Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Internet of Things

IOT on Cloud

Rishi Meghashyam Deorukhkar
Matrikelnummer: 581395

Supervisor:
Prof. Dr. Alexander Huhn

9, Feb 2022

Table of content

1. Introduction **3**

2. Application cloud architecture **3**

2.1. Docker images 4

2.2. Security groups 4

2.3. EC2 instances 5

2.4. Load Balancers 6

2.5. Lambda 6

2.6. Simple notification service (SNS) 6

2.7. API gateway 6

3. Components **6**

3.1. Hardware 8

3.2. Software 9

3.3. User Interactions 9

3.4. User 10

4. **Flows** **10**

4.1. Metric data collection and display 10

4.2. Alert notifications 11

4.3. Actuator triggers 11

5. **Summary and Outlook** **12**

6. References **12**

1. Introduction

The focus of this project is to create a generic architecture for deployment of IOT applications on the cloud. In our use case we use Amazon Web Services cloud platform for the deployment of our architecture. The idea was to create a design which is modular, secure and open for extension for multiple IOT use cases.

As part of this project we implement below listed generic functionalities.

- Metric data collection and storage.
- Dashboard for data observation and monitoring.
- Alerting or alarms for breach of thresholds for metric data.
- Corrective actions in response to alarms.

All of the code and configurations of this project can be found in the below repository.
https://github.com/RishiSD/iot_cloud

Video of the project demonstration can be viewed at below url.
https://mediathek.htw-berlin.de/video/ProITD-IOT-on-Cloud-Project-Rishi-Deorukhkar/bf87f21325ec09eb6c79e1c02ef0a63e

2. Application cloud architecture

All the architecture components are deployed in AWS cloud under default VPC in Frankfurt (eu-central-1) region.

2.1. Docker images

We have pushed all the docker images being used as part of this project to AWS public Elastic Container Registry.
Below is the list of images which can be pulled from the container registry.

- eclipse-mosquitto - public.ecr.aws/e5g0d7d7/eclipse-mosquitto
- grafana - public.ecr.aws/e5g0d7d7/grafana
- influxdb - public.ecr.aws/e5g0d7d7/influxdb
- telegraf - public.ecr.aws/e5g0d7d7/telegraf

2.2. Security groups

Below security groups are being used by the EC2 instances and the load balancers.

- **MqttBrokerSG** - This group allows inbound connection only on port 1886 which is the default port used by the Mosquitto MQTT broker.
- **GrafaTelFluxSG** - This group allows inbound connection only on port 3000 which is the default port on which grafana dashboard is hosted.

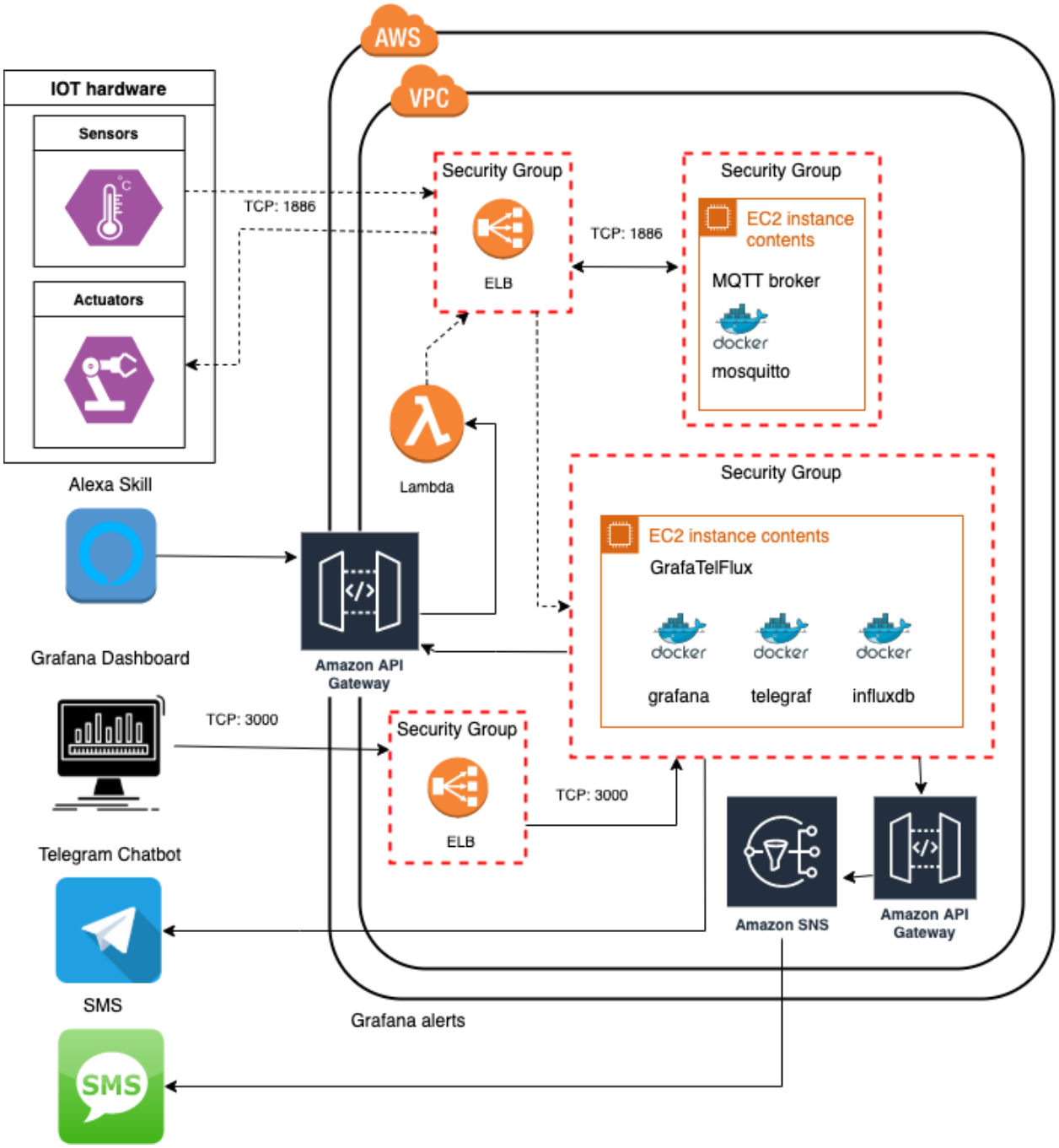

Fig. 1. Architecture of 'IOT on Cloud' application

2.3. EC2 instances

- **MQTT Broker** - This is a t2.micro instance which hosts a single docker container eclipse-mosquitto. It uses the security group MqttBrokerSG.
- **GrafaTelFlux** - This is a t2.micro instance which hosts 3 docker containers grafana, influxdb and telegraf. It uses security group GrafaTelFluxSG. The containers are managed using docker-compose. Since these containers require a persistent volume we have an Elastic Block Store of 8GB attached to this EC2 instance.

2.4. Load Balancers

We have used AWS Classic Load Balancers which sit in front of our EC2 instances.

- **MqttBrokerLB** - This load balancer forwards incoming traffic to MQTT Broker EC2 instance. It uses the security group MqttBrokerSG.
- **GrafaTelFluxLB** - This load balancer forwards incoming traffic to GrafaTelFlux EC2 instance. It uses security group GrafaTelFluxSG.

2.5. Lambda

- **IotActuatorFunction** - This lambda function publishes MQTT message on the broker on actuators topic when invoked. This is triggered via API gateway.

2.6. Simple notification service (SNS)

- We have created a new SNS topic iot_alert and a subscription to that which sends SMS to preconfigured phone numbers.
- SNS is triggered from the API gateway.

2.7. API gateway

We have created below APIs in the AWS API gateway.

- Trigger - This API triggers the lambda function IotActuatorFunction. The API is secured using an API key.
- Alert - This API publishes a message on the SNS topic iot_alert which in turns sends an SMS. The API is secured using an API key.

3. Components

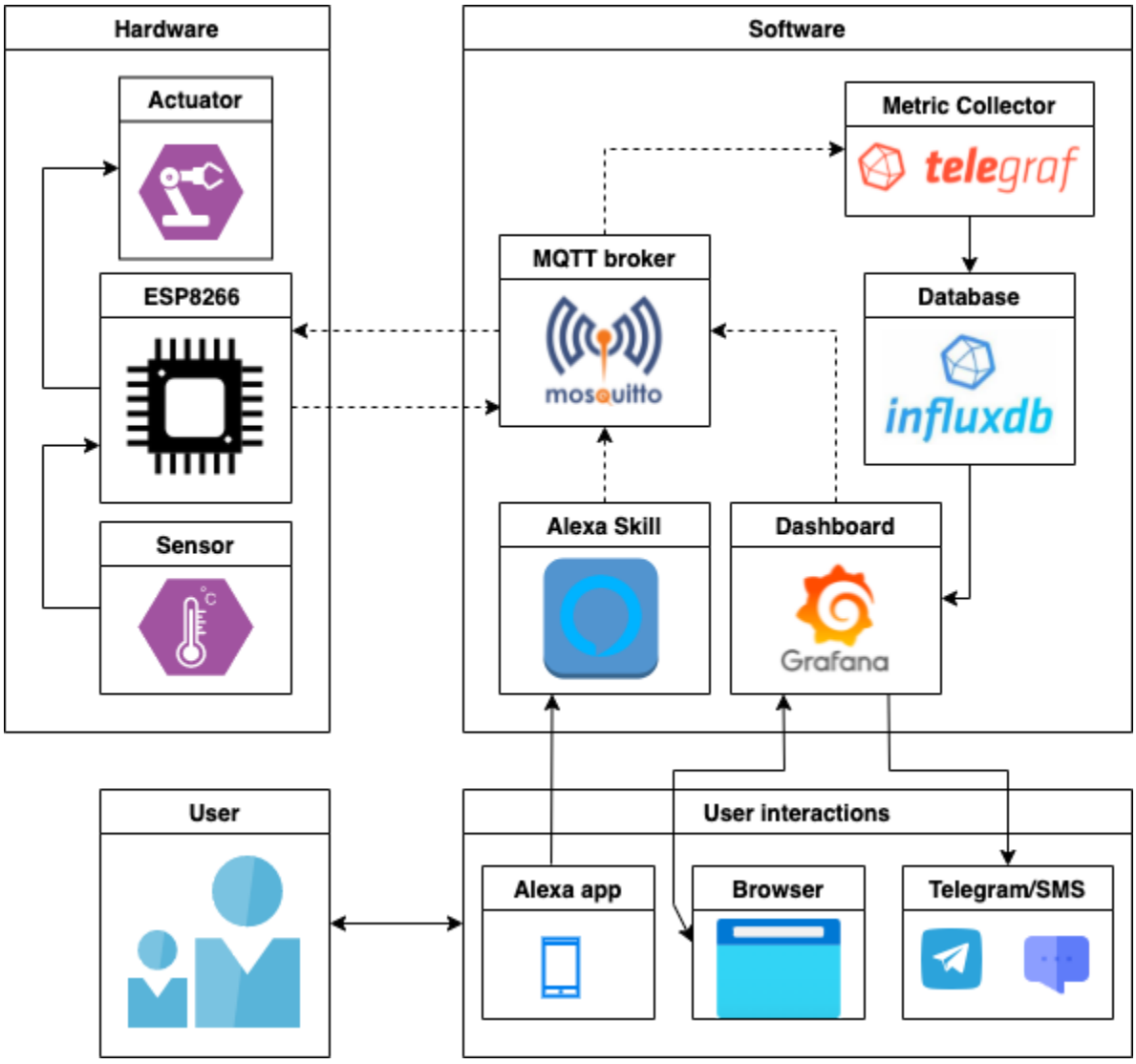

Fig. 2. Component interaction of 'IOT on Cloud' application

3.1. Hardware

3.1.1. ESP8266 NodeMcu

- The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability,
- We use it to integrate the ultrasonic distance sensor HC-SR04.

- Metric data received from the sensor is sent over to an MQTT broker over the internet.

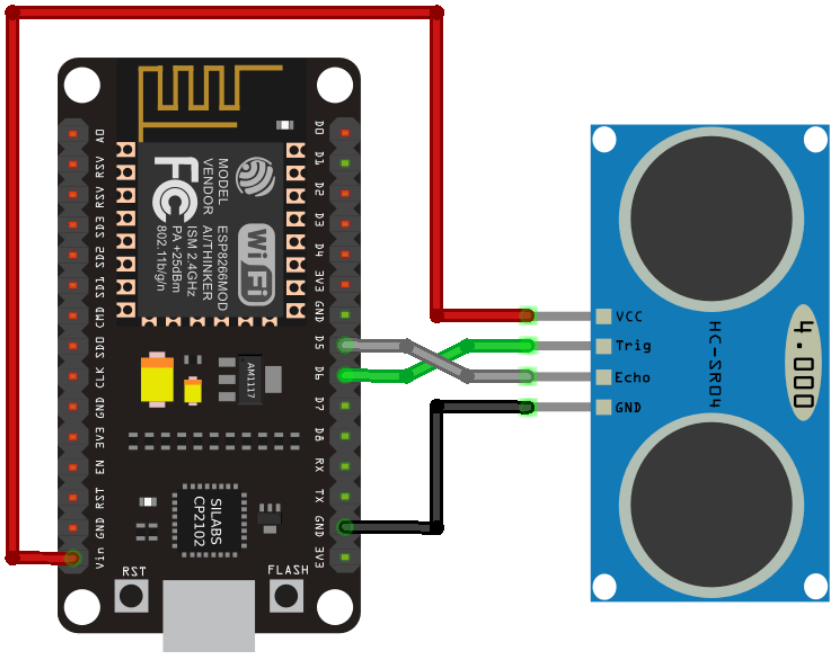

Fig. 3. Schematic diagram for ESP8266 wiring with HC-SR04
Source: https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/

3.1.2. Sensor (HC-SR04)

- **HC-SR04** is an ultrasonic distance sensor.
- This sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm.

3.1.3. Actuator

- An actuator is a machine component or system that moves or controls the mechanism of the IOT system.
- It can be something like an alarm or a switch which acts in response to alarms or user inputs.

3.2. Software

3.2.1. MQTT Broker

Eclipse mosquitto is an open source message broker which implements MQTT protocol. The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model.

3.2.2. Metric Collector

Telegraf is an open source server agent which helps in metric collection from sensors and system and pushes it to a configured target in our use case that would be an influxdb database.

3.2.3. Database

InfluxDB is an open source time series database used for storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics.

3.2.4. Dashboard

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. It has options to send alert notifications on multiple channels, we will use telegram and webhooks in our use case.

3.2.5. Alexa Skill

Skills are like apps for Alexa which is a virtual assistant that can work with voice and text, and provide a new channel for your content and services. Skills let customers use their voices to perform everyday tasks in our case we use it for triggering actuators.

3.3. User Interactions

3.3.1. Browser

Users can access the grafana dashboard using a web browser on a computer or mobile device since the dashboard is hosted on the internet. The dashboard is secured with a username and password so that only authorized users have access to it.

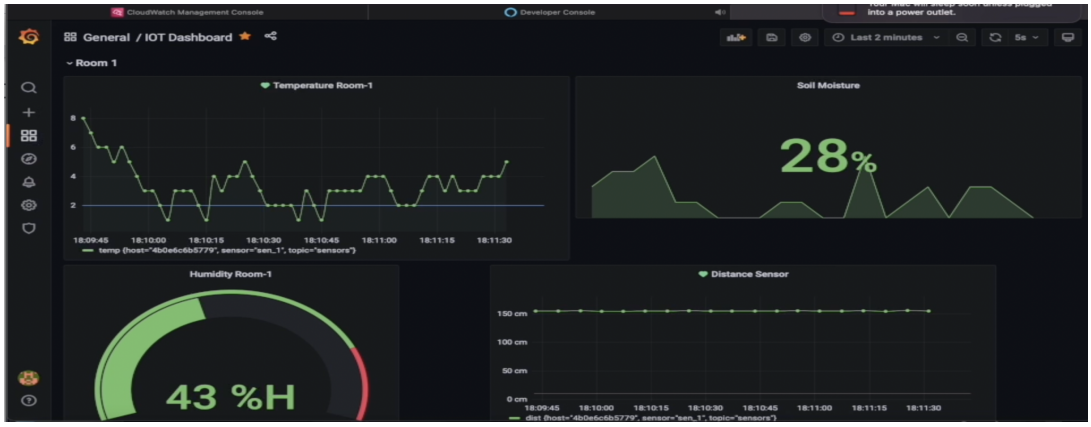

Fig. 4. Grafana dashboard on desktop browser

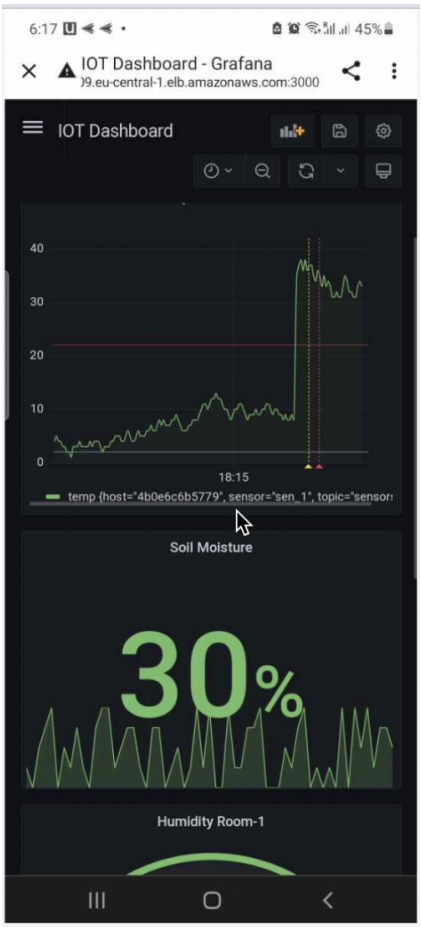

Fig. 5. Grafana dashboard on mobile browser

3.3.2. Telegram Chatbot / SMS

Alerts from grafana dashboard will be sent to Telegram Charbot and/or in the form of an SMS on preconfigured mobile phone numbers.

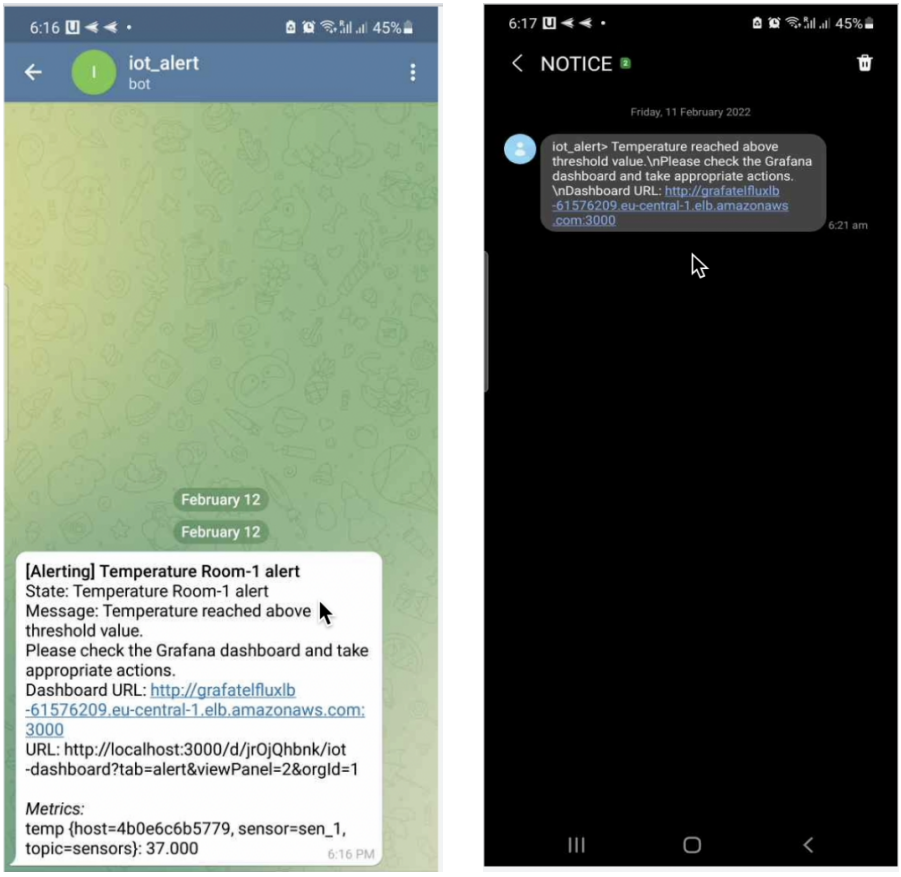

Fig. 6. Telegram chatbot and SMS alert

3.3.3. Alexa App

Users can use the alexa app to interact with actuators in response to alerts or otherwise. An actuator can be something like a switch which can be used to regulate temperature or turn the light on or off.

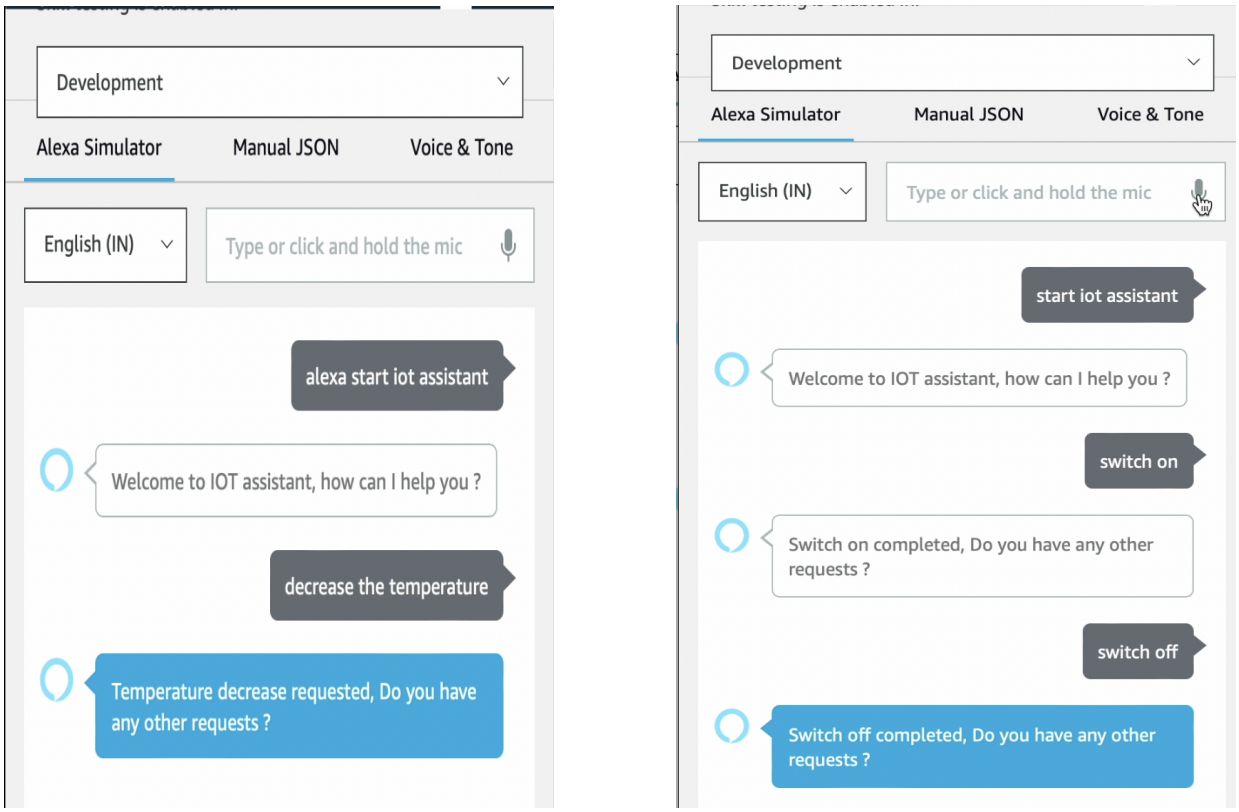

Fig. 7. Alexa app interactions

3.4. User

The user is any entity which can interact with the dashboard and the actuators, and is able to receive alert notifications.

4. Flows

4.1. Metric data collection and display

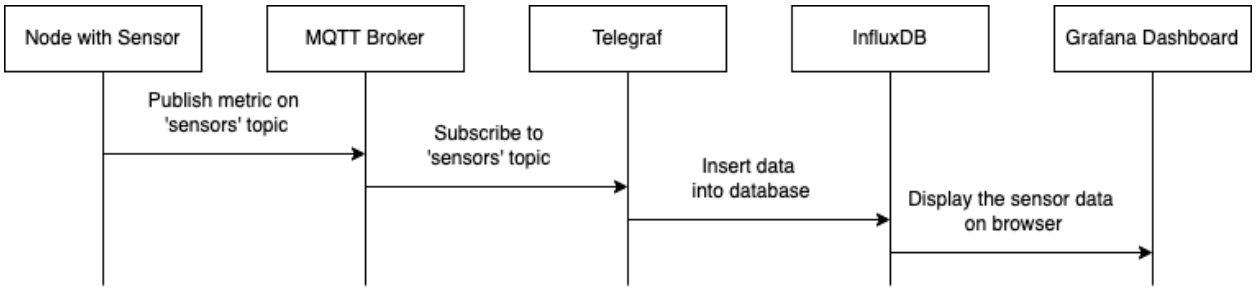

Fig. 8. Flow diagram for metric collection and presentation

- The node can be any microcontroller connected to a sensor reading some metric quantity.
- The node published value from sensor to MQTT broker on ‘sensors’ topic, the published value needs to be in below format for influxdb to support it.

```
weather,location=us-midwest temperature=82 1465839830100400200
  |    -------------------- --------------  |
  |             |             |             |
  |             |             |             |
+-----------+--------+-+---------+-+---------+
|measurement|,tag_set| |field_set| |timestamp|
+-----------+--------+-+---------+-+---------+
```

Fig. 9. InfluxDB line protocol format
Source: https://docs.influxdata.com/influxdb/v1.8/write_protocols/line_protocol_tutorial/

- Telegraf subscribes to the ‘sensors’ topic and inserts the data into the InfluxDB database under a pre-decided bucket which is mentioned in the telegraf config.
- Grafana dashboard queries data from InfluxDB as per the panel configuration for each sensor and displays the data to users in the form of a real time dashboard.

4.2. Alert notifications

- Grafana can configure thresholds for each sensor panel and trigger an alert on the breach of those threshold values.
- Alerts in our use case uses Telegram chatbot and Webhooks for SMS as channels.

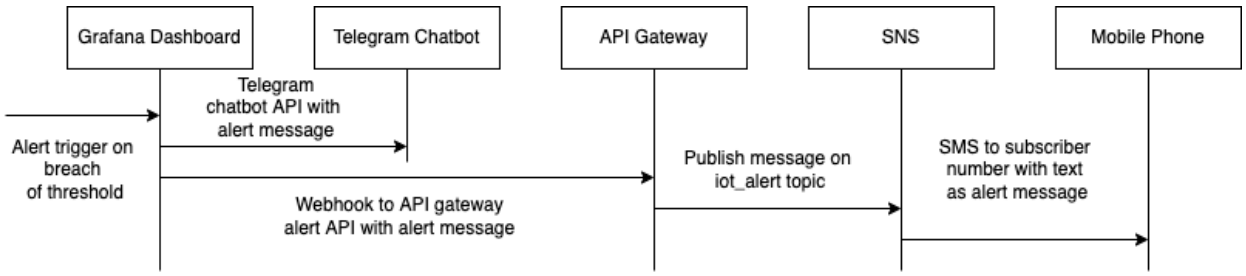

Fig. 10. Flow diagram for alert notifications

- For webhooks as notification channel grafana sends a request to API gateway alert API with body as the alert message.
- Alert API has a service proxy with AWS SNS which sends an SMS to the configured mobile phone numbers.

4.3. Actuator triggers

- Users can use the IOT assistant skill developed as part of this project to trigger actions in response to alerts.
- Currently the skill supports commands like “switch on”, “turn switch off”, “decrease temperature”.

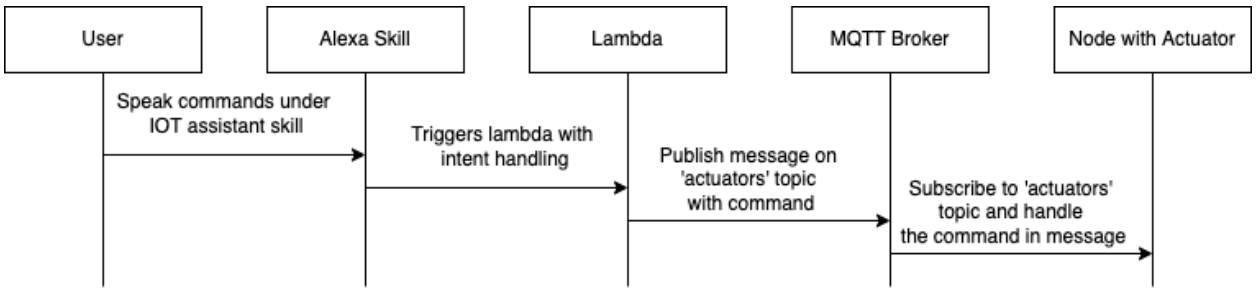

Fig. 11. Flow diagram for actuator triggers

- If the command keywords match any configured Intent the intent is triggered and an MQTT message is published on the ‘actuators’ topic to the MQTT broker.
- The ‘actuators’ topic is subscribed by the node and the corresponding action is performed.

5. Summary and Outlook

- The project tries to build a loosely coupled framework that can be extended for specific and more complex IOT projects.
- Further improvements can be made in terms of using AWS IOT core as the managed MQTT broker and replacing mosquitto broker completely, this will remove the risk of single point of failure and the need for manually managing container replication for MQTT.
- For more resilient architecture we can switch to serverless influxDB which is provided by AWS instead of using a custom container.
- Use auto-scaling of container or elastic container service for dealing with high volume of sensor data and users.

6. References

- ESP8266 Technical Reference. (n.d.). [online] Available at: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf.
- docs.influxdata.com. (n.d.). InfluxDB OSS 2.1 Documentation. [online] Available at: https://docs.influxdata.com/influxdb/v2.1/ [Accessed 12 Feb. 2022].
- cdn.sparkfun.com. 2022. [online] Available at: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf> [Accessed 12 February 2022].
- Wisintainer, Miguel. “MicroPython - Getting Started with MQTT on ESP32/ESP8266 | Random Nerd Tutorials.” Randomnerdtutorials.com, 7 Dec. 2018, randomnerdtutorials.com/micropython-mqtt-esp32-esp8266/. Accessed 12 Feb. 2022.
- Telegraf 1.21 Documentation.” Docs.influxdata.com, docs.influxdata.com/telegraf/v1.21/. Accessed 12 Feb. 2022.
- “Grafana Documentation.” Grafana Labs, Grafana Labs, 2020, grafana.com/docs/grafana/latest/.
- “Documentation.” Eclipse Mosquitto, 6 July 2020, mosquitto.org/documentation/.