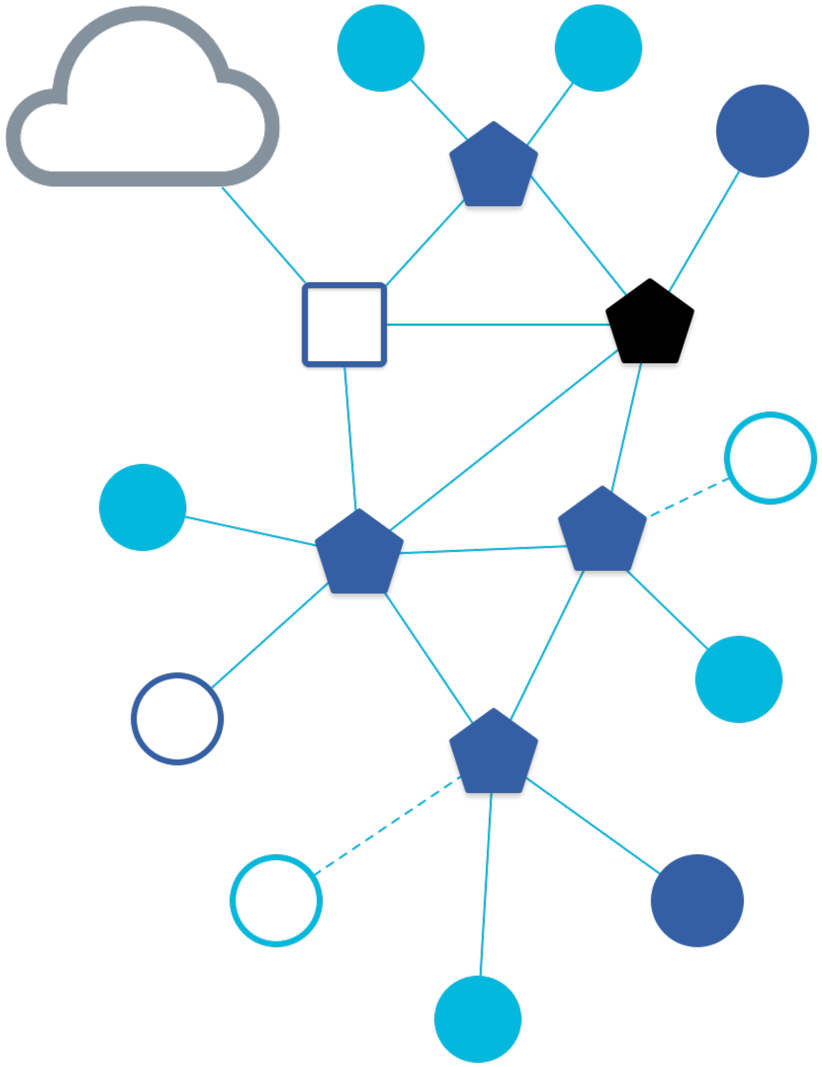
SILICON LABS

TDK

WHAT IS THREAD?

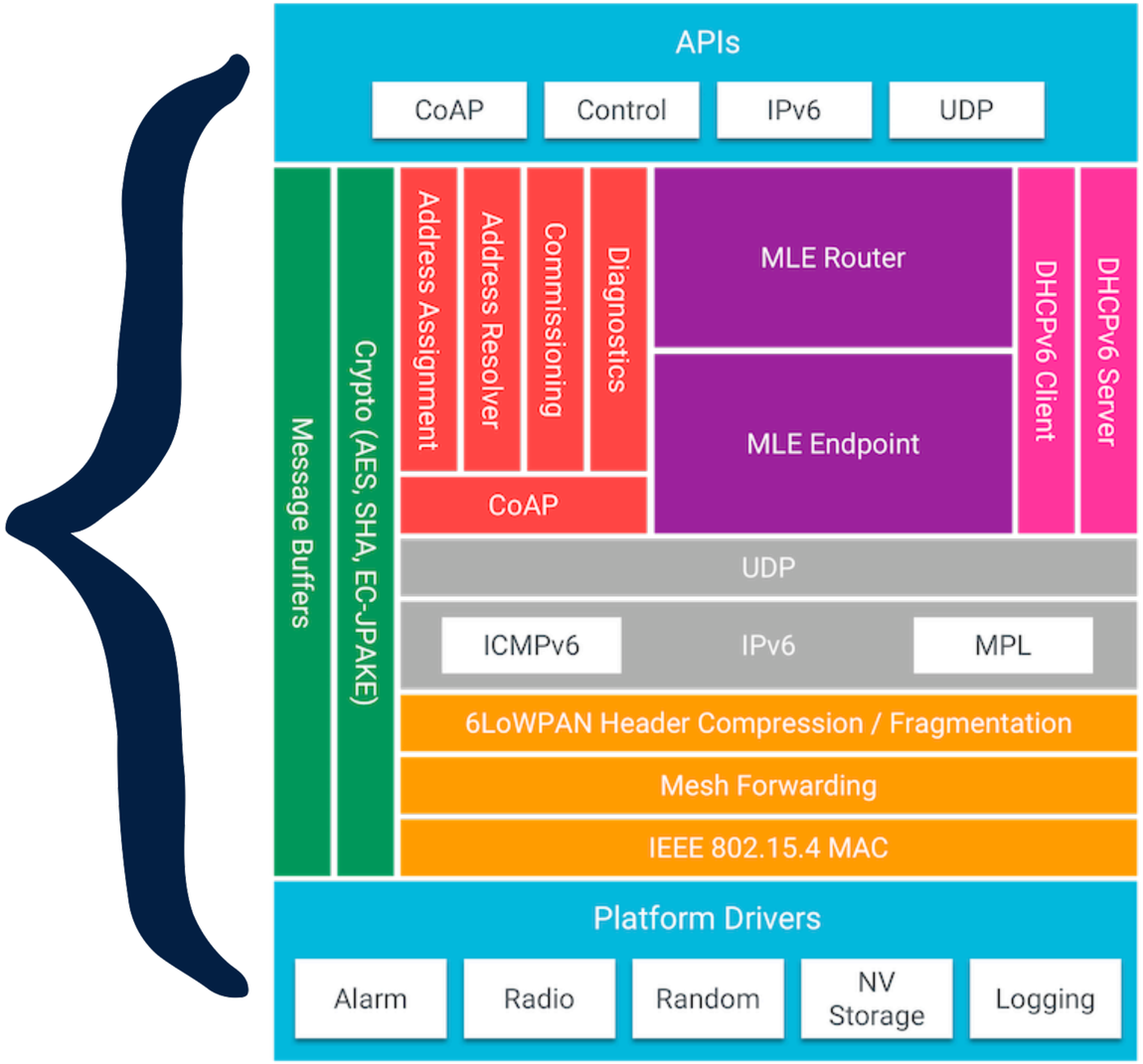
THREAD ***is a low-power networking protocol***

- IPv6-based mesh
- Wireless Personal Area Network
- No single point of failure
- Tailored to IoT Scenarios
- Can be used in concert with Wi-Fi, Cellular and Bluetooth

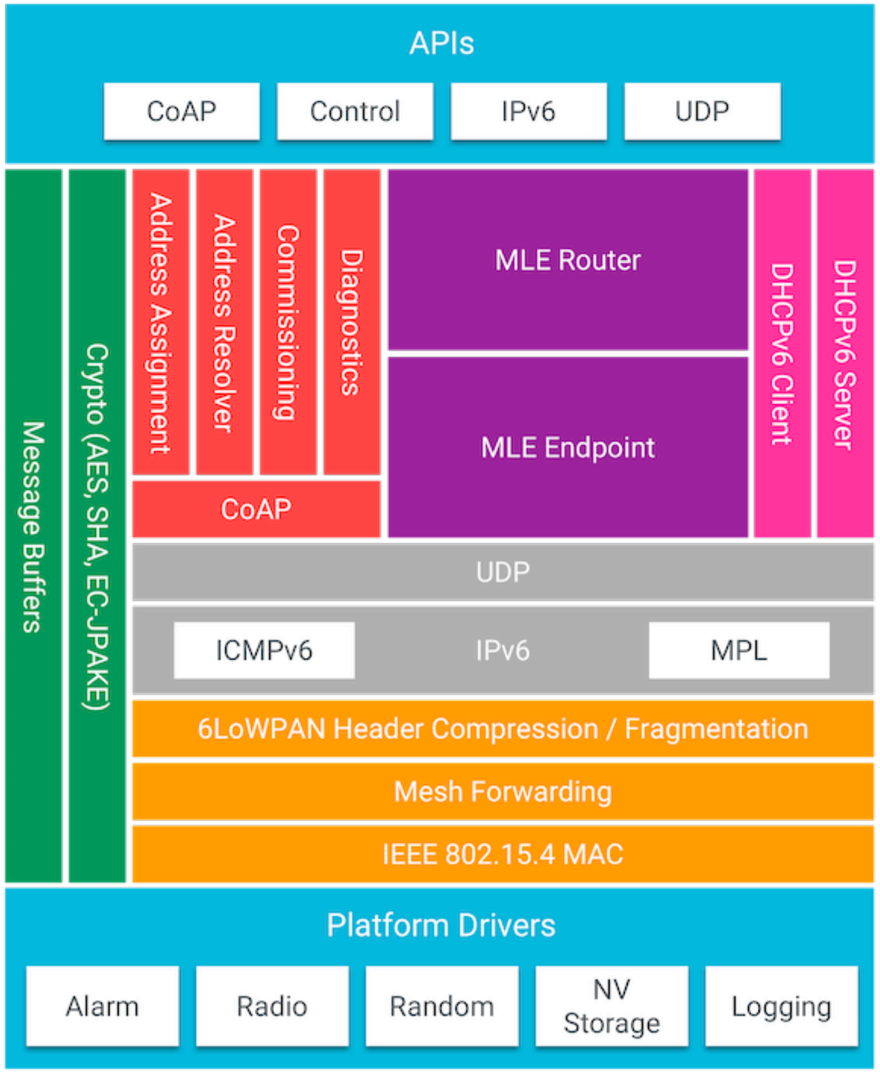


WHY PARTICLE MESH?

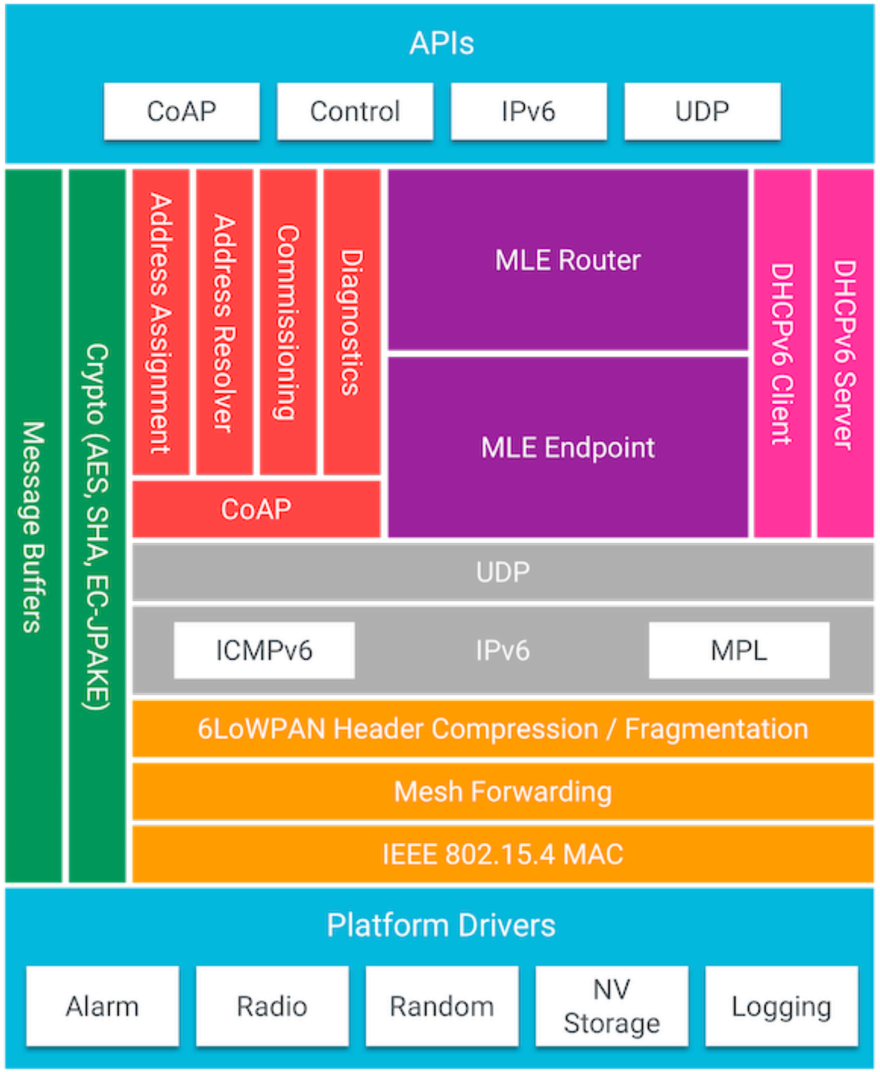
Everything you need to know to implement OpenThread

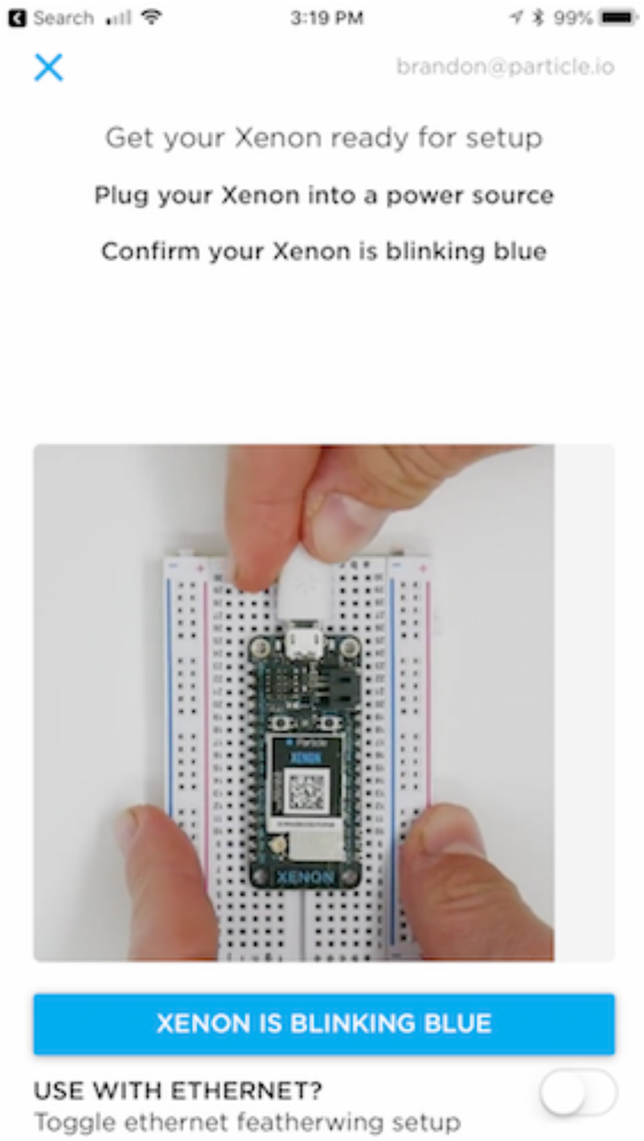


WHY PARTICLE MESH?

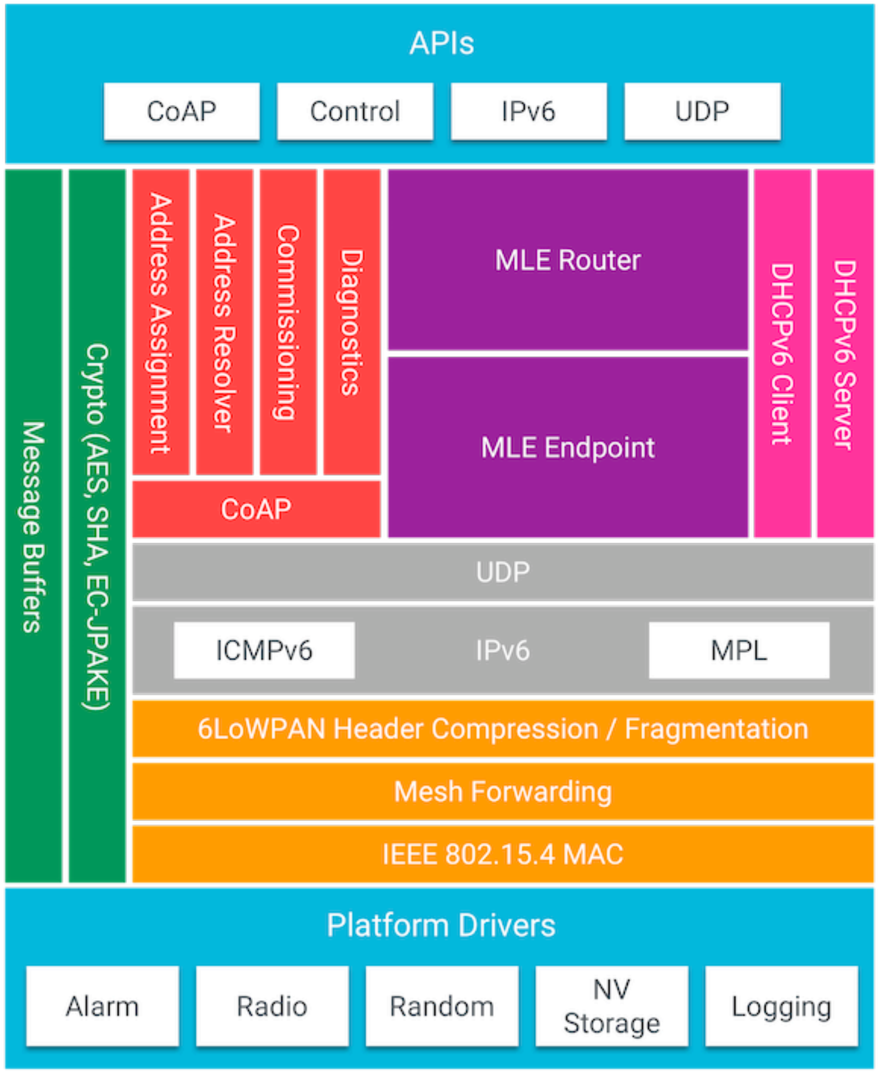


WHY PARTICLE MESH?





WHY PARTICLE MESH?




```
void pong(const char *event, const char *data)
{
  Serial.println("You got a message!");
}

void setup()
{
  Mesh.on();
  Mesh.connect();
}

void loop()
{
  Mesh.publish("hello-world", "I'm meshing !");
  Mesh.subscribe("ping", pong);
}
```

PARTICLE MESH != BLUETOOTH MESH

PARTICLE MESH != BLUETOOTH MESH

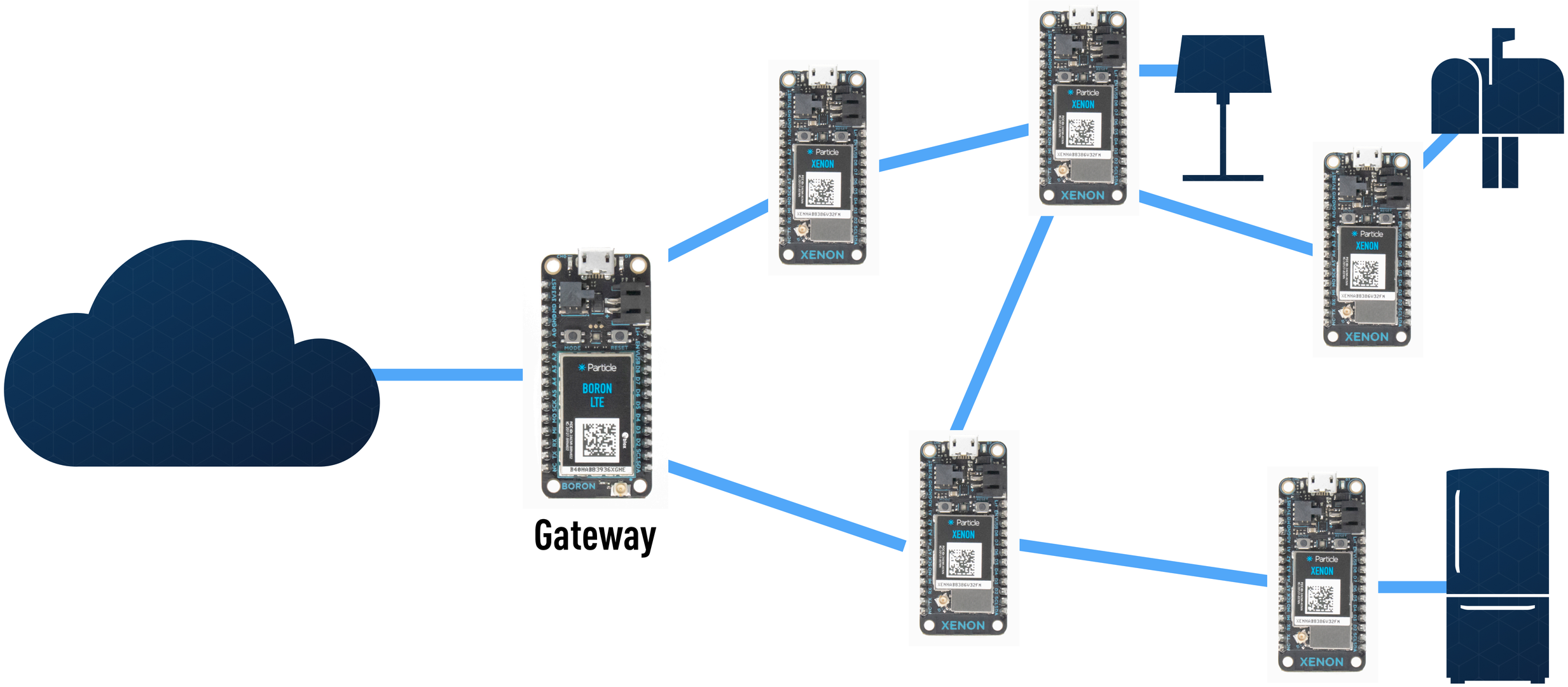
PARTICLE MESH != BLUETOOTH MESH

PARTICLE MESH != WI-FI MESH

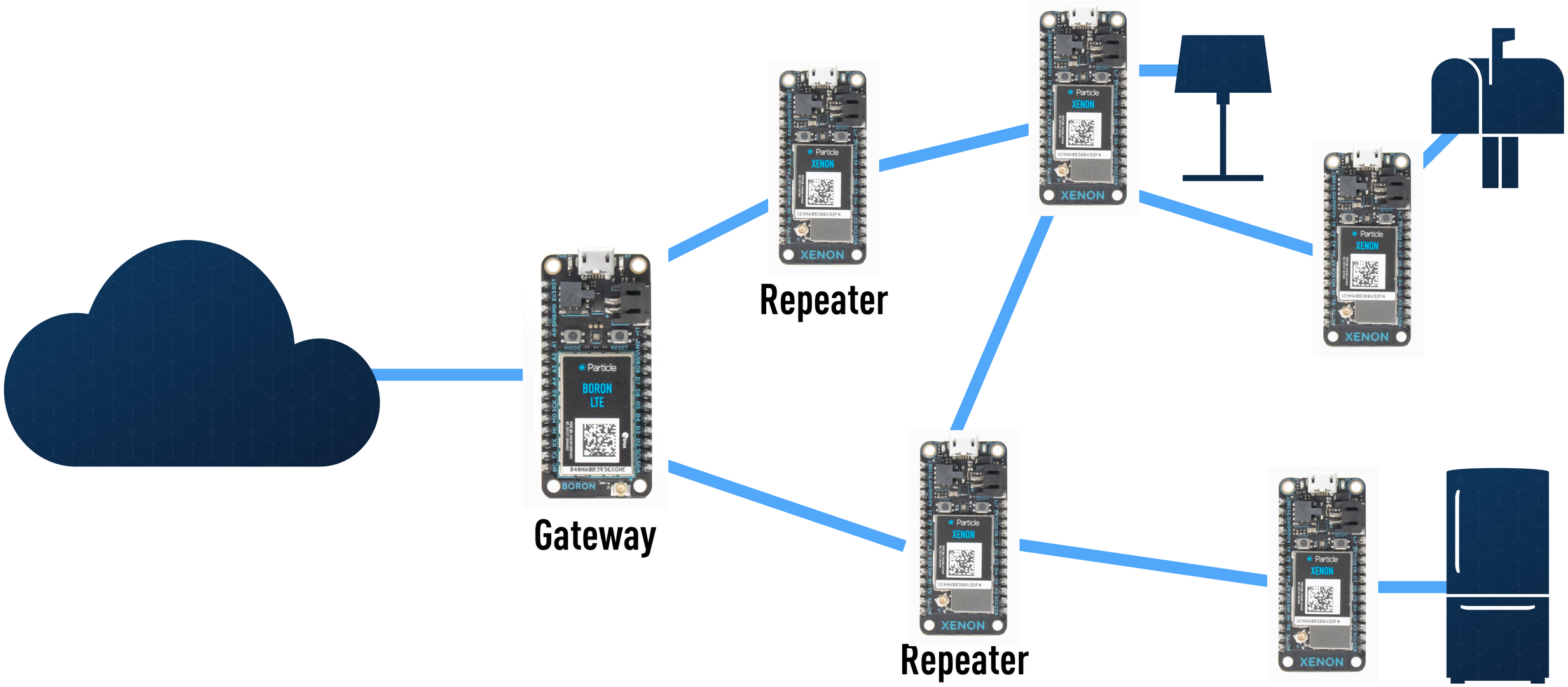
OPENTHREAD VS. ZIGBEE, ZWAVE & BT MESH

	Z-WAVE	zigbee	Bluetooth MESH	THREAD	Wi-Fi Mesh
Operating range	100 ft	35 ft	30 ft	100 ft	Varies
Max # of devices	232	65k	~32k	300+	Varies
Data rate	9.6-100 Kb	40-250 Kb	1-3 Mb	250 Kb	Varies
Cloud Connectivity	Gateway	Gateway	Smartphone	Gateway	Router
IP-Based Networking	No	No	No	Yes	Yes
Open Standard?	No	Yes	Yes	Yes	No

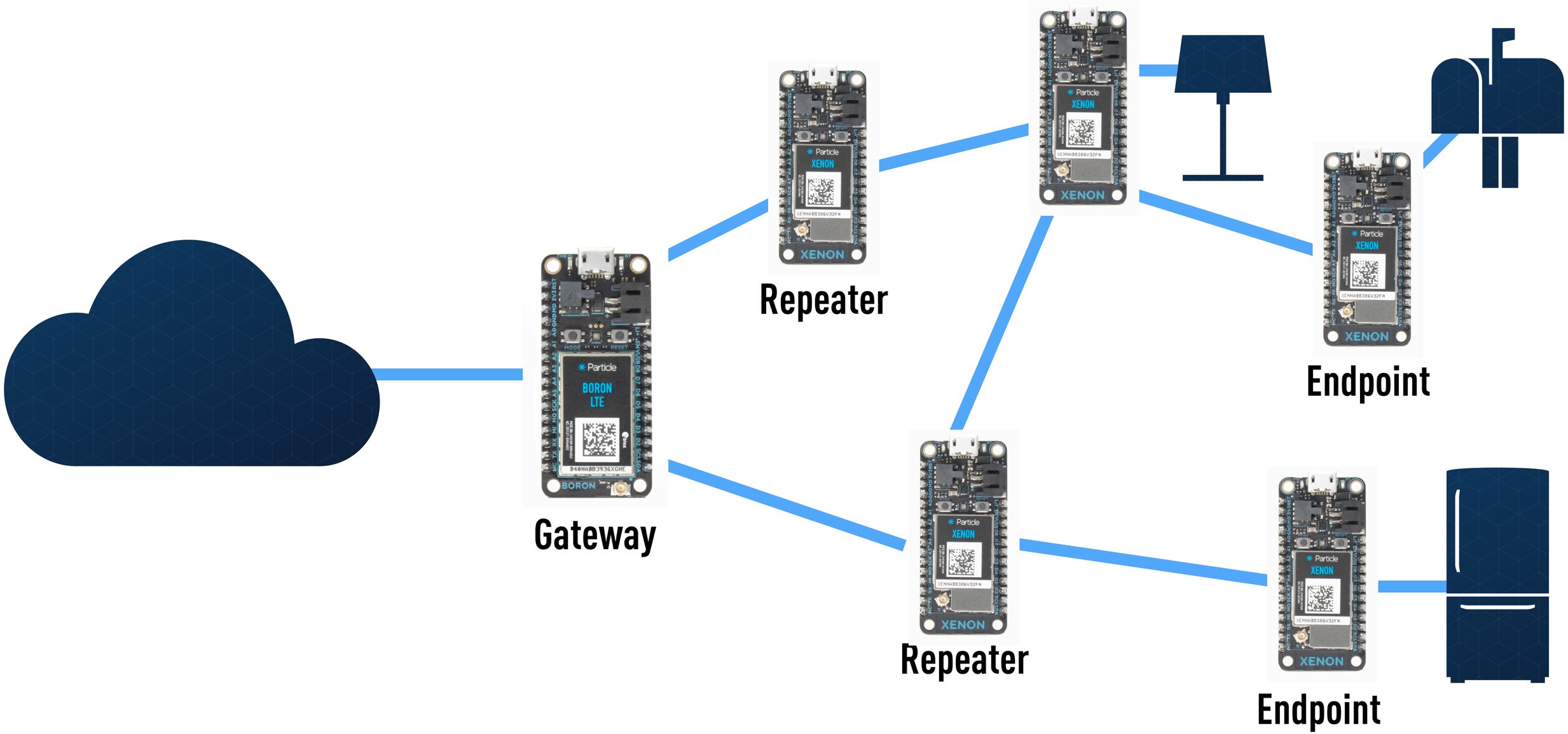
MESH DEVICE ROLES



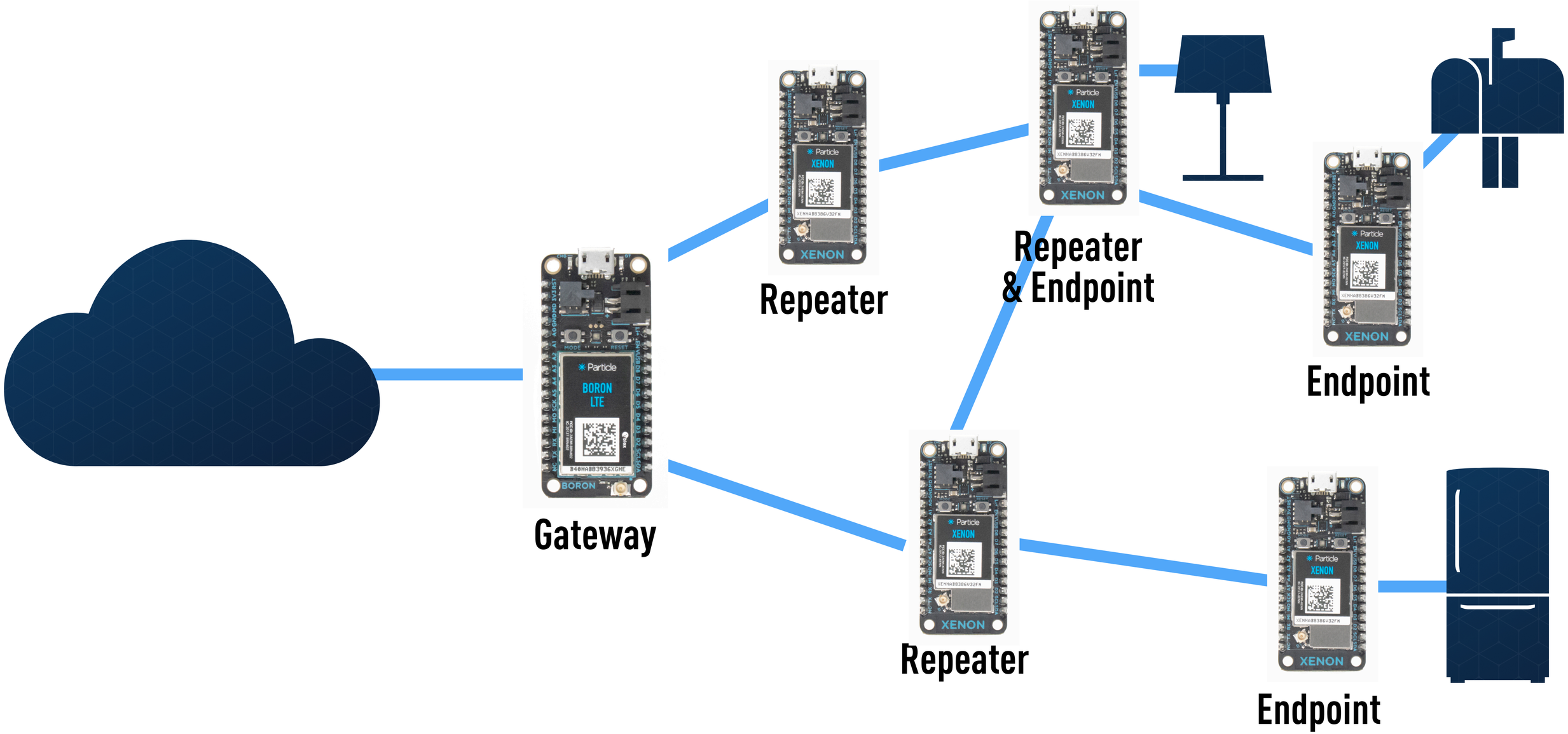
MESH DEVICE ROLES

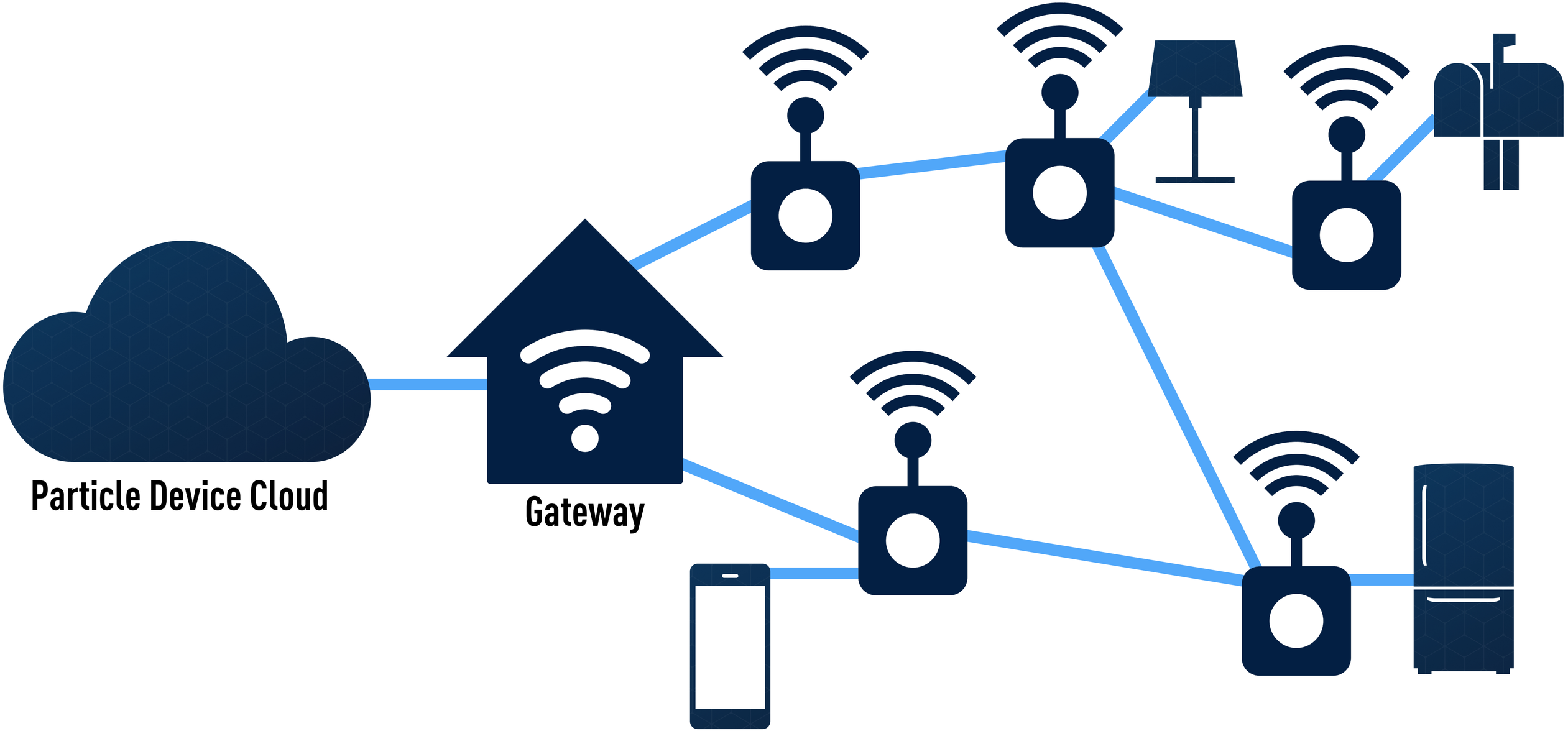


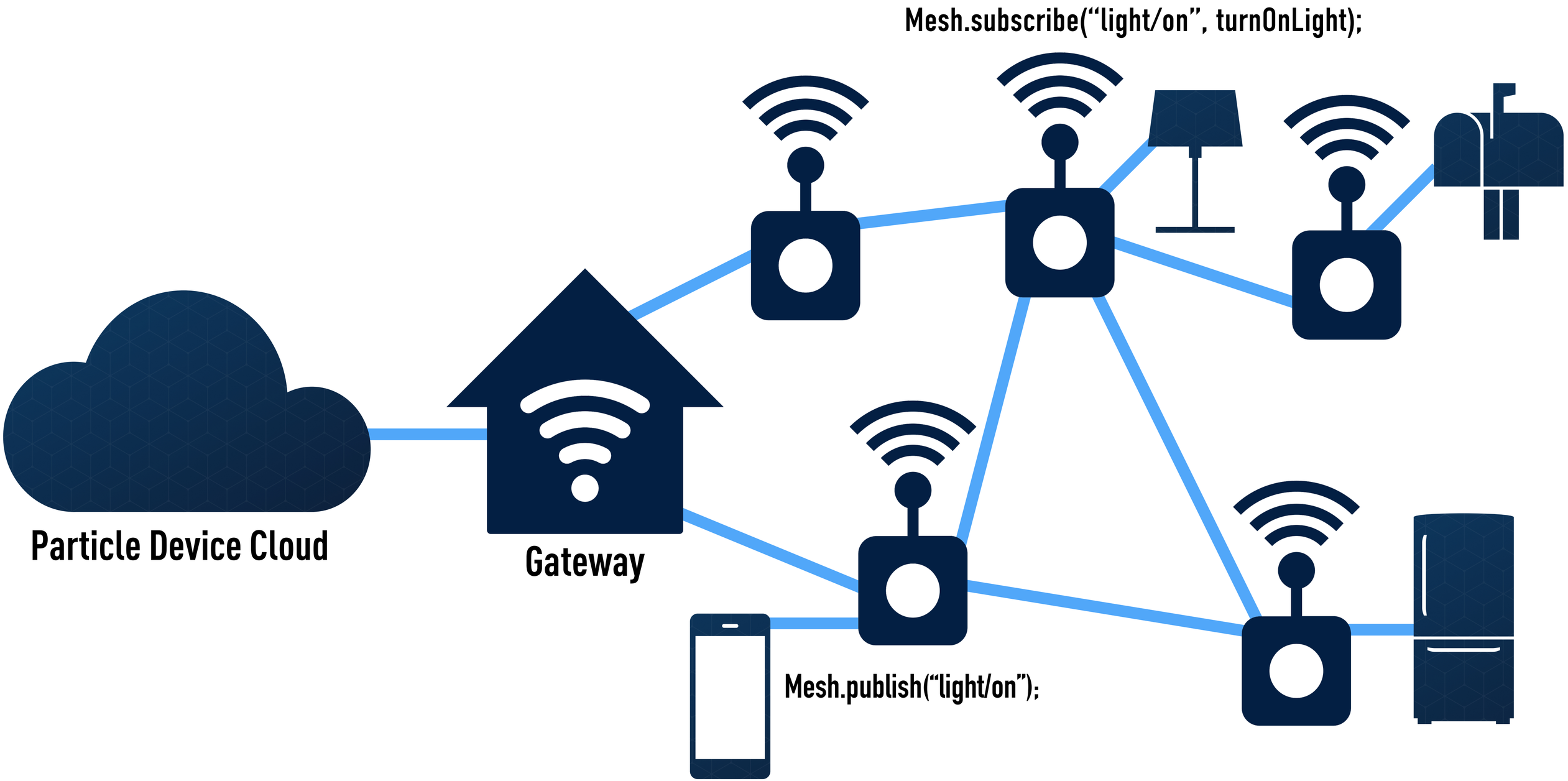
MESH DEVICE ROLES

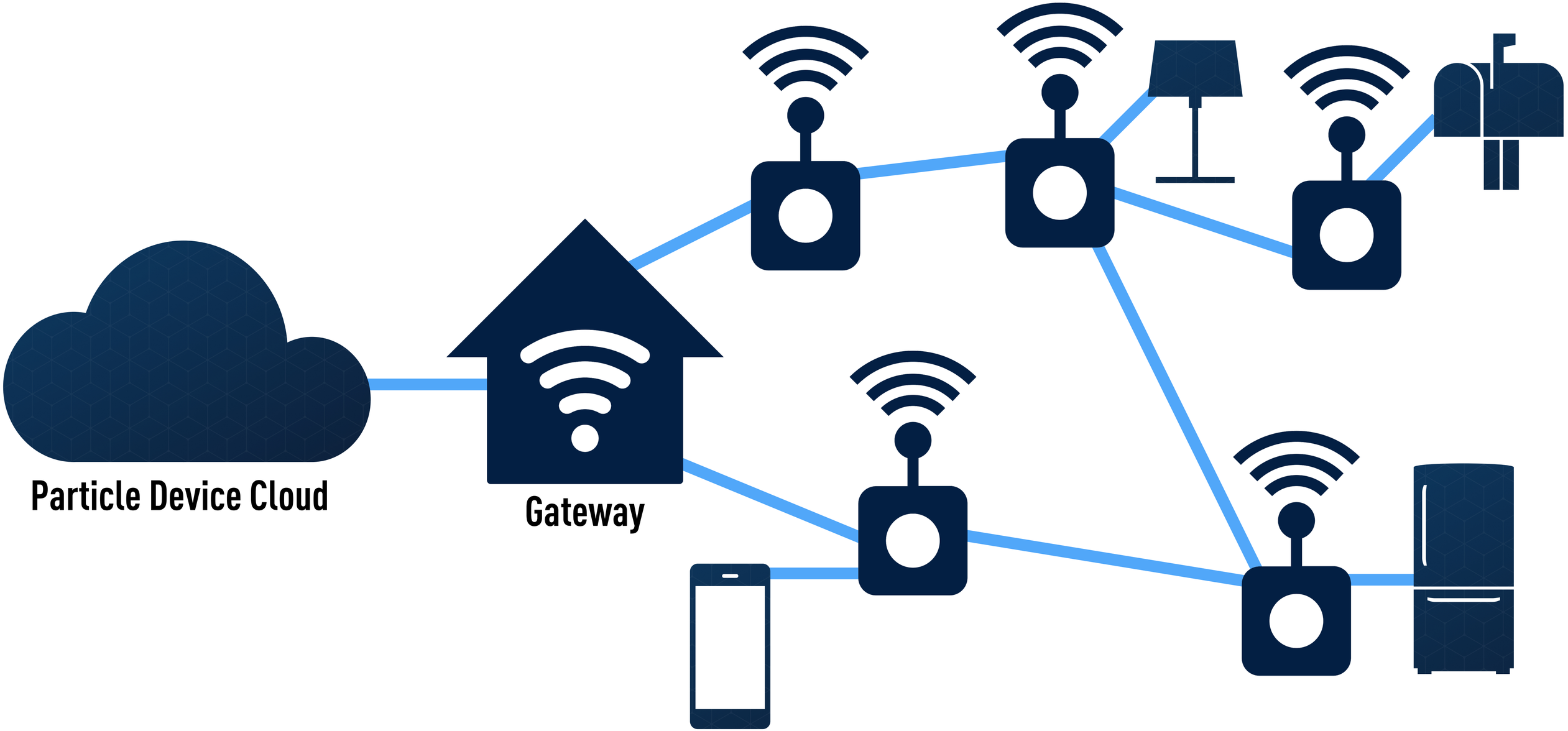


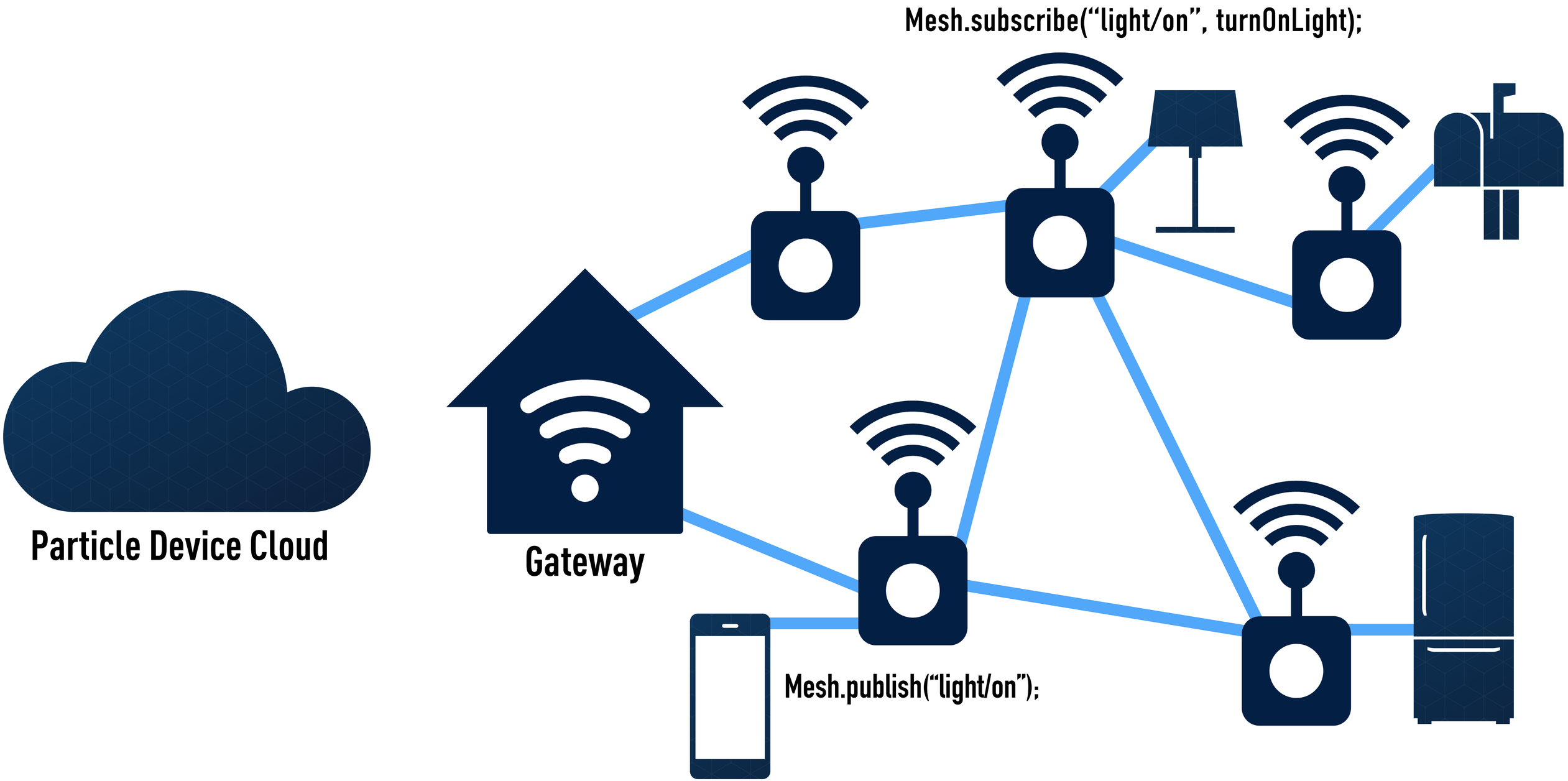
MESH DEVICE ROLES




Particle Device Cloud
Gateway


Mesh.subscribe("light/on", turnOnLight);
Particle Device Cloud
Gateway
Mesh.publish("light/on");

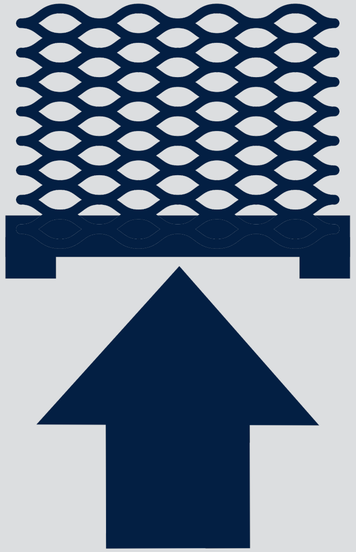

Particle Device Cloud
Gateway


Mesh.subscribe("light/on", turnOnLight);
Particle Device Cloud
Gateway
Mesh.publish("light/on");

PARTICLE MESH FUNCTIONS

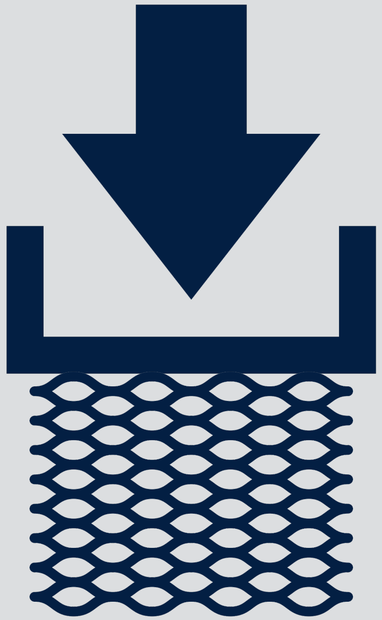
Broadcast an event to all devices in a Mesh network

Mesh.publish()



Listen for events published to the Mesh network

Mesh.subscribe()



MESH.PUBLISH()

What it does:

Publish an event that will be forwarded to all registered listeners on the local Particle mesh network.

Why it's cool:

- Enables mesh network communication
- Works even when the network isn't connected to the cloud

Usage notes:

- 63 characters max for event names

```
double tempC = 0;

void setup()
{
  Particle.variable("temp", tempC);

  pinMode(A0, INPUT);
}

void loop()
{
  analogvalue = analogRead(A0);
  tempC = (((analogvalue * 3.3) / 4095) - 0.5) * 100;

  if (tempC > 120)
  {
    Mesh.publish("temp/critical", tempC);
  }
  else if (tempC > 80)
  {
    Mesh.publish("temp/warning", tempC);
  }
}
```

MESH.SUBSCRIBE()

What it does:

Subscribe to events published by devices on the local mesh network.

Why it's cool:

- Enables mesh network communication
- Works even when the network isn't connected to the cloud

Usage notes:

- Subscriptions work like prefix filters, meaning you can capture multiple publish events via clever naming.

```
void setup()
{
  // Subscribes to temp/warning AND temp/critical
  Mesh.subscribe("temp", handleTemp);
}

void handleTemp(const char *event, const char *data)
{
  double temp = extractTemp(data);

  if (temp > 120)
  {
    deactivatePump();
  }
  else if (temp > 80)
  {
    reducePumpSpeed();
  }
}
```

LOCAL MESH PUB/SUB VS. PARTICLE CLOUD PUB/SUB

Mesh Pub/Sub is for local messages

Use Mesh Pub/Sub When:

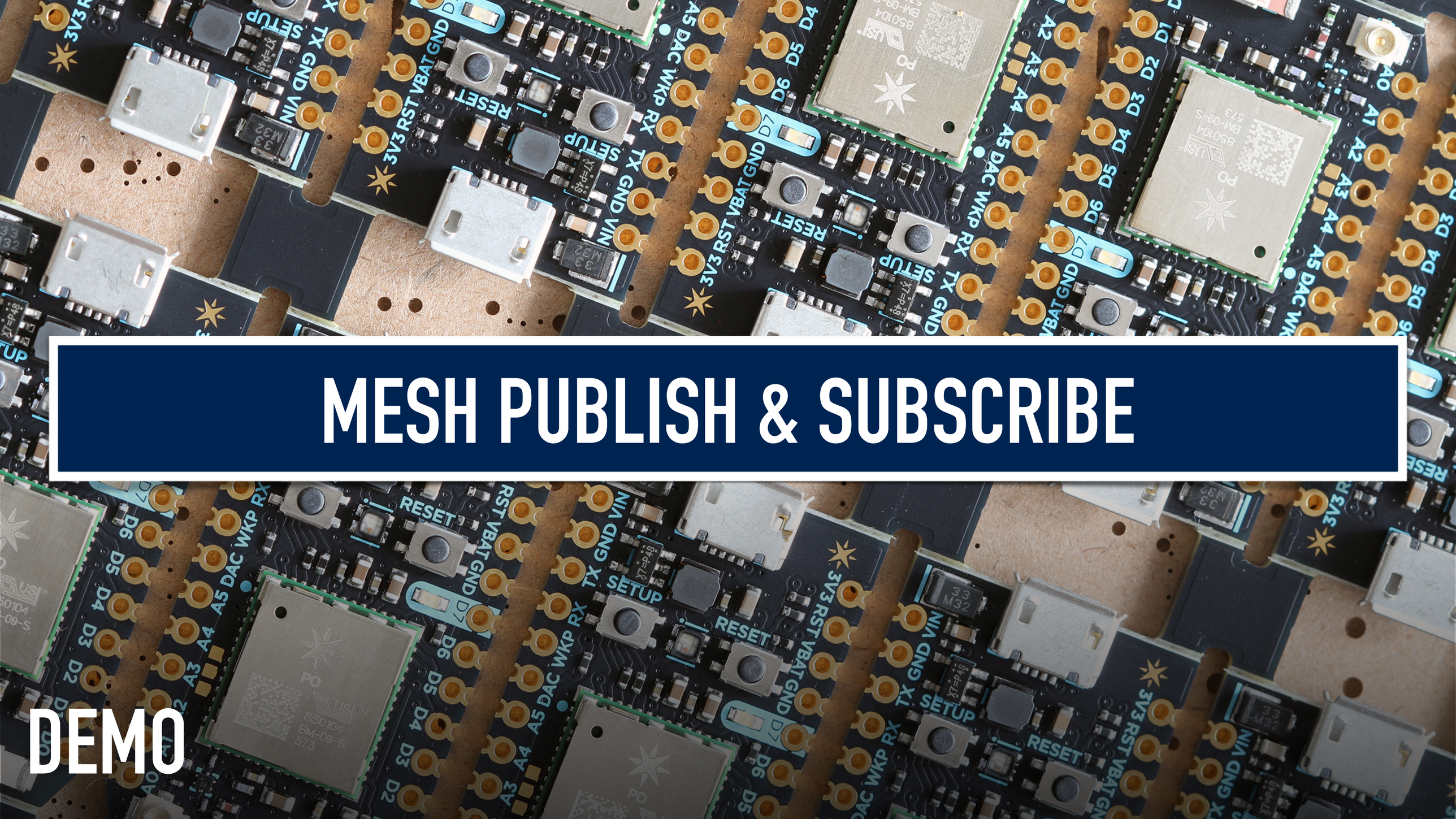
- You need to communicate between devices *only* on a mesh
- You need messages to be sent as fast as possible
- You need to communicate between devices when a connection to the cloud is unavailable.
- It's ok that not *every* message is delivered.

Particle Pub/Sub is for everything else

Use Particle Pub/Sub When:

- You need to communicate between mesh networks or with devices not on a mesh network
- You're publishing events to webhooks or cloud integrations (Azure, Google Cloud, etc.)
- You need some QOS in message delivery (retry attempts, etc.)

MESH PUBLISH & SUBSCRIBE



MANAGING DEVICES FROM THE CONSOLE

WORKING WITH PARTICLE PRIMITIVES

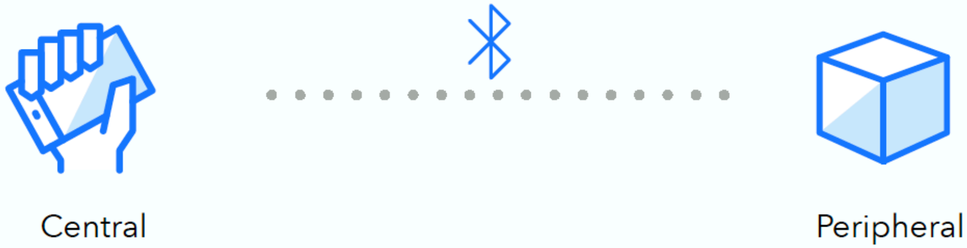
INTRODUCING PARTICLE GEN3 & MESH

MESH PUBLISH & SUBSCRIBE

BLUETOOTH & NFC

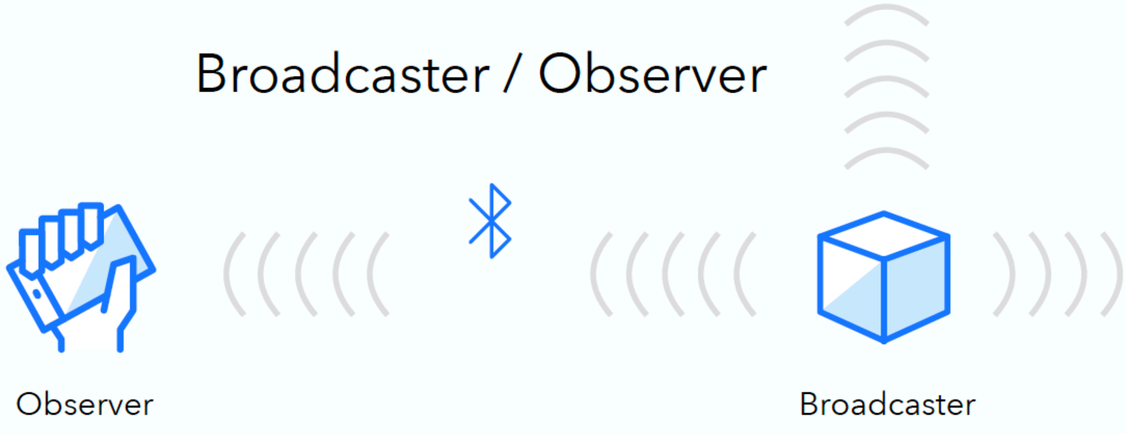
TWO PRIMARY MODES OF BLUETOOTH INTERACTION

Central / Peripheral



- Devices complete a **pairing process** and are securely connected to one another to transmit/receive messages
- **Bidirectional communications** – both the central device and peripheral device can send messages
- Central device is usually connected to some sort of display and is conveying information to the user
  - Laptop
  - Phone
  - LCD
- *Examples* – scooters, BT speakers, wearables

Broadcaster / Observer



- Devices **never pair** with one another
- **Unidirectional communications** – the broadcaster talks to the oberserver
- Broadcaster "publishes" messages on a particular channel and observer can receive them if it is listening
- There can be multiple observers of the same broadcast messaging
- Less common use case for IoT and consumer applications
- *Examples* – smart retail, smart city

EXAMPLE: BROADCASTER & OBSERVER

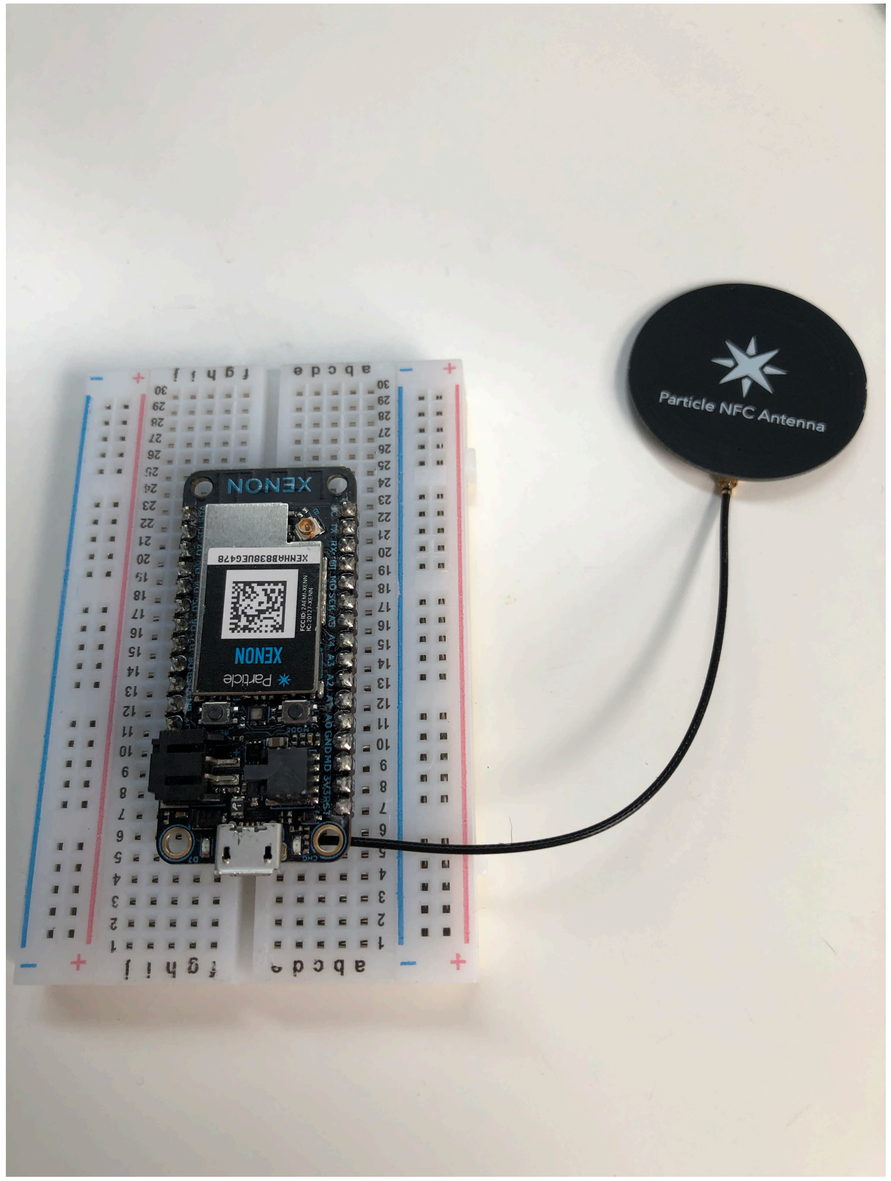
Broadcaster advertises battery voltage…

```
uint8_t buf[BLE_MAX_ADV_DATA_LEN];
size_t offset = 0;

// Company ID (0×ffff internal use/testing)
buf[offset++] = 0×ff;
buf[offset++] = 0×ff;

// Internal packet type.
buf[offset++] = 0×55;

memcpy(&buf[offset], &battVoltage, 4);
offset += 4;

BleAdvertisingData advData;
advData.appendCustomData(buf, offset);


BLE.setAdvertisingInterval(130);
BLE.advertise(&advData);
```

…which the observer can read.

```
const size_t SCAN_RESULT_MAX = 30;
BleScanResult scanResults[SCAN_RESULT_MAX];

BLE.setScanTimeout(50);
int count = BLE.scan(scanResults, SCAN_RESULT_MAX);

for (int i = 0; i < count; i++)
{
  uint8_t buf[BLE_MAX_ADV_DATA_LEN];
  size_t len;

  len = scanResults[i].advertisingData.get(
    BleAdvertisingDataType::MANUFACTURER_SPECIFIC_DATA, buf,
    BLE_MAX_ADV_DATA_LEN);
  if (len == 7)
  {
    if (buf[0] == 0×ff && buf[1] == 0×ff && buf[2] == 0×55)
    {
      float voltage;
      memcpy(&voltage, &buf[3], 4);

      Log.info("Voltage: %f", voltage);
    }
  }
}
```

NEAR FIELD COMMUNICATION (NFC)



NFC = for sending small amounts of data to mobile apps close by (< 3 inches)

» All Gen 3 devices can *emulate* an NFC tags (Device OS 1.3.0+ required)

```
NFC.on();

NFC.setText("Battery voltage: " +
  String(battVoltage, 2) + "%", "en");
NFC.update();
```

NEAR FIELD COMMUNICATION (NFC)

NFC = for sending small amounts of data to mobile apps close by (< 3 inches)

» All Gen 3 devices can *emulate* an NFC tags (Device OS 1.3.0+ required)

```
NFC.on();

NFC.setText("Battery voltage: " +
  String(battVoltage, 2) + "%", "en");
NFC.update();
```

BLE AND NFC: WHEN SHOULD I USE THEM?

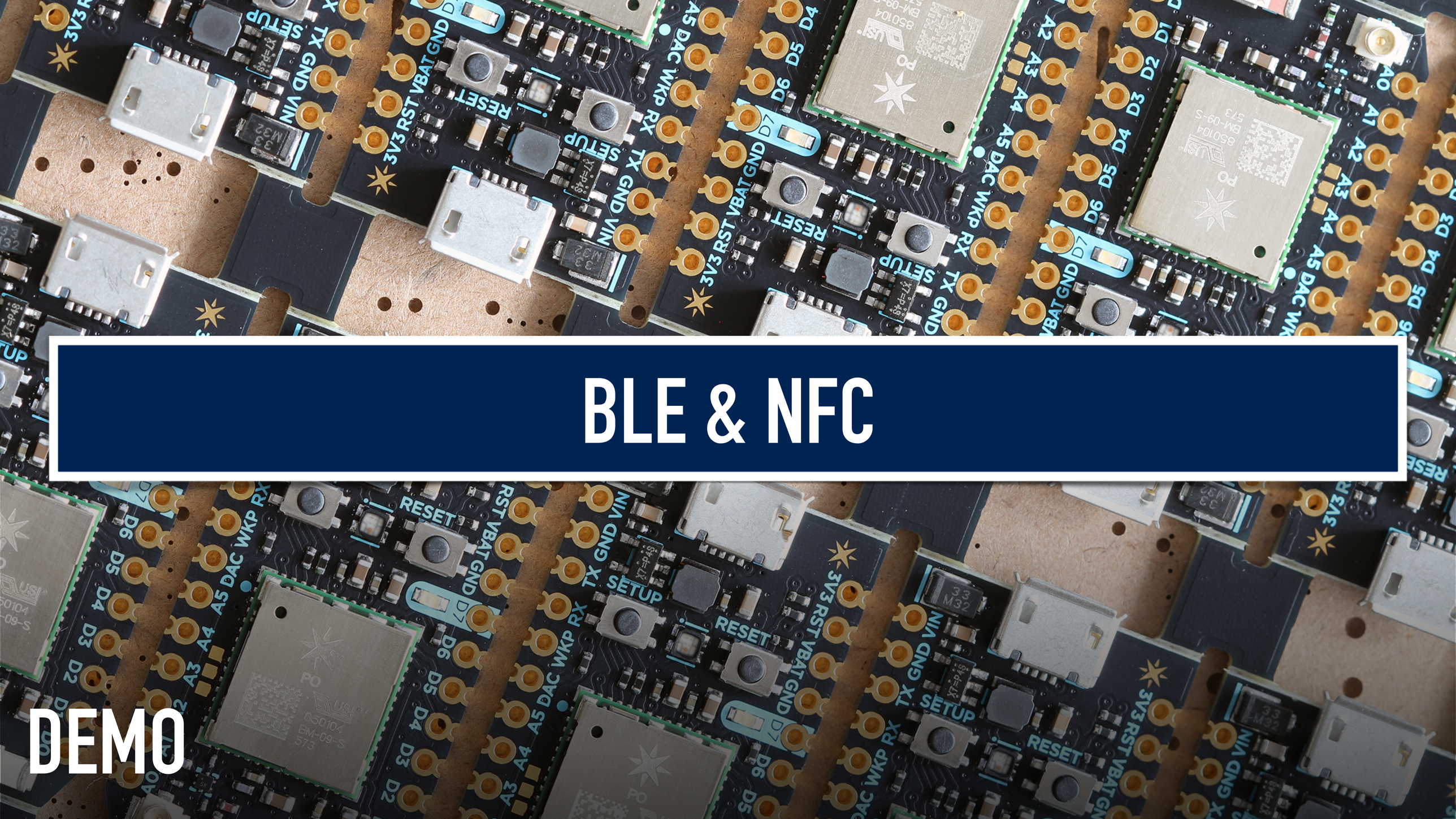
Use BLE When:

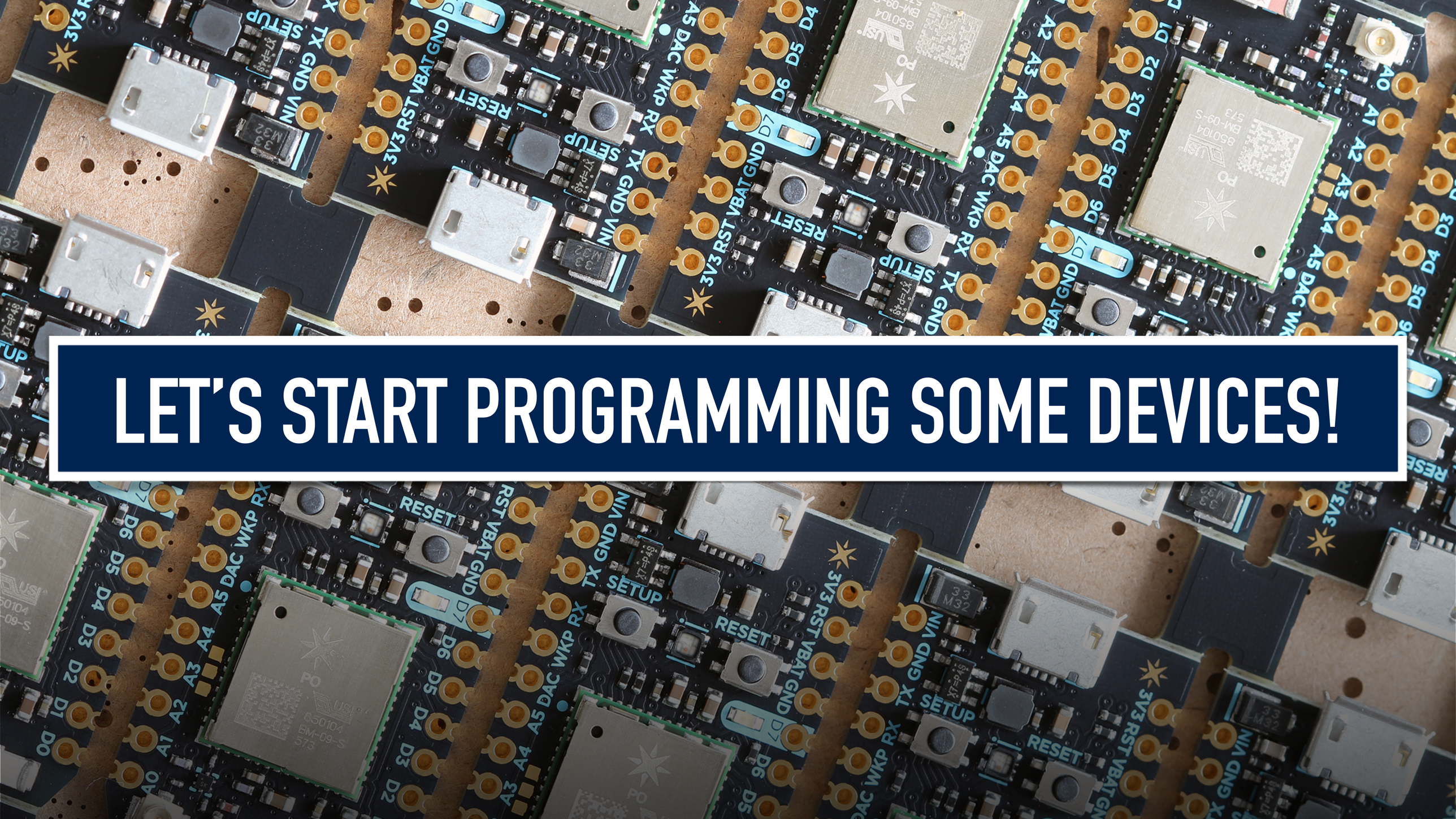
- You want to communicate between devices NOT on the same local network
- You want Particle devices to communicate with other BLE sensors (heart-rate monitors, environmental sensors, etc.)

Use NFC When:

- You want Particle devices to share sensor data with nearby mobile apps.
- To launch a Particle-powered mobile app experience on Android phones.
- To share links to docs, guides, and other web-based resources related to your product.

BLE & NFC




LET'S START PROGRAMMING SOME DEVICES!