# Table of Contents

[Table of Contents 1](#_Toc423444408)

[Table of Figures 3](#_Toc423444409)

[1. Introduction 4](#_Toc423444410)

[1.1. Concept 4](#_Toc423444411)

[1.2. Background 4](#_Toc423444412)

[1.3. Problem Definition 4](#_Toc423444413)

[1.4. Suggested Solution 4](#_Toc423444414)

[1.5. Objective 5](#_Toc423444415)

[1.6. Plan 5](#_Toc423444416)

[2. Feasibility Study and Analysis 6](#_Toc423444417)

[2.1. Feasibility Study 6](#_Toc423444418)

[2.1.1. Quora 6](#_Toc423444419)

[2.1.2. Study 9](#_Toc423444420)

[2.2. Analysis 11](#_Toc423444421)

[2.2.1. Hardware: Raspberry Pi 11](#_Toc423444422)

[2.2.2. Existing Products 12](#_Toc423444423)

[3. Requirements and Design 13](#_Toc423444424)

[3.1. Introduction 13](#_Toc423444425)

[3.1.1. Product Overview 13](#_Toc423444426)

[3.2. Specific Requirements 13](#_Toc423444427)

[3.2.1. External Interface Requirements 13](#_Toc423444428)

[3.2.2. Software product features 14](#_Toc423444429)

[3.2.3. Software System Attributes 16](#_Toc423444430)

[3.2.4. Database Requirements 17](#_Toc423444431)

[3.3. Architectural Design 18](#_Toc423444432)

[3.3.1. Introduction 18](#_Toc423444433)

[3.3.2. Architectural Representation 18](#_Toc423444434)

[3.3.3. Architectural Goals and Constraints 18](#_Toc423444435)

[3.3.4. Use-Case View 19](#_Toc423444436)

[3.3.5. Logical View 20](#_Toc423444437)

[3.3.6. Process View 21](#_Toc423444438)

[3.3.7. Module Decomposition View 22](#_Toc423444439)

[3.3.8. Data View 22](#_Toc423444440)

[3.4. Conceptual Design 23](#_Toc423444441)

[3.4.1. Experiential Flowchart 23](#_Toc423444442)

[3.4.2. Interface Mockups 23](#_Toc423444443)

[3.4.3. Implementation Plan 24](#_Toc423444444)

[4. Implementation 25](#_Toc423444445)

[5. Testing and Results 26](#_Toc423444446)

[I. Glossary 27](#_Toc423444447)

# Table of Figures

[Figure ‎2.1. Question Asked on Quora 6](#_Toc423441295)

[Figure ‎2.2. Answer by Jim Reich on Quora 7](#_Toc423441296)

[Figure ‎2.3. Answer by Zhou Mingchun on Quora 8](#_Toc423441297)

[Figure ‎2.4. Answer by Rushik Upadhyay on Quora 9](#_Toc423441298)

[Figure ‎2.5. Mobile Connected Devices Study 10](#_Toc423441299)

[Figure ‎2.6. Raspberry Pi 11](#_Toc423441300)

[Figure ‎2.7. Xively Logo 12](file:///C:\Graduation%20Project\Documentation\Project%20Documentation.docx#_Toc423441301)

[Figure ‎2.8. Particle Logo 12](file:///C:\Graduation%20Project\Documentation\Project%20Documentation.docx#_Toc423441302)

[Figure ‎3.1. Use Case View 19](#_Toc423441303)

[Figure ‎3.2. IOTA Logical View 20](#_Toc423441304)

[Figure ‎3.3. IOTA Process View 21](#_Toc423441305)

[Figure ‎3.4. IOTA Module Decomposition View 22](#_Toc423441306)

[Figure ‎3.5. IOTA Data View 22](#_Toc423441307)

[Figure ‎3.6. Implementation Plan Timeline 24](#_Toc423441308)

# Introduction

## Concept

Internet of things is a new concept that the industry is moving towards. It’s a result of the maturity of embedded, mobile and handheld devices. Embedded devices now have developed a lot, its development became much easier and the range of applications became very wide. Its capabilities have grown, now you can connect an embedded board to internet easily so embedded devices became “ONLINE”. On the other hand, cloud devices have developed a lot there is now a lot of tools that can handle storing and communicating data also data science developed so now data became the oil every one searching for because processing data using artificial intelligence, NLP, Data science, … etc. have reached a place where you can extract useful information that can be a milestone to building more useful products, so now comes the missing link connecting embedded devices to the internet so we can have full use out of it INTERNET OF THINGS.

## Background

Internet of things is a new concept that the industry moving towards it’s a result of the maturity of embedded, mobile and handheld devices. embedded devices now have developed a lot its development became much easier and the range of applications became very wide its capabilities have grown now you can connect an embedded board to internet easily so embedded devices became “Online” on the other hand cloud devices have developed a lot there is now a lot of tools that can handle storing and communicating data also data science developed so now data became the oil every one searching for because processing data using artificial intelligence, NLP, Data science, … etc. have reached a place where you can extract useful information that can be a milestone to building more useful products, so now comes the missing link connecting embedded devices to the internet so we can have full use out of it.

## Problem Definition

The problem mainly is there is not enough Internet of Things application in the market right now , this is mainly due to three reasons first the low resources of embedded devices that doesn't allow sophisticated apps running on embedded devices , Second the lack of Internet of things developers this is because IOT Requires knowledge and experience for different domains from embedded and low level knowledge to different routing protocols to server side programming , the Third reason is the difficulty of handling big number of connections and securing this connection.

## Suggested Solution

Our goal is to build a platform for developers seeking to build apps using the internet of things concept, so that it will become easier. Imagine a developer who wants to build some sensors that will acquire some data then show it to the user he needs to work with some low level and embedded code on the board he is using then he needs to start building some form of backend that will receive the data and store it and he needs to do it in an efficient and secure way and then he needs to start building a web or mobile interface the user can interact with and all of that requires different knowledge and expertise to do so our goal is to build a platform that will abstract a lot for him and let him focus on the functionality of his project without worrying about scalability , Security and reliability.

## Objective

Our goal is to build a platform for developers seeking to build apps using the internet of things concept, so that it will become easier. Imagine a developer who wants to build some sensors that will acquire some data then show it to the user he needs to work with some low level and embedded code on the board he is using then he needs to start building some form of backend that will receive the data and store it and he needs to do it in an efficient and secure way and then he needs to start building a web or mobile interface the user can interact with and all of that requires different knowledge and expertise to do so our goal is to build a platform that will abstract a lot for him and let him do the minimum amount of work .

## Plan

First, there's going to be a web IDE that developers can write server side code that they want to run when data is sent and we are searching into being able also to flash code remotely although this would need special hardware to handle and writing libraries and language wrappers to be able to communicate with the device.

Second, we are going to build a REST API to handle receiving and providing data so that the developers can use it to access information related to users from our servers.

Third, a library to be used on the embedded device itself to ensure a connection to the server with a secure and efficient way with as little footprint as possible with minimum resources consumption.

# Feasibility Study and Analysis

## Feasibility Study

### Quora

Quora is a question-and-answer website where questions are asked, answered, edited and organized by its community of users. We went on the website and asked a panel of experts about our project and what they would like to see in it that doesn’t exist in any other known platforms as shown in the figure below.

Figure ‎2.1. Question Asked on Quora

Below are some of the answers by people who are experienced in the field.

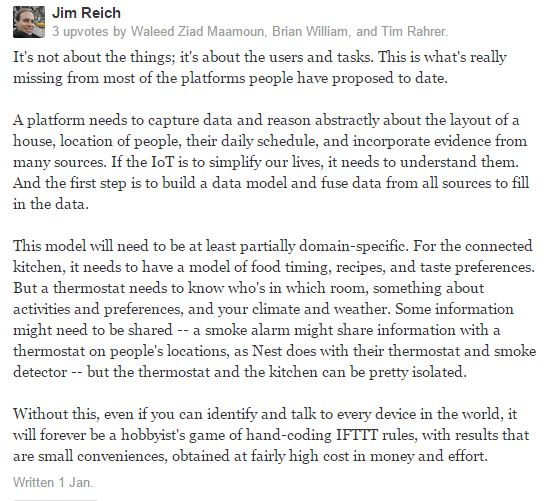


Figure ‎2.2. Answer by Jim Reich on Quora

- Jim Reich

Researcher at Carnegie Mellon University



Figure ‎2.3. Answer by Zhou Mingchun on Quora

- Zhou Mingchun

IOT Expert at Samsung Electronics

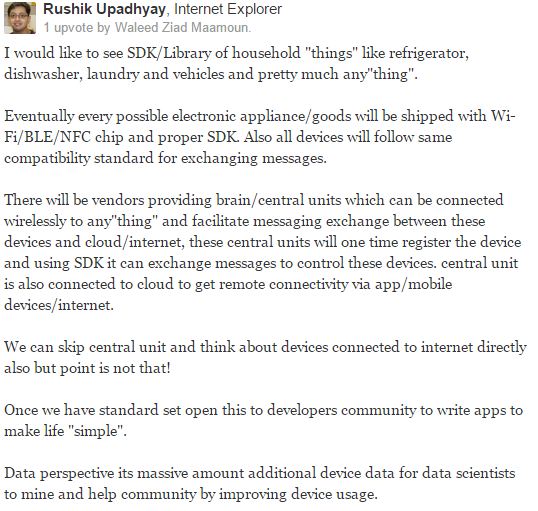


Figure ‎2.4. Answer by Rushik Upadhyay on Quora

- Rushik Upadhyay

PayPal

### Study

In 2011 there was estimated a number of 9 billion total connected devices to the internet, 6 billion of those were mobile connected devices. In 2020 those numbers are estimated to be 24 billion total connected devices, 12 billion of which are mobile connected devices which makes the revenue opportunity for mobile network operators in 2020 $1.2 trillion which is a 7 times increase on 2011 expected revenue. Also the revenue opportunity for connected devices in vertical sectors such as automotive, health, computer electronics and utilities are $202 billion, $69 billion, $445 billion and $36 billion respectively.

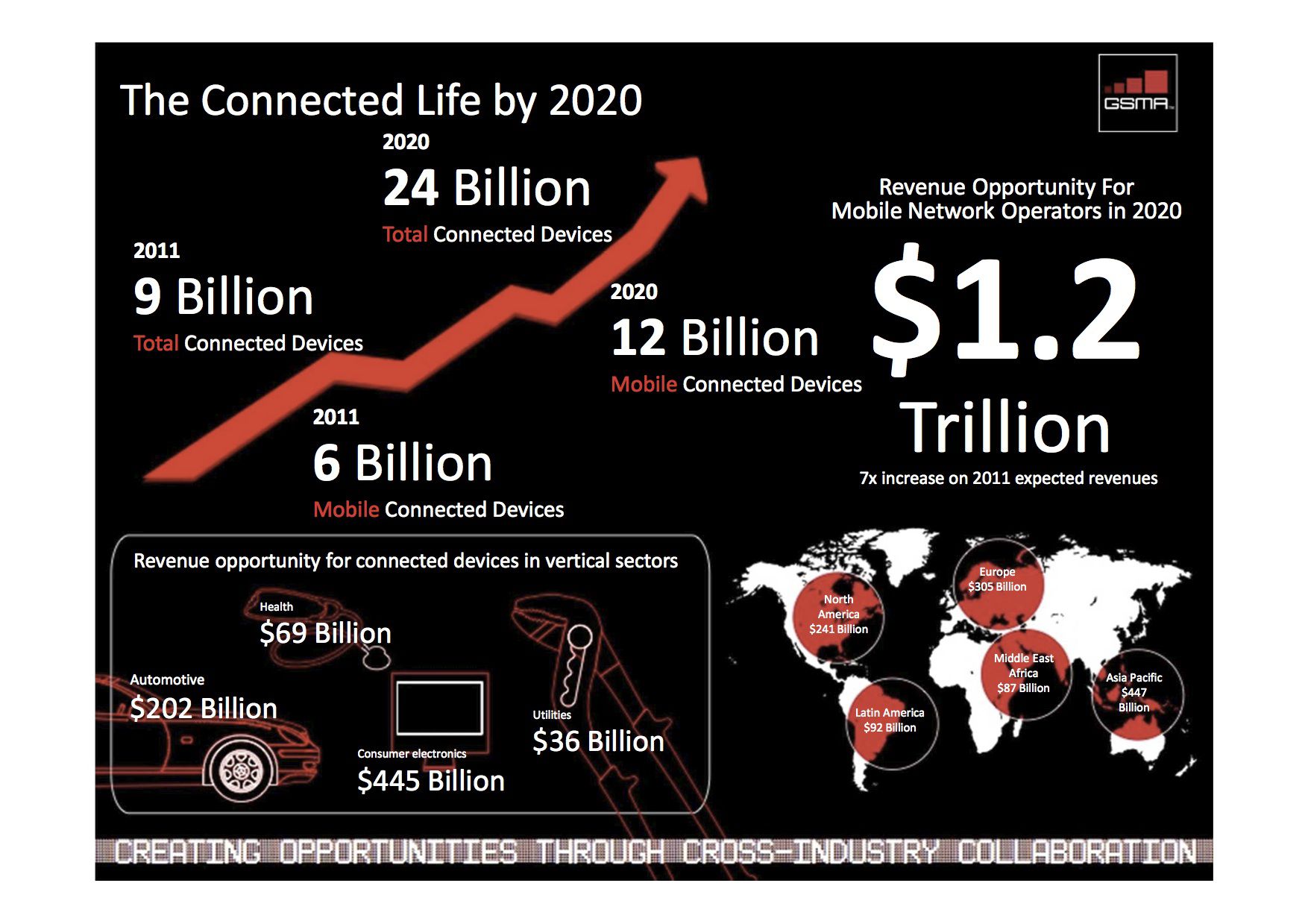


Figure ‎2.5. Mobile Connected Devices Study

## Analysis

### Hardware: Raspberry Pi

The Raspberry Pi is a low cost, credit card sized computer that plugs into any normal TV or monitor. It enables people to explore computing, whether they are old, middle aged or young. It can also be used to teach computer programming which uses special computer languages such as Scratch and Python. Raspberry Pi came from the idea that, without an affordable solution, children of the future would grow up without being computer literate in the new age of technology.

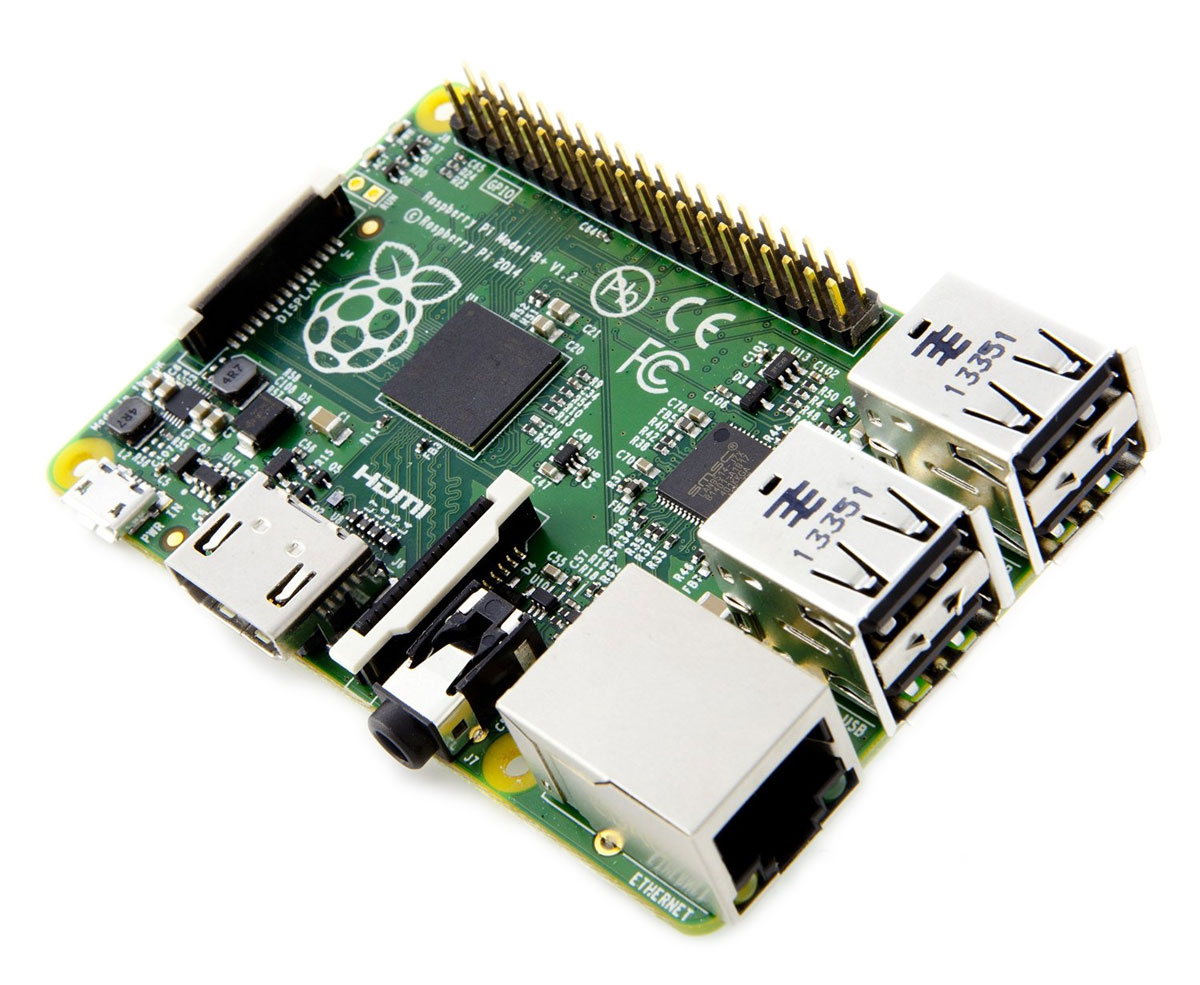


Figure ‎2.6. Raspberry Pi

What you can do with the Raspberry Pi vastly outweighs what you can’t do. The possibilities are practically endless. From something as simple as using it as normal computer with an office package and an internet browser to making your own retro games console. The GPIO (General-purpose input/output) terminals can be used in the same vein as a switch to turn objects on and off or even using it as a quick reaction game.

#### Raspberry Pi Specifications

|  |  |
| --- | --- |
| Specification |  |
| CPU | Broadcom BCM2835 SOC 700MHz |
| GPU | Broadcom VideoCore IV |
| Memory | 512 MB (shared with GPU) |
| USB | 2 x USB 2.0 |
| Video | Phono (RCA), HDMI (up to 1.4 compatible) and DSI socket |
| Audio | 3.5 mm jack |
| Storage | SD card slot |
| Network | 10/100 Mb/s |
| Power | Micro USB DC 5 V @ 800 mA for the Pi board |
| Dimensions | 85.6 mm x 53.98 mm |
| Operating Software | Supplied with Raspbian |
| I/O | 26 pin GPIO customisable pin array |
| Package Contents |  |
|  | Raspberry Pi Model B board |
|  | Wi-Fi Dongle |
|  | USB Keyboard |
|  | USB Mouse |
|  | 4 Port USB Hub |
|  | 2 Port USB PSU @ 2.1A |
|  | Micro USB Cable |
|  | SD Card with Raspbian OS pre-installed |

### Existing Products

#### Xively

Xively is a division of LogMeIn Inc., a global, public company that provides remote access and collaboration products including Rescue, Boldchat, join.me, and Cubby. Xively by LogMeIn offers an Internet of Things (IoT) platform as a service, business services, and partners that enable businesses to quickly connect products and operations to the Internet.

Figure ‎2.7. Xively Logo

#### https://tctechcrunch2011.files.wordpress.com/2015/05/particle.pngParticle

Particle (formerly known as Spark) offers a suite of hardware and software tools to help the user prototype, scale, and manage your Internet of Things products.

With Particle, the user's prototype is backed up by a real-time, secure and scalable cloud platform. The cloud platform will support your first product to your one millionth product, with our low-cost hardware modules providing a simple connection to the cloud at scale.

Figure ‎2.8. Particle Logo

# Requirements and Design

## Introduction

### Product Overview

Our product is an Internet of things development platform that ease the process of development application using the internet of things concept and abstracts a lot of work needed so that the developer would concentrate on the functionality of his app only.

## Specific Requirements

### External Interface Requirements

#### User Interfaces

1) A welcome screen with explanation of the project and some images to illustrate and links to other screens.

2) A login screen where the user is going to provide his user name and password to proceed to his account with a link to a forget password page.

3) Forget password page where he enters his mail and a send button

4) A sign up page where the user is going to provide his information to create a new account for him

5) After logging in the user is going to be shown his dashboard, a page where he can see how many device is connected last time each device sent anything and other details about his usage

6) When clicking on every device a page is shown for every device showing the data the device have sent statistics about the data the average and so in and some graphing about the data sent and links to the firmware code running on the device and the cloud code, when clicking on any of them it expands to show the code when the user press edit an instance of the web IDE is shown with code inside of it

7) The IDE page a text editor is shown in the left, a small window with folder view of files and folders below it a set of buttons which is {Run , Stop , Deploy } and then in the bottom center a window of output where the output of the code is viewed.

#### Hardware interfaces

##### Device side

Our target hardware device side is TI CC 3200 which is a microprocessor (ARM® Cortex®-M4 Core at 80 MHz) with a built in WiFi Receiver which is ideal for Internet of things applications.

##### Server side

Cloud code and data will run on server side on an average computer so it has no specific hardware interface and all hardware communications are managed by the underlying operating system on the web server.

#### Software Interfaces

The device code communicates with web server to send him data acquired and status and the server communicates with device code to send it commands or updated codes the server side code deals with REST requests to provide data and deals with DBMS to insert or update data in the database

#### Communication Protocols

The device is going to communicate with the web server using encrypted MQTT protocol on top of TCP/IP, the server is going to deal with client side using standard HTTP and deal with REST calls using HTTPs.

### Software product features

#### Functional Requirements

##### User login

The user should be able to login with his user name and password to the system

##### User sign up

The user should be able to create a new account while providing information like {name, Email, password}

##### User forget password

The user should enter his mail and the password be reset and sent to his mail.

##### Connect devices

The user should be able to connect to his devices and the system should provide him with a unique API key that the device can send to so the system would know to which user belong this device.

##### Accept data

The system should be able to accept data from the devices and store it for each user.

##### Showing Data

The system should provide the user with the data in tables.

##### Graphing data

The system should provide different graphing according to the data types

##### Device status

The system should provide the user with information about the status of the device including last time it sent , how many times it sent data , the time of every time data was sent , graph with these attributes

##### More than 1 device

The system should allow the users to have more than 1 connected device

##### Rest API

The system should have a rest API so that the user can hit the server and obtain his data

##### JS Library

A library would be made in Java script that handles REST and API calls that can be embedded in the developer code to ease the process of communicating with the System.

##### Online IDE

There should be an online IDE that the user can write different kinds of code in it.

##### Online IDE (2)

The IDE should have the basic features of a basic IDE like having a text editor, running code, stopping, deploying.

##### Server side code

The system should allow the user to write code that runs in the cloud based that can be triggered when his data arrives and the Rest API should handles how he gets his output the target language would be Java with a room for extension afterwards.

##### Flashing code remotely

The system should allow the user to write device code and flash it remotely to his device.

##### Device library

In order for the device to be able el be flashed remotely a library should be developed to allow this to happen.

##### Real time data

All the data sent must be handled and processed in real time as the type of applications running would be time critical.

##### Real time data (2)

In order for data to be real time the application protocol to be used is MQTT.

##### MQTT (Device)

To be able to provide MQTT a library should be developed on the device so that data can be sent using this protocol.

##### Device library (2)

In order for the server to deal with MQTT the server should have the right message broker to handle messages being sent.

### Software System Attributes

#### Reliability

The system should be reliable as in server must not crash and must be available all the time as developer applications rely on it , also data should have a backup in case any kind of damage happens

#### Security

The system of course must be secure as the data it contains is very critical and security is going to be achieved through:

1) Encrypting communication between the device and the server RSA and AES

2) Encrypting communication between server and client using SSL/TLS.

#### Maintainability

The System should be maintainable as in the future it can support languages other than java easily, also the library can have other targeted hardware so the cost of change should be minimal.

#### Performance

The library should be highly optimized and have the least footprint as possible so that there would be room for developers application specially with the limited resources of embedded devices so the library shouldn’t take more than 20% of processor time.

### Database Requirements

The database targeted would be MySQL but should have plans in the future for no SQL database like MongoDB so code dealing with database should be highly extendable.

## Architectural Design

### Introduction

Our product is an Internet of things development platform that ease the process of development application using the internet of things concept and abstracts a lot of work needed so that the developer would concentrate on the functionality of his app only.

### Architectural Representation

This document details the architecture using the views defined in the “4+1” model, but the views used to document the Internet of things platform application are:

##### Use Case View

Audience: All the stakeholders of the system, including the end-users.

##### Logical View

Audience: Designers.

##### Process View

Audience: Integrators.

##### Module Decomposition View

Audience: Programmers.

##### Data View

Audience: Data specialists, Database administrators

### Architectural Goals and Constraints

##### Server Side

IOTA will be hosted on one of Apache web servers running on a Java platform, and connecting to one of the MySQL Database servers. All communication with client has to comply with public HTTPS, MQTT , TCP/IP communication protocol standards.

##### Client Side

Users will be able to access IOTA only online. Clients/users are requiring using a modern web browser such as Mozilla or Chrome

##### Device Side

IOTA Libraries will be running on RTOS named “RTEMS” with portability options in any ARM Architecture board but target hardware which these libraries will be tested is “TI CC 3200”

#### Security

User’s devices can only be accessed , updated or removed by the users who created it , all communications with devices and server will be secured and cannot be sniffed or altered

#### Persistence

Data persistence will be addressed using a relational database.

#### Performance

The system should respond to the user in a real time way meaning that the level of server responsiveness that a user senses as sufficiently immediate or that enables the server to keep up with some external process such as change in sensor reading value

### Use-Case View

This is a list of use-cases that represent major functionality of the final system

* Sign up for the user
* Login for the user
* Create a new device
* Read data sent from device
* Write server code
* Deploy firmware code in device
* Deal with the system in “programmatic” way
* Send data from device to server
* Receive firmware updates from server to device

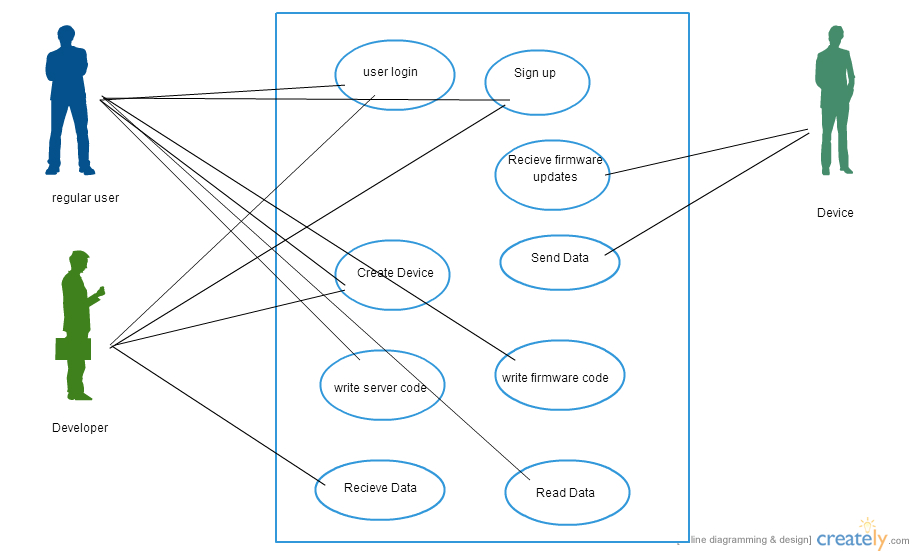


Figure ‎3.1. Use Case View

#### Actors

As described in the actors’ correspondence diagram below, web user could be one of three types.

##### Regular User

The user who can create a device, receives data from them, write server code or deploy firmware code.

##### Developer

A special type of user who can do all the user roles but in programmatic way so that he can embed it in his application

##### Device

The device that sends data to the server and receives firmware updates from server

### Logical View

IOTA is divided into layers based on the N-tier architecture

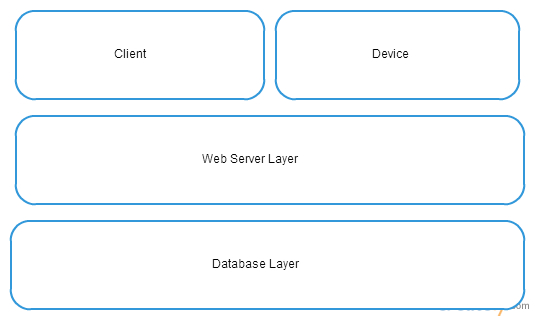


Figure ‎3.2. IOTA Logical View

### Process View

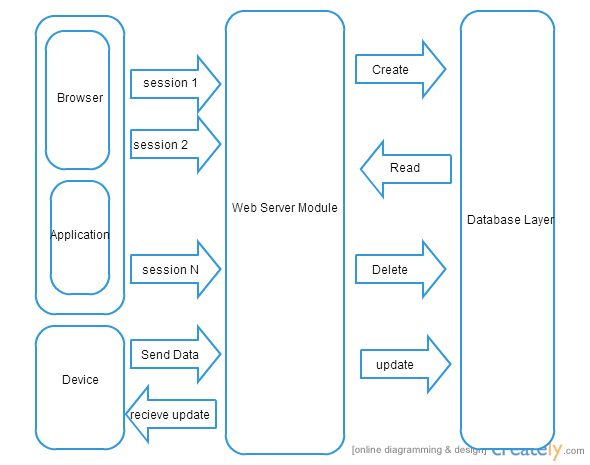


Figure ‎3.3. IOTA Process View

### Module Decomposition View

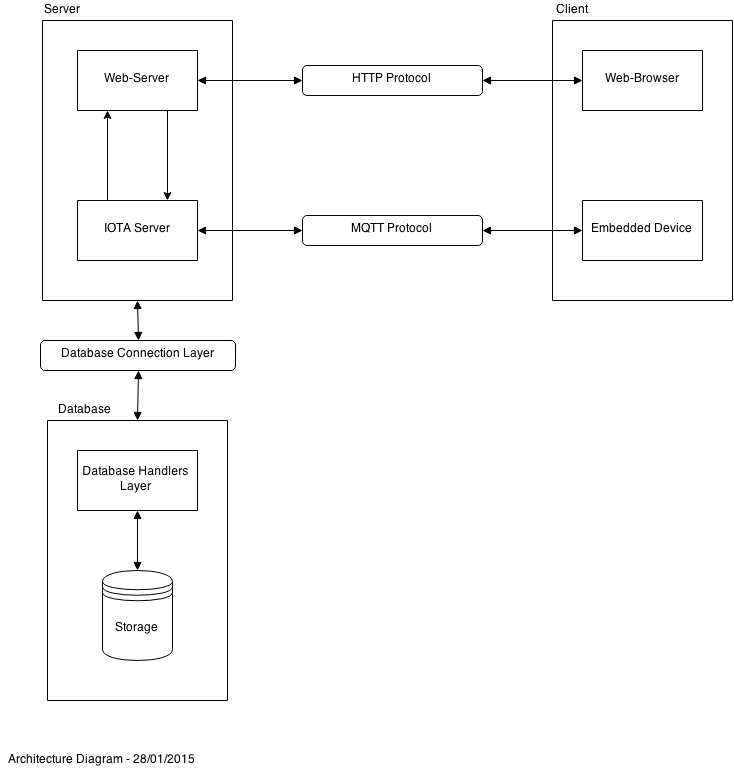


Figure ‎3.4. IOTA Module Decomposition View

### Data View

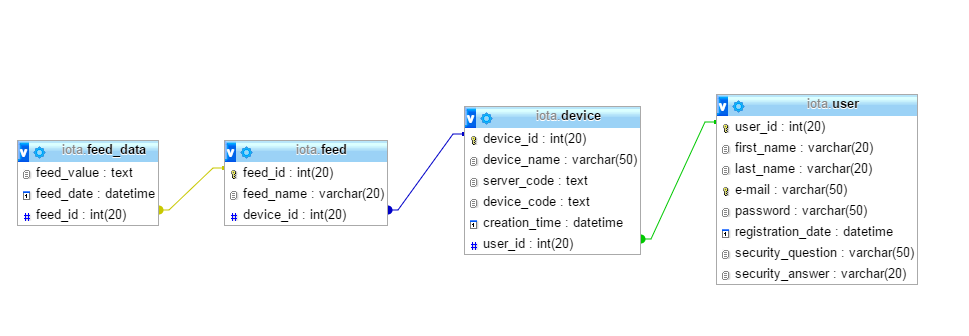


Figure ‎3.5. IOTA Data View

## Conceptual Design

### Experiential Flowchart

The Architectural Designs consists of three main parts

* The Server, consisting of
  + Web-Server
  + IOTA Server
* The Client, consisting of
  + Web-Browser
  + Embedded Device
* The Database, consisting of
  + Database Handlers Layer
  + Storage

The Web-Server and the Web-Browser are connected through the HTTP Protocol and the IOTA Server and Embedded Device are connected through the MQTT Protocol. While the Server and the Database are connected through a Database Connection Layer. See fig. hary.

### Interface Mockups

UI Screenshots

### Implementation Plan

##### Timeline

Figure ‎3.6. Implementation Plan Timeline

# Implementation

# Testing and Results

# Glossary

**AES** (Advanced Encryption Standard) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

**Client** is a piece of computer hardware or software that accesses a service made available by a server. The server is often (but not always) on another computer system, in which case the client accesses the service by way of a network. The term applies to programs or devices that are part of a client–server model.

**Cloud Computing** refers to the practice of transitioning computer services such as computation or data storage to multiple redundant offsite locations available on the Internet, which allows application software to be operated using internet-enabled devices.

**Dashboard** is an easy to read, often single page, real-time user interface, showing a graphical presentation of the current status (snapshot) and historical trends of an organization’s or computer appliances key performance indicators to enable instantaneous and informed decisions to be made at a glance.

**Data** is a set of values of qualitative or quantitative variables; restated, pieces of data are individual pieces of information. Data is measured, collected and reported, and analyzed, whereupon it can be visualized using graphs or images.

**Database** is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.

**DBMS** (Database Management System) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

**Embedded Device** is an object that contains a special-purpose computing system. The system, which is completely enclosed by the object, may or may not be able to connect to the Internet. Embedded systems have extensive applications in consumer, commercial, automotive, industrial and healthcare markets.

**Firmware** is a tangible electronic component with embedded software instructions, such as a BIOS. Typically they are used to tell an electronic device how to operate.

**Hardware** is the physical parts or components of a computer, such as the monitor, mouse, keyboard, computer data storage, hard disk drive (HDD), system unit (graphic cards, sound cards, memory, motherboard and chips), and so on, all of which are physical objects that can be touched (that is, they are tangible).

**HTTP** (Hypertext Transfer Protocol) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

**Hz** (Hertz) is the unit of frequency in the International System of Units and is defined as one cycle per second. It is named for Heinrich Rudolf Hertz, the first person to provide conclusive proof of the existence of electromagnetic waves.

**IDE** (Integrated Development Environment) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.

**IOT** (Internet of Things) is the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices.

**IP Address** (Internet Protocol Address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing.

**Java** is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

**Module** is any of a number of distinct but interrelated units from which a program may be built up or into which a complex activity may be analyzed.

**MQTT** (MQ Telemetry Transport) is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

**MySQL** is the world's second most widely used relational database management system (RDBMS) and most widely used open-source RDBMS.

**OTA** (Over-the-air programming) refers to various methods of distributing new software updates, configuration settings, and even updating encryption keys to devices where one central location can send an update to all the users, who are unable to refuse, defeat, or alter that update, and the update applies immediately to everyone on the channel.

**Protocol** (Communication Protocol) is a system of rules that allow two or more entities of a communication system to communicate between them to transmit information via any kind of variation of a physical quantity.

**RDBMS** (Relational Database Management System) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

**REST** (Representational State Transfer) is a software architecture style consisting of guidelines and best practices for creating scalable web services. It’s a coordinated set of constraints applied to the design of components in a distributed hypermedia system that can lead to a more performant and maintainable architecture.

**RSA** (Rivest-Shamir-Adleman Cryptosystem) is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret.

**Server** is both a running instance of some software capable of accepting requests from clients, and the computer such a server runs on. Servers operate within a client-server architecture where "servers" are computer programs running to serve the requests of other programs, the "clients".

**Software** is any set of machine-readable instructions that directs a computer's processor to perform specific operations.

**SQL** is a standard language for accessing databases. Our SQL tutorial will teach you how to use SQL to access and manipulate data in: MySQL, SQL Server, Access, Oracle, Sybase, DB2, and other database systems.

**SSL** (Secure Sockets Layer) provides a secure connection between internet browsers and websites, allowing you to transmit private data online. Sites secured with SSL display a padlock in the browsers URL and possibly a green address bar if secured by an EV Certificate.

**TCP** (Transmission Control Protocol) is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.

**TLS** (Transport Layer Security) is a cryptographic protocol designed to provide communications security over a computer network. It uses X.509 certificates and hence asymmetric cryptography to authenticate the counterparty with whom they are communicating, and to negotiate a symmetric session key.

**UI** (User Interface) is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision making process.

**URL** (Uniform Resource Locator) is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it.

**Web Browser** is a software application for retrieving, presenting and traversing information resources on the World Wide Web.