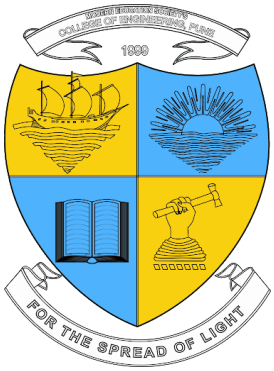
SAVITRIBAI PHULE PUNE UNIVERSITY



Modern Education Society's College of Engineering
19, Late Prin. V.K. Joag Path, Wadia College Campus, Pune- 411001

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

FINAL YEAR PROJECT REPORT ON

'Firmware Update Of IoT Devices Over The Air Interface'

SUBMITTED TOWARDS THE
FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING
(Electronics and Telecommunication)

BY

Piyush Nindane	Exam No: 71818506E
Bhagwat Suryawanshi	Exam No: 71818604E
Shubhada Deshmukh	Exam No: 71312621D

Under The Guidance of

Prof. Nabegha Masood and

Cognito Networks and Galiot Systems Inc

Abstract

Firmware is a specific class of software that provides the low-level control for a device's specific hardware. Firmware can either provide a standardized operating environment for more complex device software (allowing more hardware-independence), or, for less complex devices, act as the device's complete operating system, performing all control, monitoring and data manipulation functions.Firmware is held in non-volatile memory devices such as ROM, EPROM, EEPROM, and Flash memory. Changing the firmware of a device was rarely or never done during its lifetime in the past but is nowadays its a common procedure.Common reasons for updating firmware include fixing bugs or adding features to the device. This requires ROM integrated circuits to be physically replaced, or EPROM or flash memory to be reprogrammed through a special procedure.But these drawbacks can be overcome by using programmable memory and remotely updating the firmware of any physical devices.Hence we have proposed a mechanism to automate the firmware update mechanism using using Over The Air interface(OTA), by a package management and auto-configuration system

Keywords- Firmware,Over The Air, Flash memory, remotely, package management, auto-configuration.

Acknowledgment

It gives us great pleasure and satisfaction in presenting the final project report on 'Firmware Update Of IoT Devices Over The Air Interface'.

We would like to take this opportunity to thank 50 our internal guide ***Prof. Nabegha Masood*** *and external guide* ***Mr.Ajit Sarnaik*** *and* ***Mr.Basavraj Hooli*** *from* ***Cognito netwrok and Galiot systems Inc*** *for giving us all the help and guidance we needed. We are really grateful to them for their kind support. Their valuable suggestions were very helpful.*

We would like to thank ***Mr.Nilesh Raut*** *network admin at MESCOE and all those, who have directly or indirectly helped us for the completion of the work during this seminar.*

Piyush Nindane
Bhagwat Suryawanshi
Shubhada Deshmukh
(B.E. EnTC engg.)

Contents

1 Introduction **1**
- 1.1 Synopsis . . . 1
- 1.2 Goals and Objectives . . . 2

2 Network Architechture **3**
- 2.1 Diagram . . . 3
  - 2.1.1 Process Flow . . . 4

3 Software Requirement Specification **5**
- 3.1 Purpose & Scope of Document . . . 5
- 3.2 Operating System and Runtime environment . . . 5
  - 3.2.1 Technology and framework . . . 5
  - 3.2.2 Assumption & Dependencies . . . 6
- 3.3 Functional Requirements . . . 6
  - 3.3.1 Frontend Subsystem . . . 6
  - 3.3.2 Backend Subsystem . . . 6
  - 3.3.3 Hardware Interfaces . . . 7
  - 3.3.4 Software Interfaces . . . 7
  - 3.3.5 Communication Interfaces . . . 7

4 Package optimization and security **8**
- 4.1 File hashing and message digest . . . 8
- 4.2 Limitations . . . 9
- 4.3 Applications . . . 10

5 Conclusion and future scope **11**

Chapter 1

Introduction

1.1 Synopsis

A firmware is set of files (program) which is written in the flash memory of any physical device, these files define the functionality and behaviour of the device hardware.Any updates or rectification in the operation of existing hardware can be done by upgrading the present firmware of the device by an updated one. The authenticated files are transferred using different modes of transportation over a package release server,by the OEM (original equipment manufacturer).These files are then downloaded by the host device using various methods.Our project is all about remotely updating a smart device by a package management mechanism.In our project we will be using various network protocols and services like **TCP, SFTP, MQTT(*Publish-Subscribe*),REST** .The project involves *Package generator, Package manager, Gateway, End node* configuration and setup.

The entire process of file transfer can be categorized in the following steps:

- Patch generation by the OEM
- Package optimization (Hashing and zipping)
- Publish availability message to the package manager server(MQTT Broker)
- Delivery of the package to package manager
- Unzipping and version control validation
- Publish availability message to the SOC

- Downloading of package unto device
- Setup and installation
- Checksum and integrity test
- Rollback incase of broken package
- Device reboot(if required)

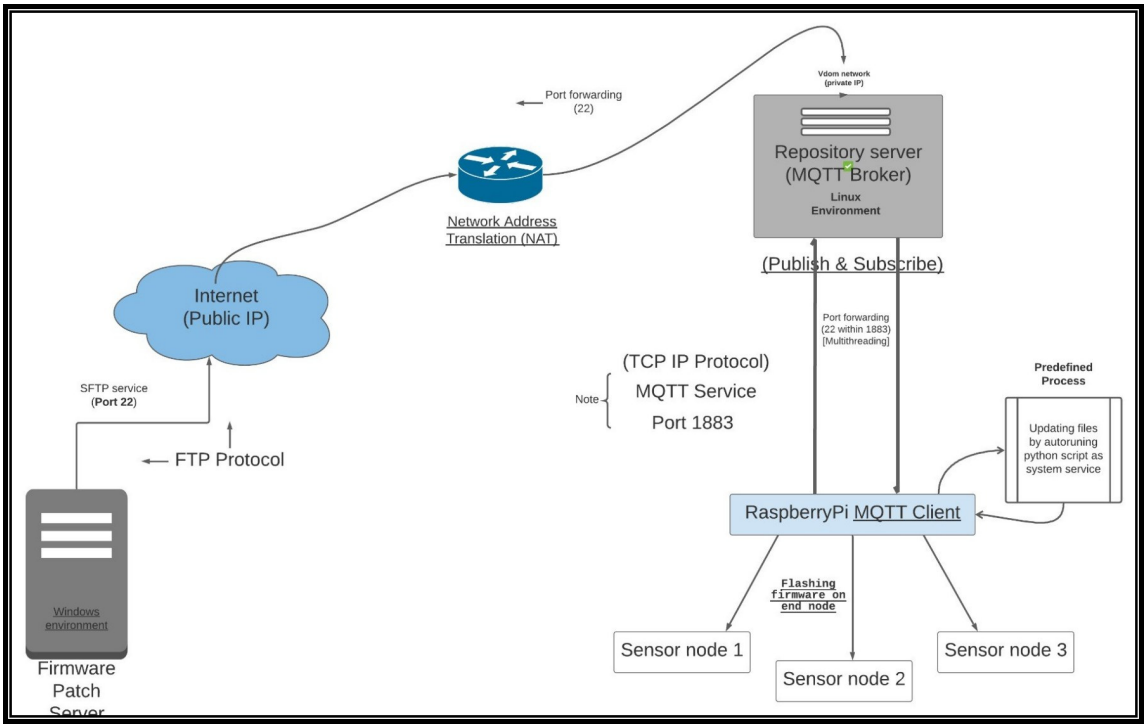
1.2 Goals and Objectives

- To make a package management system for remote firmware update process for smart devices in IoT environment.
- To generate, optimize and release the firmware update patch using **Publish-Subscribe framework**.
- To develop an automated system with least or no human intervention.
- To eliminate the drawbacks faced while updating a smart device and make the process simple,robust, secure and fast.

Chapter 2

Network Architechture

2.1 Diagram



(2.1)

Package Management System [1]

2.1.1 Process Flow

.

The entire system mainly consists of 3 physical machines :
[1] Package generator,
[2] Packager manager server(Repository server) and
[3] Sensor gateway SOC

As per Publish-Subscribe framework and MQTT service, the patch release server and the Raspberry Pi are the MQTTClient and the MQTT broker is running on the repository server.
As the package is delivered to the repository server the broker publishes an availability message to the RPi which in turn checks and compares the hash of the new file.Incase of difference in the files the RPi sends back an acknowledgement message to the repository server.
Once the file is successfully downloaded by the gateway(Raspberry Pi) an SSH command initiates the setup and installation of the files. A checksum test runs after installation of the files. Incase of corrupted files a rollback occurs and the previous version is reinstalled to the device.
The final step of the entire mechanism is device specific wherein the SOC burns the new set of files to the flash memory of the sensor device. A device re-boot is performed if needed at the end of the entire process.

Chapter 3

Software Requirement Specification

3.1 Purpose & Scope of Document

A software requirements specification (SRS) is a document that is created when a detailed description of all aspects of the software to be built must be specified before the project is to commence. It is important to note that a formal SRS is not always written. In fact, there are many instances in which effort expended on a SRS might be better spent in other software engineering activities.

3.2 Operating System and Runtime environment

As the project is in IoT environment all the dependencies and mechanisms have been setup with respect to the UNIX and its applications like the Linux and Debian operating systems.

3.2.1 Technology and framework

Python version 3.9 is used as the scripting technology in the backend and frontend of the project. The Mosquitto server is used as MQTT broker in the mechanism.The MQTT service uses the TCP protocol as transport interface.Mosquitto is a light weight and handy service which listens on port 1883. The Python Paho framework is used for implementing the mqtt client-server connection.

3.2.2 Assumption & Dependencies

We have assumed that the input file uploaded by the patch release server is a binary file.The compression and hashing of this file is further explained in the report. The architecture of this project is fully dependent on Linux and Debian environment and have a dependency on its services and libraries.

The proposed project also aims to develop a User Interface for the OEM user which shall be developed using Python Django framework

3.3 Functional Requirements

3.3.1 Frontend Subsystem

Description

The Frontend Module of our Project is the part which will be visible to the original equipment management user i.e. the User will exclusively interact with this module. The Frontend will be a GUI Web Application developed in an appropriate Django Framework for visual aesthetics. There will be an option to provide input (upload file), this input will be passed to the backend and then backend will perform the necessary computations and operation on it which will be passed back to the frontend. Once the file is finally updated to the end node the frontend shows an acknowledgement messege shared by the end node manager.

Functional Requirements

REQ-1: Django enabled Browser
REQ-2: Internet Connectivity.

3.3.2 Backend Subsystem

Description

In the Backend of our proposed project, we will be using Python(Version 3.9 or greater), Paho and Paramiko for development of the backend system.Python is a programming technology used and Paho library is used for the MQTT service. Where as Paramiko library is used for Secure File Transfer.

Stimulus/Response Sequences

The Backend module will be loaded when the user inputs firmware file and clicks on the upload button. The backend module is not loaded by default and is loaded only when the frontend makes a function call to the backend.

3.3.3 Hardware Interfaces

As the project is Iot specific it needs the integration of end node (sensor node) with the gateway which shall inturn manage and control the sensors.

3.3.4 Software Interfaces

Python 3.9 (MQTT-Paho, Paramiko,Pillow), HTML, CSS, Django Framework

3.3.5 Communication Interfaces

The requirements associated with communications functions required by this project, include any web browser with Django Support for running the model in the backend. Communication standards that will be used is HTTP and SFTP. As there are security issues as the project requires handling of sensitive or real-time data such like the firmware. The input to the system is a binary compressed firmware file & output is the update acknowledgement message.

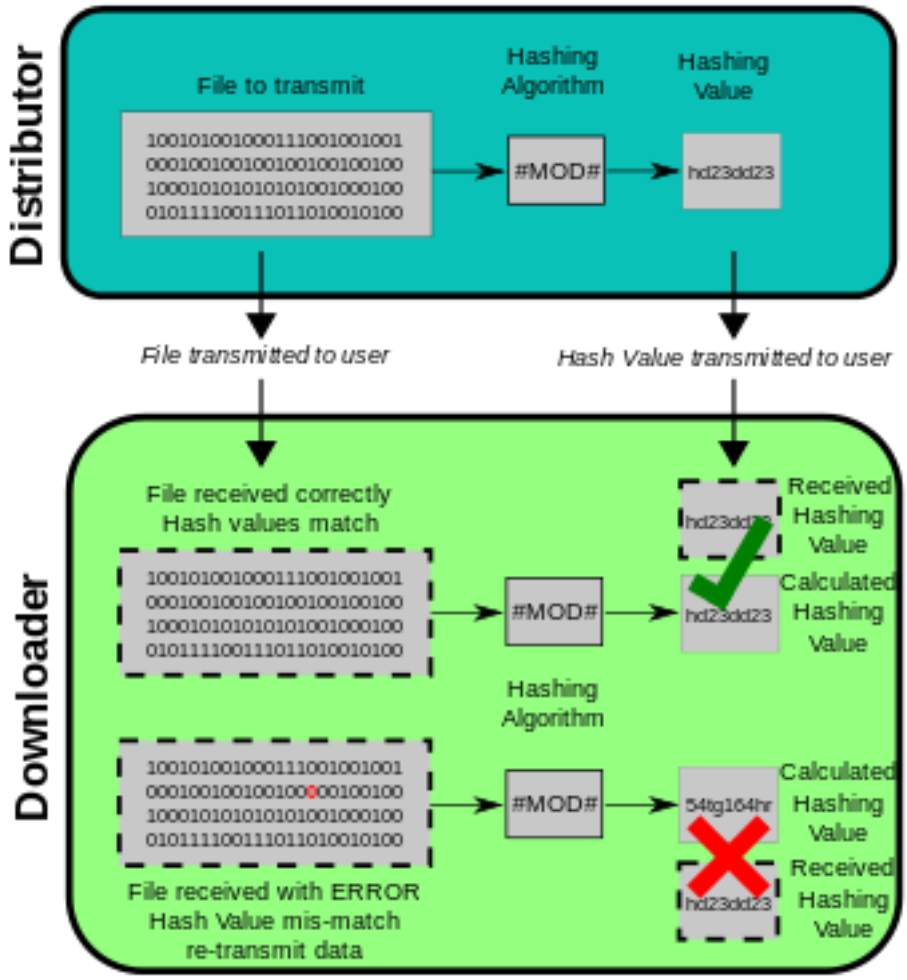
7

Chapter 4

Package optimization and security

4.1 File hashing and message digest

- For proper versioning, validation and vulnerability of firmware the file uses the MD5 hashing algorithm which generates a unique hash code of the file.
- The mechanism is such it adds a time stamp extension to the new file and hence it becomes easy for version control and roll back if needed.
- After hashing the file the executable file is added with the necessary dependencies like the libraries, documents etc.
- The folder is then zipped into an optimized package with all necessary dependencies.
- As per Debian package the .tar zipping extension is used for the optimization.



4.2 Limitations

- The system needs a 128 bit MD5 encryption which can not suitable for all environments .
- The package is zipped only in .tar and is not universal.The untarring of the package is system dependent task.

4.3 Applications

- In a Wireless Sensor Node environment to remotely update any end node.
- Updating the end node manager Gateway or and edge computing device.
- Any smart device can be monitored and maintained without physical access to the device.
- Release now update later.
- Keeping up with security protocols.

Chapter 5

Conclusion and future scope

The package management system is used to provide easy robust and fast update to any smart device at a remote location.
This method shall be fruitfull in terms of cost saving, time efficient and simplify post production techniques
The security parameter is given utmost importance which is good in the case of wireless communication.
The 7 fundamentals of IoT Anything,Anywhere,Anydevice,etc are catered by this mechanism.
The future scope of the project includes improving the automation and self configuration modules of itself.