Horizon 2020 PROGRAMME *ICT-01-2014: Smart Cyber-Physical Systems*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 643924

D4.7
Demonstrator 3
EoT Application

1. DOCUMENT INFORMATION

Authors	J.M. Rico (UCLM) J.L. Espinosa-Aranda (UCLM) N. Vallez (UCLM) J. Parra (UCLM) M. Herbst (EVERCAM) V. Quinn (EVERCAM) J. Farooq (EVERCAM A. Pagani (DFKI)
Responsible Author	Alain Pagani (DFKI) e-mail: alain.pagani@dfki.de

Keywords	Demonstrator 3 – Flexible Mobile Camera
WP/Task	WP4
Nature	Other
Dissemination Level	PU

2. DOCUMENT HISTORY

Person	Date	Comment	Version
Alain Pagani	06.06.2018	Delivered version	1.0

3. ABSTRACT

This document describes the EoT Device application of the Flexible Mobile Camera demonstrator. The Flexible Mobile Camera demonstrator describes a portable smart camera powered by the EoT Device.

This document presents the development process of the demonstrator and the requirements fulfilled and a brief explanation of the main application implemented. Moreover, the EoT libraries used are indicated.

4. TABLE OF CONTENTS

5. INTRODUCTION

This document describes the Flexible Mobile Camera demonstrator EoT Device application. In this demonstrator, the EoT Device is connected to an Android device and two cloud services, Firebase Cloud Messaging and Google Cloud Storage.

To carry out this communication, the EoT Device must be configured. This configuration assumes that:

- The EoT device has the necessary files stored in the SD card.
- The EoT device has a valid network profile.
- The EoT device can connect with Firebase Cloud Messaging and Google Cloud Storage. This step requires flashing a security certificate.
- The EoT device can change the images capture configuration.

The Flexible Mobile Camera demonstrator is configured using the configuration application.

The EoT Device application can be found in https://gitlab.com/espiaran/EoT/tree/master/WorkPackage_4/Lifelogging_Camera/EoT Device/WearableEoT Device

The reviewers will be able to access the private parts of the code on request.

6. SHORT DESCRIPTION OF THE DEMONSTRATOR

Figure 1: Narrative Clip

Other examples of secondary features include:

- Retrieval / Browsing … by grouping images by, for example, similarity, one can offer an improved image navigation experience.
- Cloud Applications such as a “Daily Collage” curating a single image with all the key events of the day.

The EoT device would differ in its ability to do image processing locally, enabling:

- Smart Storage (Only store what’s needed)
- Annotation (Compile meta data for queries such as “show me all faces”)
- Smart Alerts (Alert (e.g. to Smartwatch))

For each stored photo, the cameras would also store meta data by running each of the EoT device’s capabilities on the image to either trigger alerts, enable queries or schedule deletion.

7. EOT APPLICATION SOFTWARE DESCRIPTION

1. Requirements

1. Functional requirements

Below are listed each of the requirements with a brief description of the goal. These requirements are a revision of the original requirements. The original requirements can be consulted in the 'Annex 2: Demonstrator Requirements Documents'. The REQ008 described in that document, GPS Awareness, has been removed because the EoT Device has no GPS.

ID	Name	Priority	Difficulty	Description
REQ001	Store Images	High	Low	Store images to SD card
REQ002	Classify Images	High	High	Determine the contents of the image
REQ003	Synchronise to Cloud	High	High	Copy fresh images to cloud & free up local storage
REQ004	Firmware Updates	High	High	Retrieve firmware updates remotely
REQ005	Settings Sync	Medium	High	Retrieve new settings
REQ006	Image Content Meta Data	High	High	Transmit Meta Data about Image Content (Faces, Objects etc.)
REQ007	Connect to bridge	High	Low	Connect to mobile phone using bridge concept
REQ008	Apps Sync	High	High	Retrieve new apps
REQ009	Alerts	High	Low	MQTT style messaging

The above table shows the requirements of the complete demonstrator. It includes the EoT Device application and the Configuration application.

The EoT Device application includes the REQ001, REQ002, REQ006 and REQ009 functional requirements. There are two functional requirements shared between the EoT Device application and Configuration application. These requirements are the REQ005 and REQ007.

The WebApp includes the REQ008 functional requirement.

The REQ004 must be done using other application. This application is the Pulga Control App in the PC part and the bootloader in the EoT Device part.

2. Non-functional requirements

ID	Name	Type	Priority	Difficulty	Description
NFR001	Battery duration	Performance	High	High	Battery of the device used in the system should last for more than one hour at least
NFR002	Ruggedness	Performance	High	Low	The device must be able survive splashes and dropping
NFR003	Storage of images	Performance	High	Low	Phone must store a reasonable # of images (128GB?)
NFR004	Storage of 3rd party data	Performance	Med	Med	For social network apps

The most of the non-functional requirements are achieved due to the characteristics of the EoT device, as the NRF001 and the NFR003 non-functional requirements. The connectivity with 3rd party data (NFR004 non-functional requirements) is achieved thanks to the WebApp.

In the original non-functional specification there was five non-functional requirements. In this update, the NFR005 (Sync via mesh of other phones) requirement has been finally discarded because it was not necessary for the functionality of the demonstrator.

2. Modules

This section describes the composition of the EoT Device application. It consists of seven modules. Most of them include parts of various functional requirements. Others complete some requirement.

1. Capture modes module

1. Description

This module provides the different capture modes. The demonstrator has two capture modes:

- Normal mode: In this mode, one image is captured every X seconds.
- Smart mode: Each image is examined and in case it contains a key event (as defined in the configuration), the image will be stored.

2. Achieved Requirements

This module does not achieved a requirement, but it helps to get the REQ001, REQ002, REQ006 and REQ009 functional requirements.

2. Classification of images module

1. Description

The method used to classify the images captured by the previous module is implemented in this module. This module allows to the demonstrator to classify the following 'key events'.

- FaceDetected. There is a face in the image.
- LargeFaceDetected. There is a near face in the image. A near face may be indicative of interaction with that person.
- Anger. There is an angered face in the image.
- Disgust. There is a disgusted face in the image.
- Fear. There is a scared face in the image
- Happiness. There is a happy face in the image.
- Neutral. There is a neutral face in the image.
- Sadness. There is a sad face in the image.
- Surprise. There is a surprised face in the image.
- MotionDetected. A movement has been detected.

2. Achieved Requirements

This module does achieved the REQ002 functional requirement. It helps to get the REQ006 and REQ009 functional requirements.

3. Image Storage module

1. Description

Management of the SD card storage is included in this module. The image storage module has 3 functionalities:

- Storage. With this functionality, the images and metadata are stored in the SD card.
- Recovery. With this functionality, the images and metadata are retrieved and prepared to be sent through MQTT.
- Erasing. Using this functionality, when the images are sent to the configuration application, they will be removed from the SD card.

2. Achieved Requirements

This module does achieved the REQ001 and REQ006 functional requirements. It helps to get the REQ003 and REQ005 functional requirements.

4. MQTT Broker module

1. Description

The MQTT broker handling Wi-Fi communications is implemented in this module. This broker will run in the EoT Device. This module is formed by the following sub modules.

- MQTT topics. This sub module contains the MQTT topics, the payload of each topic and how to work with this payload will be specified.

- MQTT broker. The MQTT broker is able to:
  - Accept connections.
  - Send images and their metadata.
  - Receive configuration parameters.
  - Receive application updates.

2. Achieved Requirements

This module does achieved the REQ003 functional requirement. It helps to get the REQ005 functional requirement.

5. Notifications module

1. Description

This module is responsible for the alarm functionality. The notifications module works according to what is explained in the section 8.2.6. The notifications are received by the Android App.

2. Achieved Requirements

This module does achieved the REQ009 functional requirement.

6. Configuration section module

1. Description

With this module, the EoT Device configuration is possible. This configuration includes:

- Setting alarms.
- Setting capture mode.
- Setting network parameters.
- Setting the MQTT parameters.

The EoT Device receives the configuration parameters wirelessly using the MQTT protocol. The EoT Device application running in the EoT device stores/updates the configuration in a file in the SD card.

2. Achieved Requirements

This module does achieved the REQ005 functional requirement.

7. Connectivity

1. Description

This module adds to the demonstrator the possibility of the connection of the EoT device with a network and the Android app.

2. Achieved Requirements

This module does achieved the REQ007 functional requirement. It helps to get the REQ003, REQ005, and REQ009 functional requirements.

3. Software architecture

The following image shows the demonstrator structure. It includes the EoT device, the configuration application, the WebApp and the communications and connections.

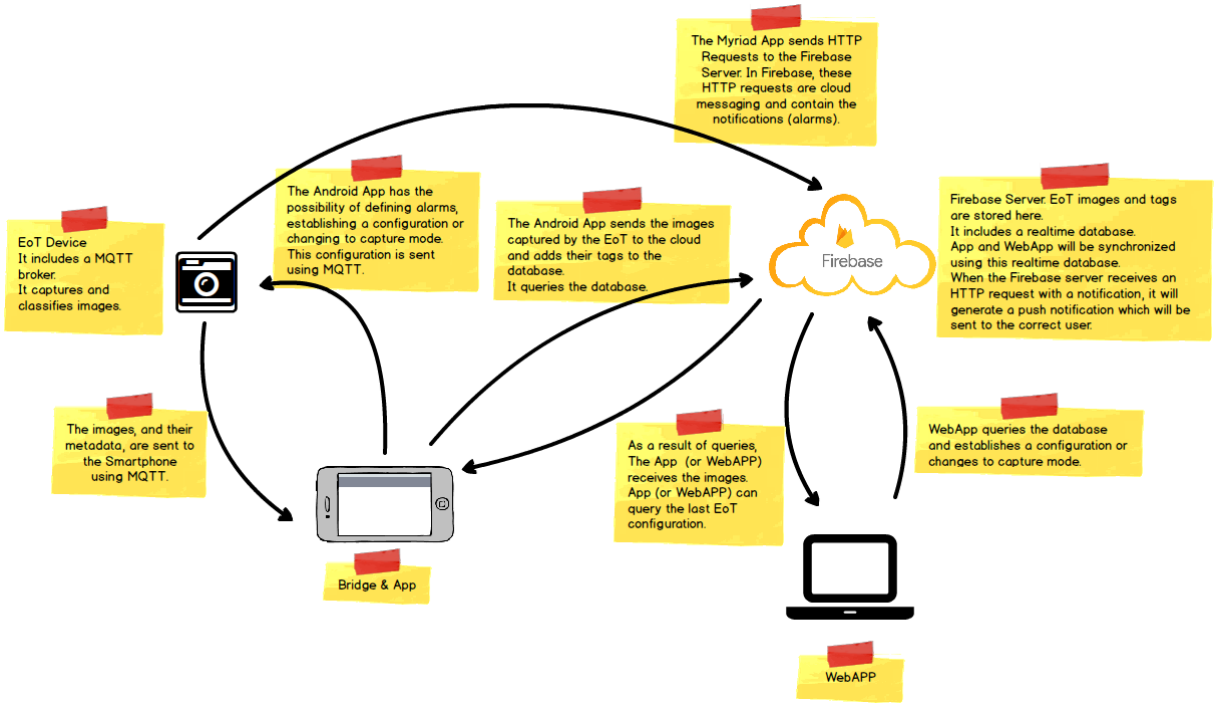

Figure 2: Demonstrator structure

This document is focused on the EoT Device Application. Therefore, the specific scheme is shown by the next figure.

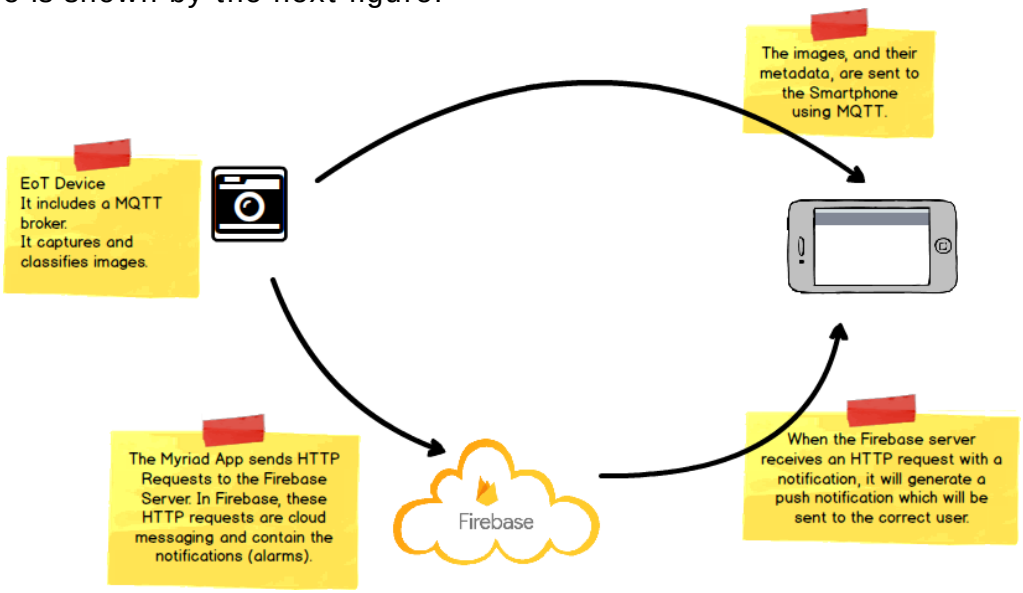

Figure 3: EoT Device Application structure

In the Figure 3, it is possible to see:

- The EoT device captures and classifies images.
- The EoT Device application starts a MQTT broker in the EoT device. Using the MQTT broker, the EoT device is able to send images to the connected Android device. In the same way, using the MQTT broker, the EoT device is able to receive the capture configuration.

- The EoT device is able to communicate directly with the Firebase Cloud Messaging and the Google Storage Cloud services. This way, the EoT Device is able to send push notifications with an image. This process is carried out with HTTP requests.

8. EOT APPLICATION SOFTWARE DOCUMENTATION

1. EoT libraries used

To develop the EoT Device application of the Flexible Mobile Camera demonstrator, the following EoT libraries have been used:

- **WifiFunctions**: This library is used to connect the EoT Device with a network. In addition, the library allows connect the EoT Device with the Firebase and Google Cloud services.
- **SDCardIO**: This library is used to store the images and their metadata in the SDCard and to load the neuronal network from the SDCard
- **TimeFunction**: This library is used to update the time and date of the EoT device.
- **Camera**: This library is used to capture the images using the camera.
- **MQTT**: This library is used to create a MQTT broker. This broker is used to communicate the EoT Device with the configuration application.
- **tiny_dnn**: tiny_dnn is a C++14 implementation of deep learning. It is suitable for deep learning on limited computational resource, embedded systems and IoT devices. This library is used to detect the facial expression of the captured images.
- **JsonParser**: This library is used to read/write the configuration file.
- **Libccv/OpenCV:** These libraries are used to work with the capture image. These libraries allow using computer vision algorithms, as face detection, image rotation, etc.

2. Configuration of the EoT Device

In order to save the network configuration, the EoT Device uses the CC3100 flash memory. The network profiles will be stored in the CC3100 flash memory. The same will happened with the security certificate.

To connect the EoT Device to the Firebase and Google cloud services is necessary to flash a security certificate in the CC3100 flash memory. Since April 2018, the necessary certificate to make this connection is the GS Root R2 certificate.

To do this, the wifiFlashCertificate application can be used. This application flashes the GS Root R2 certificate in the CC3100 flash memory with the name /cert/gsr2.der.

In order to use the wifiFlashCertificate application, the user must store the certificate file in the root of the SD card with the name “gsr2.der” and run the wifiFlashCertificate application. This application can be downloaded from: https://gitlab.com/espiaran/EoT/tree/UCLM_FFBoard_mdk_17.04.5/WorkPackage_3/myriad/apps/flashWifiCertificate. The certificate file can be downloaded from the same web (googleG3.der)

The EoT Device application uses a neuronal network able to detect facial expressions. This network must be stored in the SD card in a folder named 'nviso2' placed in the root of the SD card. In the same way, the necessary files for the face detection must be stored in the SD card in a folder named 'Rotation-invariant_faceDetector\face' placed in the root of the SD card.

Finally, a folder named 'WearableEoTImages' must be created in the SD Card root. This folder will contain the stored images and tags.

3. Use cases

1. Initial configuration

1. Description

When the EoT device is not configured, the EoT Device application will start the EoT device in AP mode and it will run a MQTT Broker. This way, the EoT device will be waiting for a MQTT client. This MQTT client will be the configuration application, and the configuration will start at this point.

Just before creating the MQTT broker, the EoT Device will create an empty JSON configuration file in the SD card.

```
{
	"Configuration": [{
		"captureMode": 0,
		"capturePeriod": 5,
		"blur": 0,
		"store": 0,
		"imu": 0,
		"refreshToken": "",
		"uid": "",
		"Alarms": [{
			"faceDetected": 0,
			"largeFaceDetected": 0,
			"anger": 0,
			"disgust": 0,
			"fear": 0,
			"happiness": 0,
			"neutral": 0,
			"sadness": 0,
			"surprise": 0,
			"motionDetected": 0
		}],
		"KeyEvents": [{
			"faceDetected": 0,
			"largeFaceDetected": 0,
			"anger": 0,
			"disgust": 0,
			"fear": 0,
			"happiness": 0,
			"neutral": 0,
			"sadness": 0,
			"surprise": 0,
			"motionDetected": 0
		}],
```

```
            "ROI": [{
                  "BS": 0,
                  "ROIx": 0,
                  "ROIy": 0,
                  "ROIw": 479,
                  "ROIh": 255
            }]
      }]
}
```

As shown, the configuration file contains the key events and the capture options. In addition, the configuration file has the 'UID' and 'refreshToken' parameters. These parameters are obtained from the configuration application in the initial configuration and they are necessary to the correct working of the EoT Device application. Without them, the EoT device will not be able to communicate with Firebase and Google cloud services.

When a MQTT client connects with the EoT Device, the EoT Device application will be waiting for the configuration request. When the configuration request occurs, the EoT Device application will send the MAC direction of the EoT device to the MQTT client and it will receive the UID and access_token parameters.

The next step is to obtain and store a network profile. This profile is sent by the configuration application and it will be stored in the flash memory of the CC3100. After that, the EoT device will finish it configuration following the next steps:

- The EoT device will change from AP mode to Station mode-
- The EoT device will try to connect with a valid network profile
- The EoT device will try to obtain a correct date.
- The EoT device will try to connect with the Google Cloud Storage and Firebase Cloud Messaging servers. In the first configuration, obtaining the 'refreshToken' is necessary. The EoT Device application will send request to the Firebase and Google servers to obtain this token.

Once the previous steps are finished, the EoT Device application is configured. The configuration file after this process can be see below:

```
{
      "Configuration": [{
            "captureMode": 1,
            "capturePeriod": 5,
            "blur": 0,
            "store": 1,
            "imu": 0,
            "refreshToken": "1/5UJMdH1VbUeCiVDzhrsmlq3h7X6pKLBW1GSnUotwuts",
            "uid": "fc0AxWAW70dUseZVXFzppXmuWoX2",
            "Alarms": [{
                  "faceDetected": 1,
                  "largeFaceDetected": 1,
                  "anger": 0,
                  "disgust": 0,
                  "fear": 0,
                  "happiness": 0,
                  "neutral": 0,
                  "sadness": 0,
```

```
                "surprise": 0,
                "motionDetected": 0
            }],
            "KeyEvents": [{
                "faceDetected": 1,
                "largeFaceDetected": 0,
                "anger": 0,
                "disgust": 0,
                "fear": 0,
                "happiness": 0,
                "neutral": 0,
                "sadness": 0,
                "surprise": 0,
                "motionDetected": 0
            }],
            "ROI": [{
                "BS": 0,
                "ROIx": 0,
                "ROIy": 0,
                "ROIw": 479,
                "ROIh": 255
            }]
      }]
}
```


If the configuration process fails, the user must delete the configuration file created in the SD card and must start the configuration process from the beginning.

2. Achieved requirements

This use case shows that the REQ007 has been developed correctly. In addition, this use case configures the EoT Device for the rest of functional requirements.

2. Start and connection to the configuration app

1. Description

If the EoT device is correctly configured, the EoT Device application will load the configuration file and will try to connect to an AP using a stored network profile.

If this step is successful, the EoT Device application will start the Normal scenario.

Normal scenario: Internet connection

After loading the configuration file, the application will try to connect to a network profile. If the connection is successful, the EoT Device application will configure the hour and the date of the EoT device and will establish a connection with the Google Cloud Storage and Firebase Cloud Messaging cloud server.

Following this, the EoT Device application will send the IP and port of the EoT Device through a push notification. This push notification will be sent to a Topic. For security reasons, the topic will be the UID of the user. This parameter is known by both sides, and the configuration application is subscribes to it.

After this step, the EoT Device application will be waiting to a MQTT client. When a MQTT client connects to the MQTT broker, the EoT Device will load the CNN for the expression detection. Finally, the EoT device will be able to capture images.

Alternative scenario: Without Internet connection

After loading the configuration file, the application will try to connect to a network profile. If the connection is not successful, the EoT Device application will configure the EoT Device in AP mode. In addition, the EoT Device application will run a MQTT broker, and will be waiting for an MQTT Client.

When a MQTT Client connects with the EoT Device, the EoT Device will load the CNN for the expression detection. Finally, the EoT device will be able to capture images.

In this scenario the push notification will not work because there is not communication with Firebase and Google Cloud servers

3. Capture settings

1. Description

The EoT Device will capture images according to it capture configuration. The capture configuration includes the active/disable 'key events'. The configuration is changed by the configuration Application and the EoT Device receives the changes using the MQTT broker.

When a capture change comes, the EoT Device application updates it configuration. However, this communication is not stored in the configuration file until the configuration application leaves the operation menu.

The same behaviour is given in the configuration of the alarms.

2. Achieved requirements

This use case shows that the REQ005 has been developed correctly.

4. Images capture

1. Description

There are two capture modes. The Normal mode and the Smart Mode. In the normal mode, one picture is stored each X seconds. The user using the configuration application can configure the X parameter.

In the smart mode, the images are stored if they have an active 'key event'.

The images are stored in the micro SD. Besides each image, a text file with the metadata of the image is stored. This text file contains the 'key events' actives in the image.

If the store option is disabled, only the text file with the metadata of the image will be stored.

2. Achieved requirements

This use case shows that the REQ001, REQ002 and REQ006 has been developed correctly.

5. Synchronise

1. Description

When the configuration application requests a sync, a MQTT message is sent to the EoT Device using MQTT. Once received, the EoT Device application will start to send the images and their metadata to the configuration application using MQTT. In this moment, the image capture stops.

When an image is sent to the configuration application, the image and its metadata is deleted from the SD card.

Once sent the images and their metadata, the EoT Device application returns the EoT device to its previous state.

2. Achieved requirements

This use case shows that the REQ003 has been developed correctly

6. Push notification

1. Description

The EoT Device application allows configuring the 'key events' for the alarms, in the smart mode. When a 'key event' occurs and it alarm is active, the EoT Device application will send the captured image to the Google Cloud Storage and it will use the FCM service to notify to the configuration control that an alarm has been generated.

The global scheme is the following:

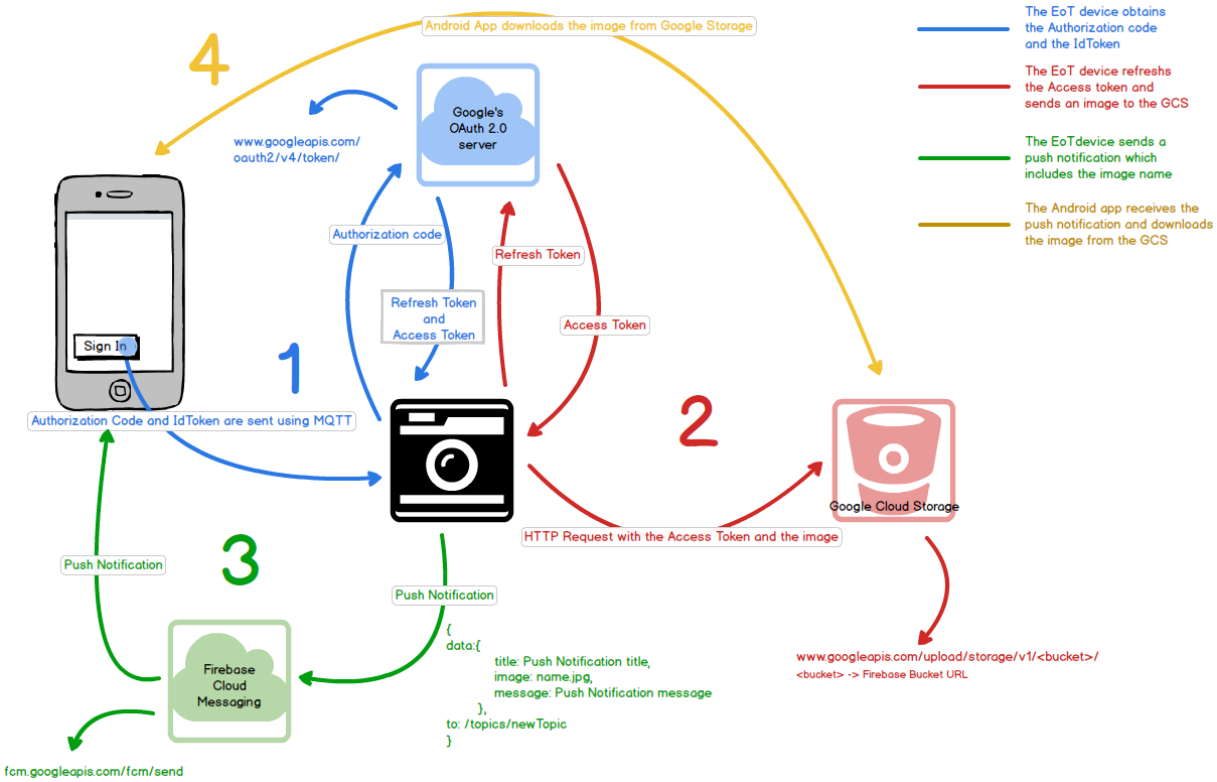

Figure 4: Push notification scheme

The Figure 4 shows the process followed by the EoT device to send an image to the GCS and generate a push notification. The process is divided in four steps:

1- In this step, the EoT Device obtains the necessary tokens to send files to the GCS. This process is only performed when the device is initially configured.
   a. When the user signs in with a Google account, the Android App obtains the *Authorization code*. This *Authorization code* can be modified to a different scope. In this case, the Android App provides permissions in the https://www.googleapis.com/auth/devstorage.read_write scope.
   b. The *Authorization code* will be sent to the EoT device using the MQTT protocol.
   c. To send a file to the GCS it is necessary to obtain the *access token*, being required in the HTTP Request in the Authentication header. When the EoT device sends it to Google's OAuth 2.0 server, this server responses with the *Refresh Token* and the *Access Token*. The *Access token* expires after 3600 seconds, but it can be updated with the *Refresh token*.

HTTP Request

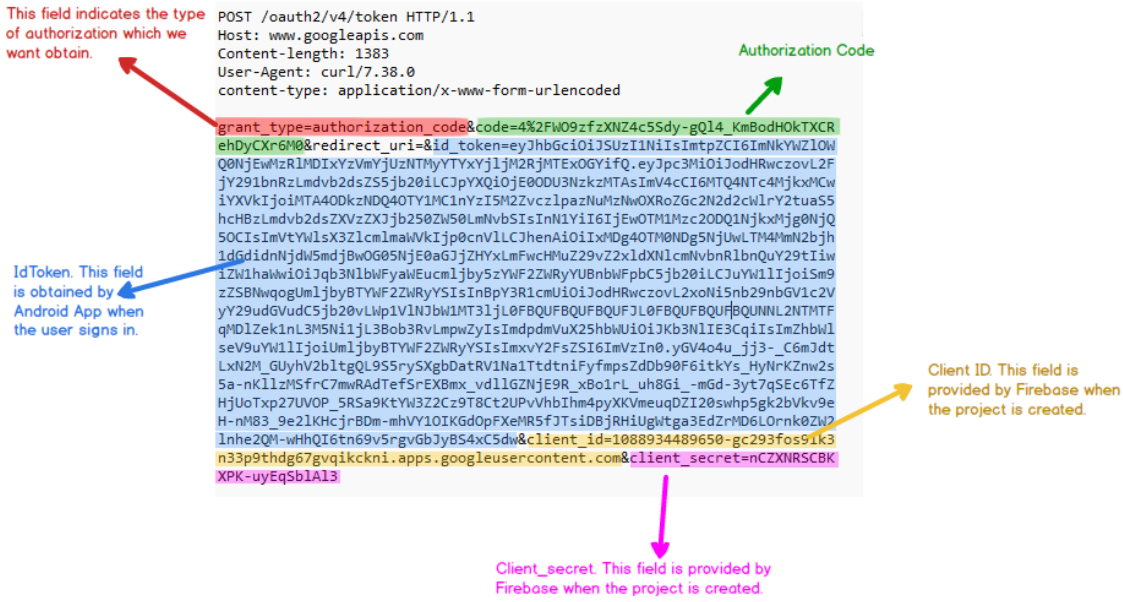

Figure 5: HTTP Request to get the access token

HTTP Response

```
HTTP/1.1 200 OK
Content-length: 266
X-xss-protection: 1; mode=block
X-content-type-options: nosniff
Transfer-encoding: chunked
Expires: Mon, 01 Jan 1990 00:00:00 GMT
Vary: Origin, X-Origin
Server: GSE
-content-encoding: gzip
Pragma: no-cache
Cache-control: no-cache, no-store, max-age=0, must-revalidate
Date: Mon, 30 Jan 2017 16:12:14 GMT
X-frame-options: SAMEORIGIN
Alt-svc: quic=":443"; ma=2592000; v="35,34"
Content-type: application/json; charset=UTF-8
{
  "access_token": "ya29.GlvjA-G0P8WKg_yFyGvgkg4XC7hh_rj5MExV6bdpqNaRCIjK
  OTxoEuBezIM1hZHgyTl6X6elnRorR9wXg-GJqYv3j2EQJMttddK9acf-rbZc82VpcVvOCr
  PXkPSs",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "1/BhGLbExB1NMBUKzYZu95RwPL7AiQNgquhcmEQCYGA34"
}
```

Figure 6: HTTP Response with the access token and the refresh_token

2- This step is performed every time the EoT device sends an image to the GCS.

   a. First, the EoT Device updates the *Access token*. This procedure can be done at any time, even if it has not expired. The EoT device sends an HTTP Request with the *Refresh token* to Google's OAuth 2.0 server and the server responses with a new *Access token*.

HTTP Request

```
POST /oauth2/v4/token HTTP/1.1
Host: www.googleapis.com
Content-length: 163
content-type: application/x-www-form-urlencoded
user-agent: google-oauth-playground
client_secret=nCZXNRSCBKXPK-uyEqSblAl3&grant_type=refresh_token&refresh_
token=1/BhGLbExB1NMBUKzYZu95RwPL7AiQNgquhcmEQCYGA34&client_id=1088934489
650-gc293fos9ik3n33p9thdg67gvqikckni.apps.googleusercontent.com
```

Figure 7: HTTP Request to update the access token

HTTP Response

```
HTTP/1.1 200 OK
Content-length: 199
X-xss-protection: 1; mode=block
X-content-type-options: nosniff
Transfer-encoding: chunked
Expires: Mon, 01 Jan 1990 00:00:00 GMT
Vary: Origin, X-Origin
Server: GSE
-content-encoding: gzip
Pragma: no-cache
Cache-control: no-cache, no-store, max-age=0, must-revalidate
Date: Mon, 30 Jan 2017 16:50:46 GMT
X-frame-options: SAMEORIGIN
Alt-svc: quic=":443"; ma=2592000; v="35,34"
Content-type: application/json; charset=UTF-8
{
  "access_token": "ya29.GlvjA1lpY7Iuy6f-x1sch_NcGnkIVthuuNFCssHe_y3wruL8
  gkjZ_kV1y5HjPXSUg8yvdv1uaCxU547S6zIO-ynBEWDdT3mFnHnvRG3dJTRXKl98Ac9Xwq
  jxQ-2K",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Figure 8: HTTP Request with the updated access token

b. After that, the EoT Device sends an image to the GSC using an HTTP Request. The *Access token* is included in the Authorization header.

HTTP Request

```
POST /upload/storage/v1/b/pushnotification-b8cc9.appspot.com/
o?name=name     HTTP/1.1
Host: www.googleapis.com
Content-length: 84710
Content-type: image/png
Authorization: Bearer ya29.GlvjA1lpY7Iuy6f-x1sch_NcGnkIVthuuNFCssHe_y3wr
uL8gkjZ_kV1y5HjPXSUg8yvdv1uaCxU547S6zIO-ynBEWDdT3mFnHnvRG3dJTRXKl98Ac9Xw
qjxQ-2K

Image
```

Figure 9: HTTP Request to send an image to GSC

3- After the GSC has received the image, the EoT Device sends a push notification to the Android App.
   a. The EoT Device generates a push notification through an HTTP Request and sends it to the FCM server.

Figure 10: HTTP Request to send a push notification using FCM

   b. The FCM server sends the push notification to the Android App.

4- When the Android app receives the push notification, it extracts the necessary information. This information does not contain the image (the maximum size of a push notification with data is 4 kb), it contains the name of the image. This way, the Android app is able to download the defined image from the GCS using the Firebase Cloud Storage library.

2. Achieved requirements

This use case shows that the REQ009 has been developed correctly.

9. CONCLUSIONS

In this deliverable, the EoT Device application of the Flexible Mobile Camera demonstrator has been described. The document collects the functional and non-functional requirements and the planning of the development.

The demonstrator was divided in different modules. Each module has an objective and achieves one or more functional requirements. In addition, the software architecture of the EoT Device application has been explained.

The EoT libraries used to build the Flexible Mobile Camera demonstrator have been indicated.

Finally, the use cases of the EoT Device application have been exposed. Each use case is accompanied by a description and the functional requirements achieved by it.

- End of document -