SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

Generic SDK for zephyr OS to connect AWS IoT cloud

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Computer Engineering)

BY

Tejashree Naphade	Exam No: B120054364
Vinayak Kalunge	Exam No: B120054472
Tejas Sangani	Exam No: B120054418
Shubham Kolhe	Exam No: B120054312

Under The Guidance of

Prof. A.R.Deshpande

DEPARTMENT OF COMPUTER ENGINEERING
Pune Institute of Computer Technology
Sr. No 27, Pune-Satara Road, Behind Bharati Vidyapeeth College, Dhankawadi, Pune, Maharashtra 411043

College Name
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project Entitled

Generic SDK for zephyr OS to connect AWS IoT cloud

Submitted by

Tejashree Naphade	Exam No: B120054364
Vinayak Kalunge	Exam No: B120054472
Tejas Sangani	Exam No: B120054418
Shubham Kolhe	Exam No: B120054312

is a bonafide work carried out by Students under the supervision of Prof. Guide Name and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering) Project.

Prof. A.R. Deshpande	Prof. R.B.Ingle
Internal Guide	H.O.D
Dept. of Computer Engg.	Dept. of Computer Engg.

Abstract

Zephyr OS is a Operating System that is supported by all IoT devices. Zephyr OS supports only C as a programming Language. Best of our knowledge, there is no library or SDK to connect hardware to AWS IoT cloud for zephyr OS. This project is about developing the generic SDK that is library in C language that will connect to AWS IoT cloud, so that it will be easy for any developer across the globe to connect hardware using Zephyr OS to AWS IoT cloud.

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***'Developing a generic SDK to connect hardware to AWS IoT cloud for zephyr OS'****.*

I would like to take this opportunity to thank my internal guide ***Prof. A.R. Deshpande*** *for giving me all the help and guidance I needed. I am really grateful to her for her kind support. Her valuable suggestions were very helpful.*

I am also grateful to ***Prof. Ingle****, Head of Computer Engineering Department, Pune Institute of Computer Technology for his indispensable support, suggestions.*

In the end our special thanks to ***Mr. Abhijit Gadgil*** *for providing various resources such as preliminary reference material, example dataset and for his guidance throughout the tenure of our project.*

Tejashree Naphade
Vinayak Kalunge
Tejas Sangani
Shubham Kolhe
(B.E. Computer Engg.)

Contents

List of Figures

List of Tables

Chapter 1

Synopsis

1.1 Project Title

Developing a generic SDK to connect hardware to AWS IoT cloud for zephyr OS.

1.2 Project Option

Project sponsored by L3 Cube, Pune.

1.3 Internal Guide

Prof. A. R. Deshpande

1.4 Sponsorship and External Guide

Mr. Abhijit Gadgil

1.5 Technical Keywords (As per ACM Keywords)

Please note ACM Keywords can be found : http://www.acm.org/about/class/ccs98-html
Example is given as

1. C. Computer Systems Organization

(a) C.2 COMPUTER-COMMUNICATION NETWORKS
    i. C.2.2 Network Protocols
        A. Client/server
        B. Computer science education
        C. Network communication [Operating Systems]
        D. Network Protocols
        E. Embedded Operating System

1.6 Problem Statement

Developing a generic SDK to connect hardware to AWS IoT cloud for zephyr OS.

1.7 Abstract

Zephyr OS is a Operating System that is supported by all IoT devices. Zephyr OS supports only C as a programming Language. Best of our knowledge, there is no library or SDK to connect hardware to AWS IoT cloud for zephyr OS. This project is about developing the generic SDK that is library in C language that will connect to AWS IoT cloud, so that it will be easy for any developer across the globe to connect hardware using Zephyr OS to AWS IoT cloud.

1.8 Goals and Objectives

- To connect devices using Zephyr OS to AWS cloud.
- To create a generic SDK in C for Zephyr OS.

1.9 Relevant mathematics associated with the Project

System Description:
S : System of Solution

S = {s, e, I, O, Fmain, DD, NDD}

s : start state
: {Empty Set}

e : end state
: SDK is developed

I : Input set
I = {D}, P : Published messages
P = {pi — p : message, $\forall i \in (1, 2, 3, ...N))\}$
$O : Outputset$
$O = \{Q\}, Q : setofmessagesstoredontheAWScloud$
$Q = \{q|q : message, \forall q \exists pi \in P\}$

$Fmain : setofMainFunctions$
$Fmain = \{f1\}$
$f1() : reqbyMQTTprotocol()$

$f1() : X1 \rightarrow Y1$
$X1 = I$
$Y1 = \{1, 0\}$
$1 = MessagestoredonAWScloud$
$0 = MessagenotstoredonAWScloud$

$DD : DeterministicData$
$DD = \{Messagesasinput, Actionperformedonthemessages\}$

$NDD : NonDeterministicData$

NDD ={ Connection path to the AWS cloud, Hop counts }

Su : Success Case Connection to IoT cloud established through Zephyr OS.

Fl : Failure Case Connection to IoT cloud failed.

1.10 Names of Conferences / Journals where papers can be published

- 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT) 18 May - 19 May 2018
- 018 2nd International Conference on Applied Electromagnetic Technology (AEMT) 09 Apr - 12 Apr 2018

1.11 Review of Conference/Journal Papers supporting Project idea

[1] J. Boman, J. Taylor and A. H. Ngu, "Flexible IoT middleware for integration of things and applications," 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Miami, FL, 2014, pp. 481-488

[2] M. Eisenhauer, P. Rosengren and P. Antolin, "A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems,"2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, Rome, 2009, pp. 1-3.

[3] M. Beveridge and P. Koopman, "Jini meets embedded control networking: a case study in portability failure,"Proceedings of the Seventh IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. (WORDS 2002), San Diego, CA, 2002, pp. 11-18.

[4] J. E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel and D. Mosse, "Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes,"2012 Eighth International Conference on Intelligent Environments, Guanajuato, 2012, pp. 206-213.

[5] Hongsen Yu, V. Wijeratne and Zhihong Cai, "An energy-efficient network implemented with Junos SDK,"2013 IEEE 10th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, 2013, pp. 797-800.

[6] V. Maffione, F. Salvestrini, E. Grasa, L. Bergesio and M. Tarzan, "A

software development kit to exploit RINA programmability,"2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, 2016, pp. 1-7.

[7] S. Tayeb, S. Latifi and Y. Kim, "A survey on IoT communication and computation frameworks: An industrial perspective,"2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2017, pp. 1-6.

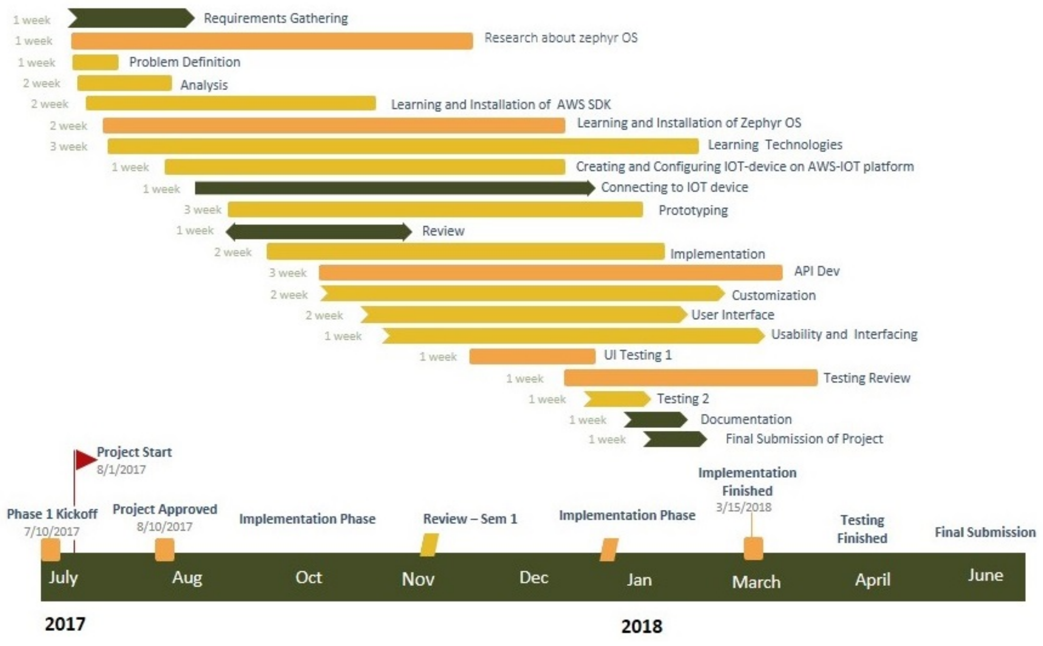
[8] Anthony Rowe, Rahul Mangharam and Raj Rajkumar, "FireFly: A Time Synchronized Real-Time Sensor Networking Platform" 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, 2016, pp. 1-7.

[9] S. Lee, H. Kim, D. k. Hong and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level,"The International Conference on Information Networking 2013 (ICOIN), Bangkok, 2013, pp. 714-717.

[10] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT),"2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 746-751.

[11] J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez and P. Boronat, "Handling mobility in IoT applications using the MQTT protocol,"2015 Internet Technologies and Applications (ITA), Wrexham, 2015, pp. 245-250.

1.12 Plan of Project Execution


1 week Requirements Gathering
1 week Research about zephyr OS
1 week Problem Definition
2 week Analysis
2 week Learning and Installation of AWS SDK
2 week Learning and Installation of Zephyr OS
3 week Learning Technologies
1 week Creating and Configuring IOT-device on AWS-IOT platform
1 week Connecting to IOT device
3 week Prototyping
1 week Review
2 week Implementation
3 week API Dev
2 week Customization
2 week User Interface
1 week Usability and Interfacing
1 week UI Testing 1
1 week Testing Review
1 week Testing 2
1 week Documentation
1 week Final Submission of Project
Project Start 8/1/2017
Phase 1 Kickoff 7/10/2017
Project Approved 8/10/2017
Implementation Phase
Review – Sem 1
Implementation Phase
Implementation Finished 3/15/2018
Testing Finished
Final Submission
July
Aug
Oct
Nov
Dec
Jan
March
April
June
2017
2018

Chapter 2

Technical Keywords

2.1 Area of Project

Project Area

2.2 Technical Keywords

1. C.2 - COMPUTER-COMMUNICATION NETWORKS.
   (a) Internet Of Things
   (b) Operating System
   (c) IoT cloud
   (d) Zephyr OS
   (e) AWS

Chapter 3

Introduction

3.1 Project Idea

- Developing the generic SDK that is library in C language that will connect hardware to AWS IoT cloud, so that it will be easy for any developer across the globe to connect hardware running Zephyr OS to AWS IoT cloud.

3.2 Motivation of the Project

- Different hardware developers create their own OS and SDK to connect to AWS IoT cloud. To make things generic Linux has started the Zephyr Project. Zephyr OS is generic RTOS which supports most of the embedded system hardware. To communicate and develop application for these devices with IoT clouds we need a generic SDK. As per our Knowledge there is no library or SDK to connect hardware running Zephyr OS to AWS IoT cloud.

3.3 Literature Survey

- R. Matei and A. Radovici,"Remote management system for embedded devices: Wyliodrin,"2016 15th RoEduNet Conference: Networking in Education and Research, Bucharest, 2016, pp. 1-5.

  Keywords: Internet of Things; cloud computing; embedded systems; mobile computing; Wyliodrin platform; remote management system

  Review : This paper presents a lightweight robust and reliable implementation of the device-side software platform to program embedded devices. This is achieved by allowing the user to control and program its devices through the web and by providing the means to monitor and interact with the environment. This paper helps us in understanding how to make the embedded technologies more accessible and how to securely establish connection between embedded device and cloud.

- M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 746-751. doi: 10.1109/CSNT.2015.16

  Keywords: Internet of Things; access protocols; telecommunication security; CoAP; MQTT-SN; communication protocols; constrained access protocol; device to device communications; elliptic curve cryptography; information communication technology; message queue telemetry transport.

  Review : Innovations in digital things and information communication technology are driving deployment of Internet of Things. Device to Device communications in IoT is envisaged through various protocols such as Constrained Access Protocol (CoAP), Message Queue Telemetry Transport (MQTT) and MQTT-SN (or Sensor Networks). But these protocols are devoid of security features. This paper proposed a secure version of MQTT and MQTT-SN protocols (SMQTT and SMQTT-SN) in which security feature is augmented based on Key/Cipher text Policy-Attribute Based Encryption using lightweight Elliptic Curve Cryptography. Feasibility of CP/KP-ABE to enable communication security for IoT devices based on Pub-Sub architecture is analyzed. This will help us to use secured version of communication protocols to get information more reliably.

- S. Tayeb, S. Latifi and Y. Kim,"A survey on IoT communication and computation frameworks: An industrial perspective," 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2017, pp. 1-6.

  Keywords : Internet of Things; cloud computing; embedded systems; IoT communication frameworks; IoT computation frameworks; IoT sensor networks ; embedded systems ; Smart Objects

  Review : In this paper, many concepts around IoT architectures are discussed. How modularity and security can be addressed with an array of various models and frameworks are discussed. This paper will help us to understand how we can encompass different modules and security.

Chapter 4

Problem Definition and scope

4.1 Problem Statement

Developing a generic SDK to connect hardware to AWS IoT cloud for zephyr OS.

4.1.1 Goals and objectives

Goal and Objectives:

- In order to efficiently send data from embedded devices to IoT cloud and vice-versa we are developing a generic SDK in C language. It will connect embedded devices to IoT cloud using MQTT protocol.

4.1.2 Statement of scope

- The feasibility of our problem statement is determined by classifying it as a P Complete problem.
- The proposed implementation consists of cloud services. The basis of working is to connect to IoT cloud and perform operations. Basic operations are connect, get data, and put data. Complexity will depend on these operations.
- This operation can be performed in constant time complexity O(1) and O(n) at worst case. Thus, both our problems can be solved in Polynomial time. Hence, our problem belongs to class P .

4.2 Software context

- The business or product line context or application of the software is to be given

4.3 Major Constraints

- The SDK supports Zephyr OS supported embedded devices only.

4.4 Methodologies of Problem solving and efficiency issues

- State of object is created when cloud connectivity function is called. Objects are stored in JSON File format and are parsed from it. Then, it is serialized in byte code. Connection to TLS is established by MQTT protocol using through specific port. If TLS connection is established successfully, the device is partially connected to cloud as verification is required.Then, certificates and subscription of the user will be checked. If they are present, device will be connected to cloud else the connection will be terminated. When the connection is established, output will be given from cloud to device as requested.

4.5 Scenario in which multi-core, Embedded and Distributed Computing used

We are using embedded IoT chip in our project which contains single core. Hence, in our project, there will be no multi-core or distributed computing but embedded computing will be used in the project when the operations for sending data to cloud and getting data from cloud will be called.

4.6 Outcome

- A generic SDK that will be used in Zephyr OS supported embedded systems to connect to the AWS Iot Cloud and hence be able to use different cloud services.

4.7 Applications

- Nowadays almost all kinds of devices are connected to the internet. The Embedded devices however have limited resources like less memory. Hence to support the use of various services on these devices, they are connected to the cloud.
- The embedded devices can update the state of the appliance in which it is used. This state can be stored on the cloud which is accessible from anywhere using internet.

4.8 Hardware Resources Required

Sr. No.	Parameter	Minimum Requirement
1	CPU Speed	2 GHz
2	RAM	3 GB

Table 4.1: Hardware Requirements

4.9 Software Resources Required

Platform :

1. Operating System: Linux
2. Qemu machine emulator
3. Programming Language: C
4. Zephyr OS on the embedded device

Chapter 5

Project Plan

5.1 Project Estimates

In project management (e.g., for engineering), accurate estimates are the basis of sound project planning. Many processes have been developed to aid engineers in making accurate estimates, such as Analogy based estimation, Compartmentalization (i.e., breakdown of tasks), Cost estimate, Documenting estimation results, Educated assumptions, Estimating each task, Examining historical data, Identifying dependencies, Parametric estimating, Risk assessment, Structured planning.

5.1.1 Reconciled Estimates

The completed project may significantly differ from the planned tasks and projected conditions upon which the initial estimate was based. The initial estimate must be adjusted to account for these differences if a meaningful comparison between the estimate and the actual project effort is be established. The purpose of the Reconciliation is to recalculate the estimated effort using the actual statistics and results from the completed project. The Reconciliation gathers actual project data through a question and answer process similar to that used when the system requirements were gathered for the initial estimate. The questions differ only in that the past tense is used: "Did the application replace a mission critical or line of business process?quot; instead of quot;Does the application replace a mission critical or line of business process?"

5.1.1.1 Cost Estimate

Cost estimate of the project is estimated budget required for the project. In our project, the device needed for implementation is one embedded chip called Samsung Exynos I T200 that is used for IoT applications. It costs USD 30. Also, the membership cost of AWS is USD 9.99. Hence the complete project cost USD 40.

5.1.1.2 Time Estimates

Time estimate of the project is estimated time required to complete the project. Hence, according to the planning of our project, estimated time required is 9 months.

5.1.2 Project Resources

The project is developed by 4 people. The hardware required for the project is embedded IoT chip Samsung Exynos I T200 on which zephyr OS will be installed. Softwares required for the project are qemu emulator, Zephyr OS, gcc compiler. Also, AWS membership and certifications are required.

5.2 Risk Management w.r.t. NP Hard analysis

This section discusses Project risks and the approach to managing them.

5.2.1 Risk Identification

For risks identification, review of scope document, requirements specifications and schedule is done. Answers to questionnaire revealed some risks. Each risk is categorized as per the categories mentioned in [?]. Please refer table 5.1 for all the risks. You can refereed following risk identification questionnaire.

1. Have top software and customer managers formally committed to support the project? The internal as well as the external guides have approved the project and are ready to support the project in every possible way. A discussion between the team and managers took place to come down to this decision.

2. Are end-users enthusiastically committed to the project and the system/product to be built? MQTT protocol validation is a vital step in

order to enhance performance in later stages for establishing a connection with cloud. The system built will enhance the performance and the entire system will be improved.

3. Are requirements fully understood by the software engineering team and its customers? Requirement engineering was a crucial step in the early stages. The team members and the guide had 2 meetings in order to understand and evaluate the feasibility of all the requirements listed by the customer.

4. Have customers been involved fully in the definition of requirements? As mentioned above the manager and the customer along with the software engineering team had 2 meeting held wherein all the requirements were discussed. The requirements have been defined to the very precise level by the customers.

5. Do end-users have realistic expectations? A check on feasibility of all requirements were made and the requirements which may lead to infinities were told to check upon and come up with necessary amendments.

6. Does the software engineering team have the right mix of skills? The team consists of enthusiasts and fast learners with all the necessary qualities as a software engineering team. The documentation of the projects is kept up to date in order to avoid any loss of crucial data.

7. Is the number of people on the project team adequate to do the job? Currently there are 4 members in the software engineering team and it seems to pretty much suffice the requirements.

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk.

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	User Requirement	Medium	Medium	Medium	Medium
2	System architecture	Low	High	Low	Medium
3	Performance	Low	Low	High	Medium

Table 5.1: Risk Table

Probability	Value	Description
High	Probability of occurrence is	$> 75\%$
Medium	Probability of occurrence is	$26 - 75\%$
Low	Probability of occurrence is	$< 25\%$

Table 5.2: Risk Probability definitions [?]

Impact	Value	Description
High	$> 10\%$	Schedule impact or Unacceptable quality
Medium	$5 - 10\%$	Schedule impact or Some parts of the project have low quality
Low	$< 5\%$	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

Table 5.3: Risk Impact definitions [?]

Risk ID	1
Risk Description	Software requirements capture all user needs with respect to the software system features, functions, and quality of service.
Category	Requirements.
Source	Software requirement Specification document.
Probability	Medium
Impact	Medium
Response	Mitigate
Strategy	Identifying the problems which are not solvable and denying it acceptance.
Risk Status	Identified

Risk ID	2
Risk Description	System architecture
Category	Design and Development
Source	Software Design Specification documentation review.
Probability	Low
Impact	Medium
Response	Mitigate
Strategy	Selecting a proper software design model will solve the issue.
Risk Status	Identified

Risk ID	3
Risk Description	Performance
Category	System Quality
Source	This was considered as a potential risks at the earlier stages.
Probability	Low
Impact	Medium
Response	Mitigate
Strategy	Achieving maximum accuracy
Risk Status	Predicted

5.3 Project Schedule

5.3.1 Project task set

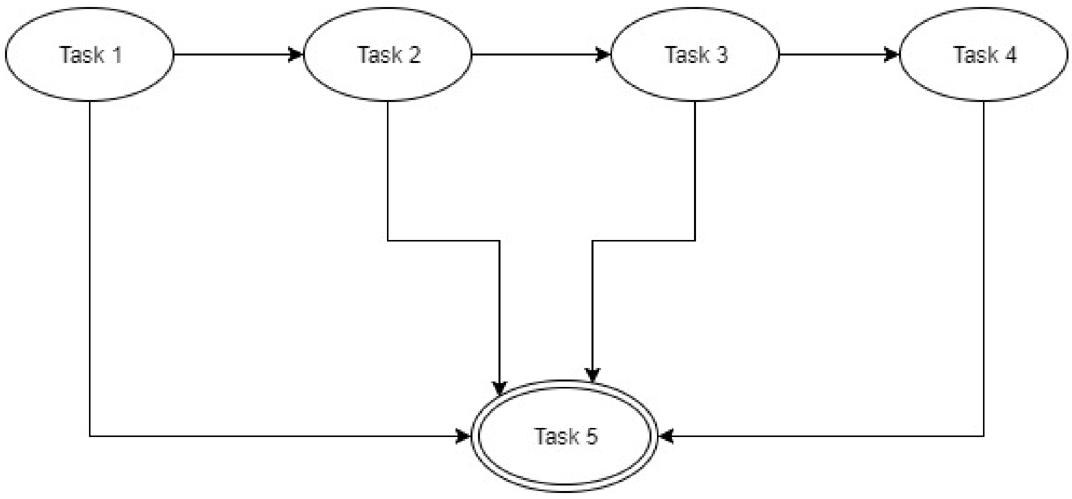
Major Tasks in the Project stages are:

- Task 1: Implementing basic server client model using MQTT protocol which would work on virtual machine through QEMU.
- Task 2: Connection to TLS is established by MQTT protocol using through specific port.
- Task 3: Certificates and subscription of the user will be checked.
- Task 4: Generate the files for the different Module mentioned in module diagram.
- Task 5: Compiling all modules of the project together.

5.3.2 Task network

Project tasks and their dependencies are noted in this diagrammatic form.

5.3.3 Timeline Chart



5.4 Team Organization

Tasks to be done are divided among team members.

5.4.1 Team structure

Connection to TLS using MQTT protocol and checking for Certificates and subscription of the user, the two major modules of the project, are being worked upon in pairs of group members.

5.4.2 Management reporting and communication

Work done in pairs is collaborated later and weekly reporting is done to internal and external guides about the progress of the project till date.

Chapter 6

Software requirement specification (SRS is to be prepared using relevant mathematics derived and software engg. Indicators in Annex A and B)

6.1 Introduction

6.1.1 Purpose and Scope of Document

The purpose of this SRS is to describe the expected behaviour of our system. It covers functional requirements and non-functional requirements of the system. Also it includes various UML diagrams describing the system like Usecase Diagram, Activity Diagram, Dataflow Diagram and State Diagram.

6.1.2 Overview of responsibilities of Developer

Developer is responsible for :

- Creating the SDK
- Testing the SDK
- Maintaining the SDK

6.2 Usage Scenario

6.2.1 User profiles

The system can be said to have two user categories :

1. Developer : A person responsible to create the SDK to connect to the AWS cloud.
2. User : A person using the software developed.

6.2.2 Use-cases

Sr No.	Use Case	Description	Actors	Assumptions
1	User working	User interacting with the device which is using the SDK, expecting output	User	User knows how the SDK works
2	Developer working	Developer modifying the system	Developer	Developer knows what changes are to be made

Table 6.1: Use Cases

6.2.3 Use Case View

Use Case Diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. The specifications also described use case diagram as a specialization of a class diagram, and class diagram is a structure diagram. Use case diagrams are in fact twofold -

they are both behavior diagrams, because they describe behavior of the system, and they are also structure diagrams- as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations. Example is given below

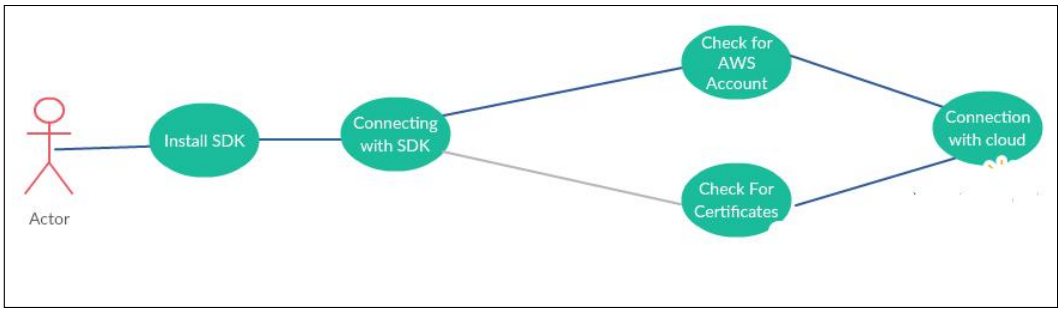


Figure 6.1: Use case diagram

6.3 Data Model and Description

6.3.1 Data Description

Data objects for one use case include a user request for the connection of AWS cloud, an after authentication with the AWS cloud and checking for legal certificates the user will get the output i.e there is connection between user Operating system and AWS cloud.

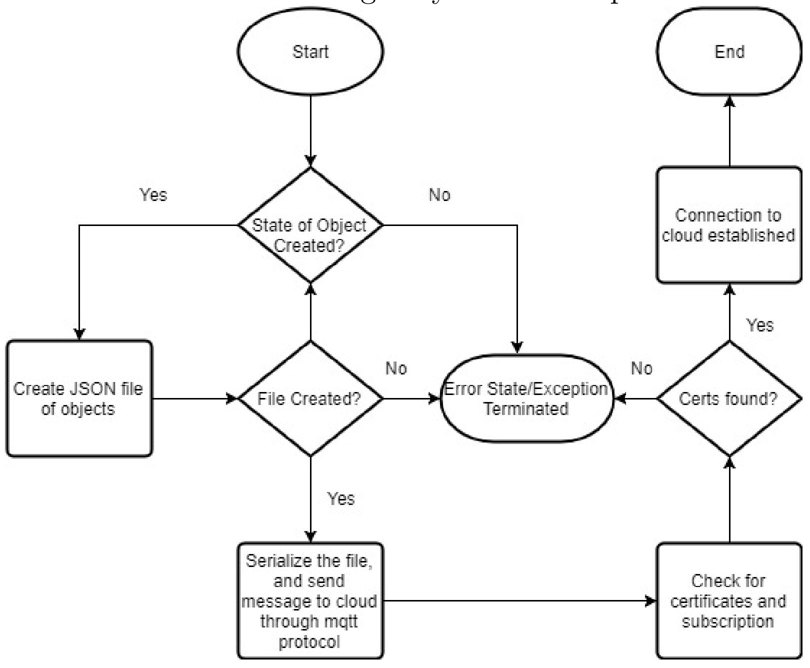
6.3.2 Data objects and Relationships

Objects are serially dependent on each other, the request sent by the user for connection with AWS cloud goes through many files and after sequentially applying methods on it to finally the user will be able to get connection with AWS cloud.

6.4 Functional Model and Description

6.4.1 Data Flow Diagram

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows. This allows anyone to view the flowchart and logically follow the process from beginning to end.


Start
End
Yes
No
State of Object Created?
Connection to cloud established
Yes
Create JSON file of objects
File Created?
No
Error State/Exception Terminated
No
Certs found?
Yes
Serialize the file, and send message to cloud through mqtt protocol
Check for certificates and subscription

6.4.2 Description of functions

State : It is the state of the device.

MQTT : The Device SDK provides functionality to create and maintain a mutually authenticated TLS connection over which it runs MQTT.

Thing Shadow : Thing Shadow:- The Device SDK implements the specific protocol for Thing Shadows to retrieve, update and delete Thing Shadows adhering to the protocol that is implemented to ensure correct versioning and support for client tokens.

Serialize : To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object.

Certificates and Subscription : It will check if the device using AWS IoT cloud has Subscription and Certificates to connect the cloud.

6.4.3 Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control ow is drawn from one operation to another. This ow can be sequential, branched, or concurrent. Activity diagrams deal with all type of ow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system.

- The Activity diagram represents the steps taken.

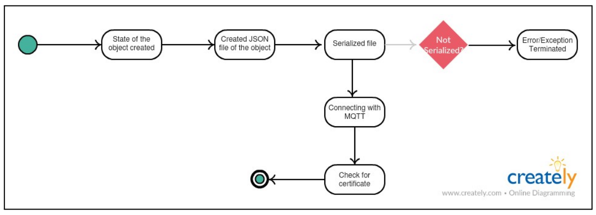

State of the object created
Created JSON file of the object
Serialized file
Not Serialized?
Error/Exception Terminated
Connecting with MQTT
Check for certificate
creately
www.creately.com • Online Diagramming

Figure 6.2: Activity diagram

6.4.4 Non Functional Requirements:

- Performance Requirements - The system must efficiently perform the task required and get user connected with AWS cloud.
- Software quality attributes -
    1. Availability : The system must be accessible all the time.
    2. Modifiability : The system must be flexible to be changed by the developer.
    3. Reusability : The system can be used as a module in other tasks.
    4. Testability : The system must be testable.

6.4.5 State Diagram:

State Transition Diagram
Fig.6.3 example shows the state transition diagram of our system. The states are represented in circles, starting from start state and ending in an end state, and state of system gets changed when certain events occur. The transitions

from one state to the other are represented by arrows. The Figure shows important states and events.

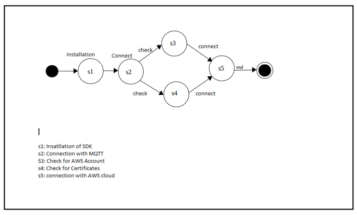


Figure 6.3: State transition diagram

6.4.6 Design Constraints

The system efficiently works on Embedded devices on which Zephyr OS is present.

6.4.7 Software Interface Description

There is no Interface created for our SDK. The SDK works in the background of the Operating System.

Chapter 7

Detailed Design Document using Appendix A and B

7.1 Introduction

This document specifies the design that is used to solve the problem of Product.It includes mainly the architecture diagram and the class diagram.

7.2 Architectural Design

The architectural diagram is as shown in the Figure 7.1.

7.3 Data design (using Appendices A and B)

7.3.1 Internal software data structure

Messages from the device and the cloud are passed to each other based on the topic subscribed by the device.

7.3.2 Global data structure

The messages sent to and fro are the major data in the software.

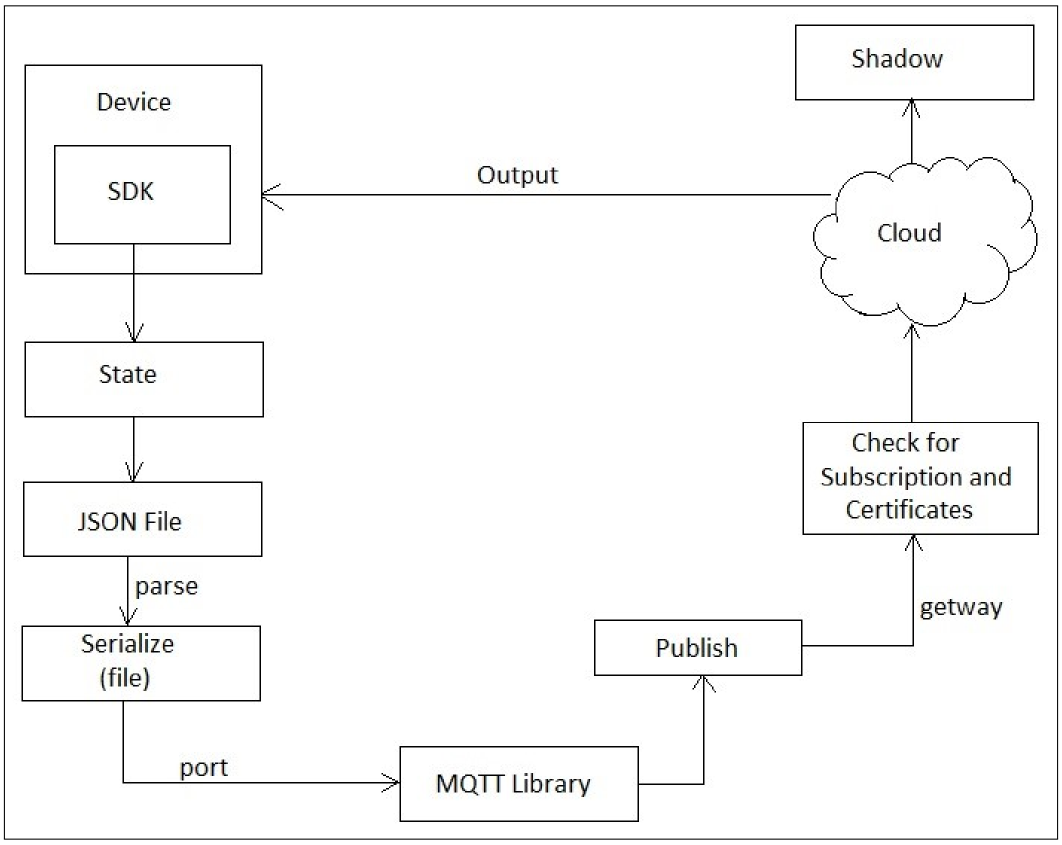


Figure 7.1: Architecture diagram

7.3.3 Temporary data structure

No temporary data is created as such. A traffic less connection has to be created between the devices using Zephyr OS and the AWS Cloud.

7.3.4 Database description

The database contains the information about the attributes of the devices, their subscribed topics and the messages sent and received from them.

7.4 Component Design

7.4.1 Class Diagram

Figure 7.2 represents the class diagram for the system. It contains the Binarization class which contains different attributes and functions (representing different operations required) to get a binarized image. The Bounding box class is required to get a bounding box around different components of the document. These classes are in turn inherited by the Classification class which uses the functionalities of Binarization class as well as Bounding box class to classify the components to text or non-text.

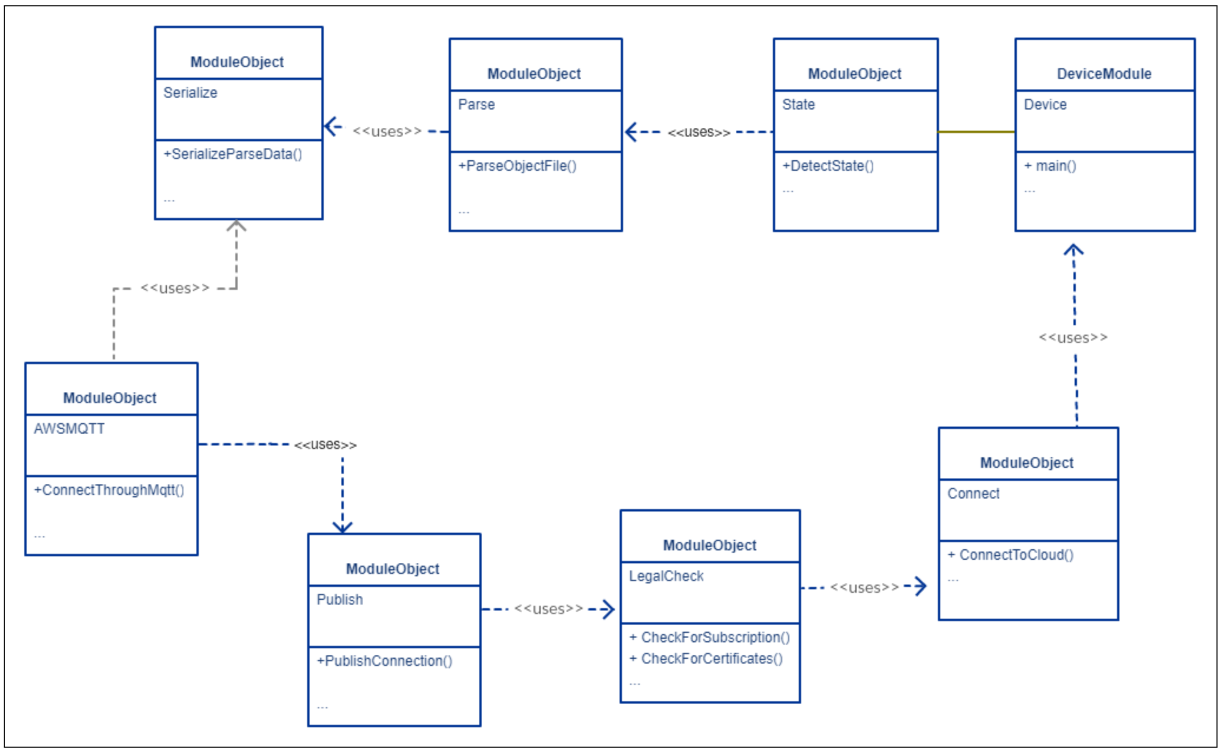


Figure 7.2: Class Diagram

Chapter 8

Summary and Conclusion

- From this project we have successfully created a generic SDK for Zephyr OS which is supported by a variety of hardware, to AWS IoT cloud.

Annexure A

Laboratory assignments on Project Analysis of Algorithmic Design

- To develop the problem under consideration and justify feasibilty using concepts of knowledge canvas and IDEA Matrix.
  Refer [?] for IDEA Matrix and Knowledge canvas model. Case studies are given in this book. IDEA Matrix is represented in the following form. Knowledge canvas represents about identification of opportunity for product. Feasibility is represented w.r.t. business perspective.

I	D	E	A
Increase	Drive	Educate	Accelerate
Improve	Deliver	Evaluate	Associate
Ignore	Decrease	Eliminate	Avoid

Table A.1: IDEA Matrix

- Project problem statement feasibility assessment using NP-Hard, NP-Complete or satisfy ability issues using modern algebra and/or relevant mathematical models.
- input x,output y, y=f(x)

Annexure B

Laboratory assignments on Project Quality and Reliability Testing of Project Design

It should include assignments such as

- Use of divide and conquer strategies to exploit distributed/parallel/concurrent processing of the above to identify object, morphisms, overloading in functions (if any), and functional relations and any other dependencies (as per requirements). It can include Venn diagram, state diagram, function relations, i/o relations; use this to derive objects, morphism, overloading

- Use of above to draw functional dependency graphs and relevant Software modeling methods, techniques including UML diagrams or other necessities using appropriate tools.

- Testing of project problem statement using generated test data (using mathematical models, GUI, Function testing principles, if any) selection and appropriate use of testing tools, testing of UML diagram's reliability. Write also test cases [Black box testing] for each identified functions. You can use Mathematica or equivalent open source tool for generating test data.

- Additional assignments by the guide. If project type as Entreprenaur, Refer [?],[?],[?], [?]

Annexure C

Project Planner

Using planner or alike project management tool.

Annexure D

Reviewers Comments of Paper Submitted

(At-least one technical paper must be submitted in Term-I on the project design in the conferences/workshops in IITs, Central Universities or UoP Conferences or equivalent International Conferences Sponsored by IEEE/ACM)

1. Paper Title:
2. Name of the Conference/Journal where paper submitted :
3. Paper accepted/rejected :
4. Review comments by reviewer :
5. Corrective actions if any :

Annexure E

Plagiarism Report

Plagiarism report