Design Specification
Gateway

Version	1.0
Status	Baseline
Date	28-Aug-2020

Contents

FIGURE

1. DOCUMENT DETAILS

1.1 Revision & Approval History

Version	Author		Reviewer		Approver	
	Name	Date (DD-MMM-YYYY)	Name	Date (DD-MMM-YYYY)	Name	Date (DD-MMM-YYYY)
Baseline 1.0	*Viren Moradiya*	*25-Aug-2020*				

Version	Description Of Changes
Baseline 1.0	*Added Gateway firmware design detail.* *Added Gateway architecture detail.*

1.2 Definition, Acronyms and Abbreviations

Definition/Acronym/Abbreviation	Description
MQTT	*Message Queuing Telemetry Transport.*
BLE	*Bluetooth low energy*

1.3 References

No.	Document	Version	Remarks
1.			
2.			

2. INTRODUCTION

2.1 Purpose of the Document

The purpose of this document is to describe in detail design of EFR32 IoT Gateway project modules. This document translates the requirement specifications into a documentation with which developers will create the actual system. It defines top level system architecture and detail design of each module and their interdependencies.

2.2 Intended Audience

NA

3. MODULE OVERVIEW

3.1 Functional description

Dragon board 410 works as gateway and ZigBee Coordinator device is connected to gateway device. Sentimate sensor device connects to ZigBee Coordinator and sending sensor data periodically at fixed interval of time. Gateway connect to AWS cloud and register device to cloud and sending sensor data to cloud.

- ZigBee Coordinator
  ZigBee Coordinator is a service on a Gateway which is responsible to create, manage and commission ZigBee network by controlling the ZigBee NCP device. Network coprocessor (NCP) is the ZigBee capable device which initialize, handle and monitor the ZigBee network based on the command/instruction received from the ZigBee Coordinator application running at the HOST (Gateway) side over UART interface. ZigBee Coordinator communicate with other services on defined MQTT topic for controlling command, responses and end device data exchange.

- Growhouse Server
  Growhouse-server runs as service on board which is responsible to communicate with ZigBee Coordinator, awsapp (IoT stack and application) and BLE Server via mqtt broker.

- BLE server
  BLE Server runs as a service on board which is responsible to communicate with EFR32_Gateway mobile application over BLE. Mobile application connects with BLE Server and request gateway for provision of device.

- awsapp Service
  awsapp application runs as service on a board which is responsible to create gateway on AWS cloud and to communicate with growhouse-server for device provision and sending telemetry data on cloud.

- EFR32_Gateway APP
  EFR32_Gateway mobile application used to provision gateway and sentimate sensor device. EFR32_Gateway app connect to BLE Server running on gateway device, and request to register gateway on AWS cloud.
  For provisioning end device EFR32_Gateway app request gateway to provide available list of sensors from ZigBee Coordinator and then request to provision selected sentimate sensor devices.

- AWS cloud
  Gateway and sensor would be provision on AWS cloud Gateway collects the sensor data from sentimate sensor devices and send it to cloud. Sensor data would be stored in dynamoDB and also used to prepare quicksight Dashboard.

3.1.1 Features

- Growhouse services providing an interface to provision Gateway and ZigBee sentimate sensor devices through EFR32_Gateway mobile app. User can select devices to join in ZigBee Coordinator network.
- Provide an interface to Un-provisioning Gateway through EFR32_Gateway mobile app
- Provide an interface to Un-provisioning ZigBee Sentimate sensor devices from ZigBee network
- Receiving sensor data from ZigBee Sentimate sensor devices and send it to cloud.
- User can set threshold value for Alert/Notification through EFR32_Gateway mobile app.

3.1.2 Use Cases

- Provision Use-case
  Provision of gateway is required to create gateway on the cloud and connect gateway to cloud for sending sentimate sensor data to cloud.
  Provision of devices is required to maintaining the list of devices should be connected to ZigBee Coordinator in one Gateway.

- EFR32_Gateway app is used to provision gateway as well sentimate devices.

- Sensor Data send/receive use-case
  Gateway will receive data from sentimate device and send it cloud for monitoring system.

3.1.3 Functional Block Diagram

Following figure shows the top-level system block diagram of EFR32 IoT Gateway platform.

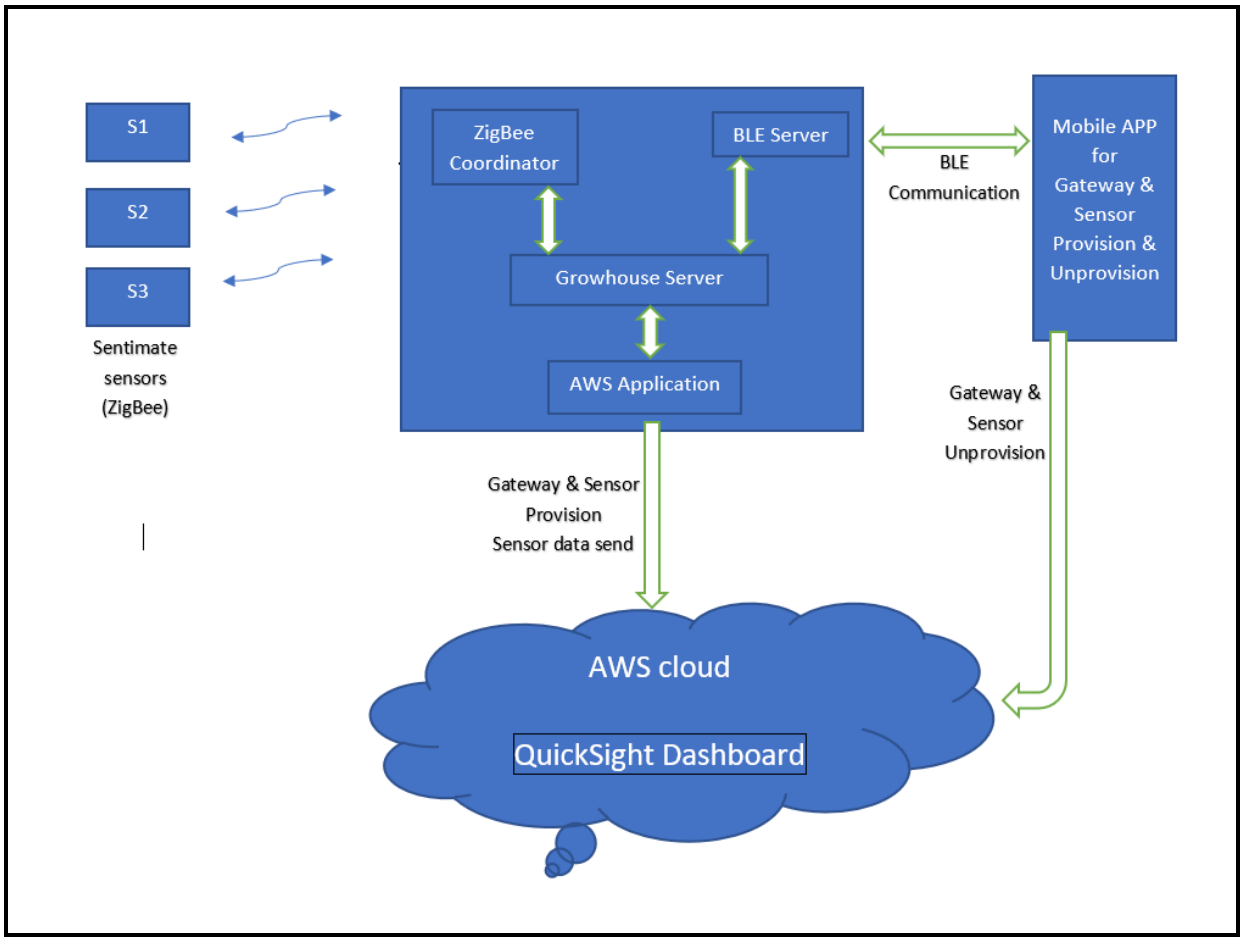

Figure 1 : System Block Diagram

3.1.4 Module Work Flow

1. When Gateway device boots up first, Gateway (410) will start advertising itself as BLE peripheral device named as EFR32_gw_xx:xx:xx:xx:xx:xx (Used gateway mac address).
2. Mobile application will authenticate and connect to this peripheral device while doing provision process.
3. Once request triggered to gateway for provision from mobile application, gateway will be register on AWS cloud platform and store necessary details from response for cloud connectivity.
4. Gateway connects itself with AWS cloud via Ethernet interface.
5. Gateway acknowledges the provision response to mobile application over BLE.
6. If gateway is connected successfully to cloud, gateway will wait for mobile application command "discover devices"
7. After gateway gets "discover devices" command, gateway turns its ZigBee network on and open network security for new device connection and will act as ZigBee Coordinator.
8. Gateway will make ZigBee network open for 3 minutes and nearby ZigBee devices (Sentimate Sensor) would wait for ZigBee network. Once end devices find open network to connect, base on button press on sentimate board, it will send join request. Gateway will generate list of discovered ZigBee end devices.
9. Gateway will provide this generated ZigBee end devices list to mobile application.
10. Mobile application user will select desired devices to be provisioned by select those devices as from application as "provisioned listed". Mobile application will provide request gateway to provision sensor on AWS cloud. Then, Mobile application will wait for response to get device provisioned.
11. Gateway will disconnect all other devices from ZigBee network.
13. After successful device registration gateway will send sensor data payload to awsapp telemetry topic and awsapp will send to cloud.

3.2 Hardware description

3.2.1 System Interface

The following figure shows the top-level block diagram of the hardware design of the Grow House Iot gateway.

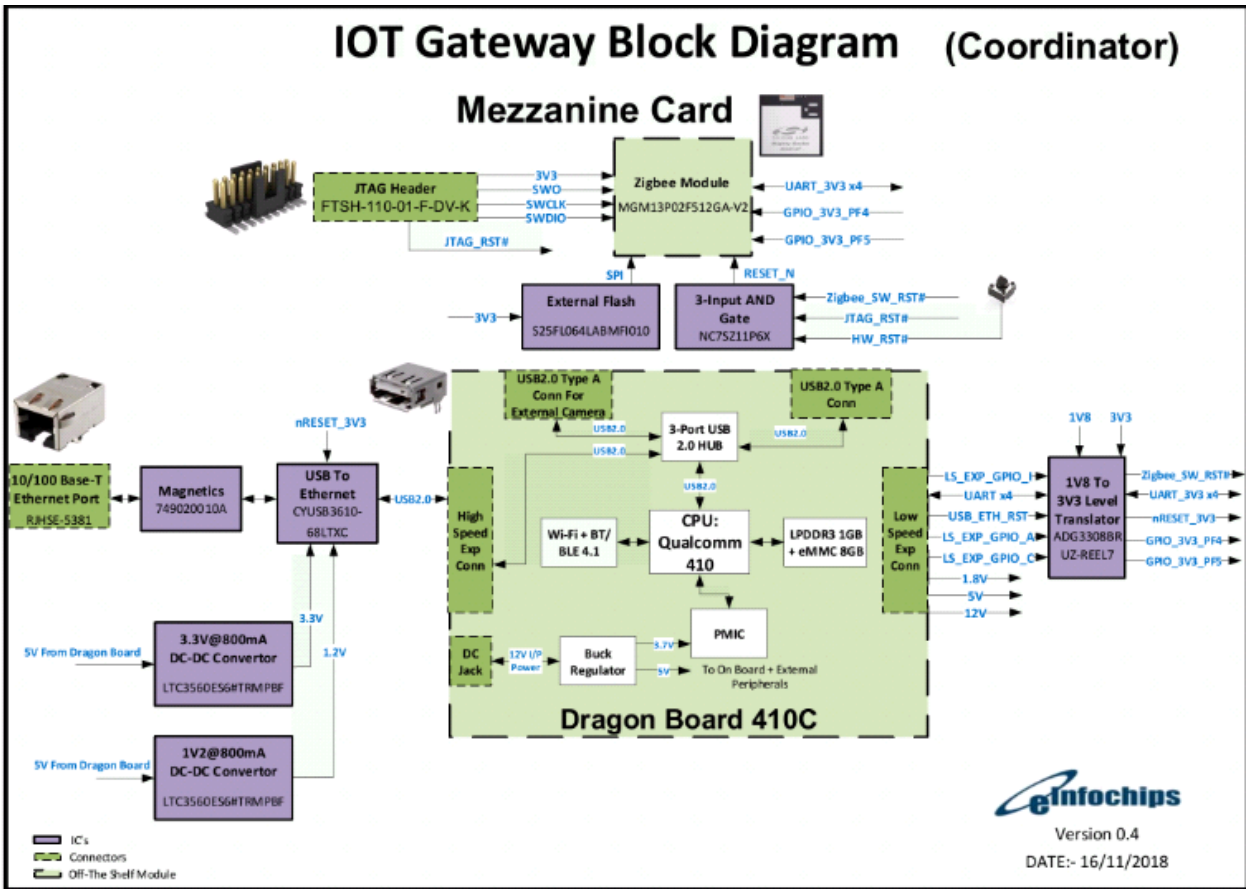

Figure 2 : Top level Block Diagram Of IoT Gateway

4. SOFTWARE OVERVIEW

EFR32 IoT Gateway project basically performs to get sensor data from the Sentimate sensor devices in gateway and send it to IOT cloud and to take action based on sensor value for maintaining overall system.

4.1 Software Implementation

- Startup service description
  When board boot up, below services runs as a growhouse-services.
  1. Mosquitto service
  2. Awsapp Service
  3. Growhouse server service
  4. ZigBee Coordinator service
  5. BLE server service

- **ZigBee Coordinator**

Description: ZigBee Coordinator is a service on a Gateway which is responsible to create, manage and commission ZigBee network by controlling the ZigBee NCP device over UART interface. ZigBee Coordinator communicate with other services on defined MQTT topic for controlling command, responses and end device data exchange.

Location on board: /usr/sbin/zigbee-coordinator

ZigBee Coordinator subscribed and publish for following topic:

1. “gw/+/commands”

   Growhouse server publish all command to ZigBee Coordinator on above topic for creating network, open-network, binding devices, leave devices etc.
2. "gw/+/devices"
   ZigBee Coordinator publish the list of binding devices on above topic.

- **Growhouse-Server**

Description: Growhouse-server runs as service on board which is responsible to communicate with ZigBee Coordinator, awsapp and BLE Server via mqtt broker.

Process Flow and payload information:

1. Growhouse-server will connect to local mqtt server and subscribe for below topics using mosquitto library:

   a. "gw/+/devicejoined"
   Whenever any end-device comes into ZigBee Coordinators network, it will joined into gateway’s network. ZigBee Coordinator gets all information from end-device like endpoints, cluster information,eui64, device state etc.& publish it to “gw/+/devicejoined” topic.

   b. "gw/+/zclresponse"
   Zigbee Coordinator receives attribute data buffer and data type from end-device and publish to this “gw/+/zclresponse” topic.

c. "gw/+/devices"
ZigBee Coordinator publish the list of binding devices on above topic.

d. "ble/gw/discoverCommand"
Growhouse server receives discovery command payload form BLE Server and open or close ZigBee Coordinator network accordingly.

e. "ble/gw/provisionDevice"
Growhouse server receives list of devices to be provision from BLE Sever on above topic and take action accordingly.

f. "ble/gw/shuttingDown"
If mobile application disconnects from gateway BLE then BLE Server publish payload on above gateway.

g. "awsapp/provision/response"
awsapp will provide response to growhouse-server on this topic for gateway and device registration.

- **Awsapp service**

Description: awsapp runs as service on a board which is responsible to provision gateway and sensor device on AWS cloud and to communicate with growhouse-server for sensor device bind/leave and to receive sensor telemetry data and send to cloud.

Location on board: /usr/bin/awsapp
Config file: /opt/awsapp/config.json

Process flow:

1. Initially, after successfully flashing of board, awsapp have an empty config.json file.
2. Gateway provision will be done by mobile app. App will send all required information to BLE server and BLE server will publish to growhouse-server on “ble/gw/provisionDevice” topic and then growhouse-server will publish to awsapp on “awsapp/provision” topic.

3. Awsapp will provision gateway on cloud according to provided payload and store detail in config.json based on response.

Awsapp will subscribe and publish for below topic:

a. “awsapp/provision”
Awsapp will register device on AWS cloud.

b. “awsapp/delete”
Awsapp will delete device related detail from config file and restart service.

c. “awsapp/mqtt/tel”
Awsapp will receive sensor data after sensor provision and send it to AWS cloud.

- **BLE server**

Description: BLE server runs as a service on board which is responsible to communicate with growhouse mobile application. Mobile application connects with BLE server and request for provision and deletion of device.

Location on board: /usr/sbin/ble-server

1. On startup ble-advertise script run and advertise gateway BLE as name EFR32_gw _xx:xx:xx:xx:xx:xx
2. Mobile application connects with gateway BLE by paring device and request for register gateway on AWS cloud as information provided by user.
3. On successfully registration of gateway on cloud, mobile application will get response from gateway and stored detail in local cache.
4. Then If user click on add-device in mobile application then mobile app send discovery command to BLE server and BLE server publish discovery command to growhouse-server.
5. Growhouse-server discover list of devices and publish to BLE server.
6. BLE server notify list of devices to mobile application.
7. After receiving all devices, Mobile application will display available devices for provision and based on user input request to BLE server for sensor provision.
8. BLE server publish the list of devices to be provision to growhouse-server.
9. Growhouse-server will publish the response to awsapp and publish acknowledgement response from awsapp to BLE server.
10. BLE server notify the acknowledgement to mobile application.