Software Requirements Specification

for

Morse-Pi

**Version 1.0 approved**

**Prepared by Chigozie Aham & Andrew McGuire**

**February 10, 2018**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Product Scope 1

1.4 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 Reading the Morse Signal 4

4.2 Transmitting the Message 4

4.3 View the Message on the App

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Introduction**

## Purpose

The purpose of this document is to give a detailed description of the requirements for the Morse-pi software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

## Intended Audience and Reading Suggestions

This document is intended for the developers and the any project leads involved in the development of the system. The SRS will contain the layout of how the entire system will work and its interdependencies. With this document, a developer should be able to recreate a similar system to that which will be created for the project.

## Product Scope

The goal of Morse-Pi v1.0 project allows for communication between two different parties using Morse code. Encoding of a message helps to ensure content protection, so only skilled users are able to do its translation. Messages could be transmitted directly between two raspberry Pis or over the cloud between one raspberry pi and then viewed on the Morse-pi android app. The user is able to view the original Morse code as well as its equivalent translation from the android app. A web application will also be used in order to modify the database, by being able to remove or edit messages that have been transmitted.

## References

http://www.galleryofthelouvre.com/morse-code-today.html

# Overall Description

## Product Perspective

This project is a self-contained system that allows users to learn and use Morse code to communicate over a network connection. The system requires a remote database that will allow users of the android app to view messages that they have sent via the raspberry pi.

## Product Functions

* Reads signal from Morse key
* Translate signal to text
* Creates sound for user feedback
* Transmits messages to Pis and cloud
* Android app listens to messages for learning experience

## User Classes and Characteristics

Pi users, will have the Morse key and will be using the Morse Pi, the Pi users will need to be able to send messages via Morse code and be able to hear what they are typing and make sure it is correct based on that as they do not see their own messages. This is the most important user experience as they will actually be learning how to Morse with a Morse key rather than with the app that lets then click dahs and dits to make their messages.

App users, will be using the app and some will be able to log in and see the messages transmitted from the Pis. This user will experience a simpler version of Morse as they won’t have to be skilled to create a Morse messages and then use the translator to get the text.

The Web users, will be like moderators they will see all the messages sent from the Pis in the database, and be able to edit and modify as needed.

## Operating Environment

The system will operate on three hardware platforms, two raspberry Pis 3.0, an Android phone that supports API 26 or newer, and a database which will be hosed on the cloud. The raspberry Pi will be operating in Python code and the Android App will be created with Java and xml in android studios. Both the Python code and the Java code will have connected to the database on the cloud, the Pi will send data to the database and to the second Pi and the Android App will fetch data from the database to display to the users, as well as use the database to insure login credentials, as not all users will be able to login and see the messages from the Pis.

## Design and Implementation Constraints

* User must have an internet connection for most of the functionality to work
* Low project budget so damaging part is NOT AN OPTION!
* Time constraint of 10 weeks to get system fully functional with all components working

## User Documentation

User manual, as well as build instructions will be provided (Link not yet available)

## Assumptions and Dependencies

API level 26 is assumed to be what the mobile phone will be running however the app does support a minimum of API level 23. The python compiler is also assumed to be 3.6.1.

# External Interface Requirements

## User Interfaces

The user interface that will need work/tweaking is the interface for the app. The app will display messages for the User in both Morse and in the text translation of the mores. The app will display the message one word at a time with the Morse above the text, this is done for readability so that the user can help to understand which bit of Morse represents what words.

## Hardware Interfaces

The hardware will connect to the Pi via its GPIO header which the Pi will be taking input from for the GPIO header.

## Software Interfaces

The Python’s code server/client code for the Pi will interface with a secondary python file that handles the translations from the dots/dahs to text. The App have classes dedicated for the Text to Morse translations, Morse to Text translations and a Morse to Sound.

## Communications Interfaces

The Pi and the app will connect to the database over the internet via a TCP connection in order to fetch and push data. The primary and secondary Pi will connect via TCP in order to transmit their messages directly. The Web interface will operate over http and allow the user in order to fetch and modify data within the database.

# System Features

The main functionality of the project is the translated text from the Morse that is read from the signal from the Morse key.

## Reading the Morse Signal

4.1.1 Description and Priority

This is the highest priority feature. This feature is the translation of the signal from the GPIO header to Morse, without this feature the project can’t be done.

4.1.2 Stimulus/Response Sequences

User begins to enters in a Morse letter on the first pi, user feedback should be the audio from the Pi letting the user know what they are inputting.

4.1.3 Functional Requirements

User keys in Morse letter on the contact key and hears instantaneous response from the Raspberry Pi. Any none Morse keys enters are ignored and after a brief moment a space in made of the second display and the primary Pi is able to start typing in the next Morse letter.

## Transmitting the Message

4.2.1 Description and Priority

This feature is the second highest priority feature, without this feature the system will not be able to use the database or connect to the secondary Pi. This feature allows the Pi to transmit the messages to the secondary Pi and the database.

4.2.2 Stimulus/Response Sequences

After The user enters a full Morse letter, the second Pi (to display the letter) and the Database should receive a message (a full word) after it has been entered,

4.2.3 Functional Requirements

The Pis must connect to each other and the database via TCP in order to transmit the messages.

**4.3 Viewing the Message on the App**

4.3.1 Stimulus/Response Sequence

The app will query the database for messages and refresh at an appropriate rate in order to display messages that have been sent by the Pi to the database. This is the lowest priority feature of the project but still one that is required for the end product.

4.3.2 Stimulus/Response Sequence

The user will navigate to the Pi monitor section of the app and be prompted to login. Once proper login credentials have been entered the user will be able to view all of the messages that the Pi’s have sent into the database and it will appear like a text conversation between both Pis. The user will be able to scroll through the messages to view messages that have been sent recently or messages that have been sent in the past.

4.3.3 Functional Requirements

The app must query the database and then either refresh manually or periodically in order to receive new messages

# Other Nonfunctional Requirements

## Performance Requirements

* Frequency of user entry is designed for learners, so the user would experience slower entry as compared to the general standard.
* The Morse control key may exhibit some debouncing issues and needs to be addressed via software or hardware.

## Security Requirements

Protection of the Morse codes sent over the network would use data encryption and decryption. This is implemented on the database level.

## Software Quality Attributes

Morse legends for letters and numbers are provided in the Android app to assist users in perfecting their knowledge of the Morse codes.

**Appendix A: Glossary**

API - Application Programming Interface

GPIO - General-purpose input/output

TCP - Transmission control protocol

**Appendix C: To Be Determined List**

TBD