MINS
Inc

W1 Project Proposal

Idir Zerrouk
Neethan Sriranganathan
Mohammad Issa
Solan Siva

Problem Statement:

Nearly 360,000 Canadians have identified as deaf and blind [1]. This toll has continued to increase and has been recorded at nearly 108 times that recorded in 2005 [2]. As the numbers increase the need for resources that help the deaf-blind in everyday tasks increases. Here at MINS Inc, we believe the next generation SMART Braille Box will provide every deaf-blind individual the means of understanding any typed text by converting each character of a word to braille and creating individual character pulses on the thumb of the reader.

The SMART Braille Box will take text as an input, convert the text to individual braille characters and provide the individual with character pulses on the thumb of the reader controlled by a set of 3 by 2 pins. No more feeling different and no more difficulties dividing non-disabled people from deaf-blind people. The box has a wide variety of uses that do not limit its function to receiving texts from any individual. A few of the box's functions are that it will provide blind-deaf individuals the ability to sit in on classrooms and understand all the notes, the ability to read any books and text content from any website, and with further development, the box will be able to receive texts from remote locations and have the ability to move forward and backward through the received text. Not only does the box provide the deaf-blind with an easy source of reading but also a cheaper option by removing the cost of an interpreter at nearly 26 dollars an hour [3]. With all these uses for the box and many more, the SMART Braille Box is the future and the now.

The SMART Braille Box will:

- Receive input text messages, notes, and any text through the android app
- The word will be split into individual letters
- The Letters will convert the text to braille and allow the deaf-blind individual to read notes, text messages, books and any kind of text with no worries
- The box will then use the converted character to control the individual pins. The set of 3 by 2 pins will pierce through the hole in the box where the individual will place their thumb and provide a pulse for every character in the received text

Design

System Architecture

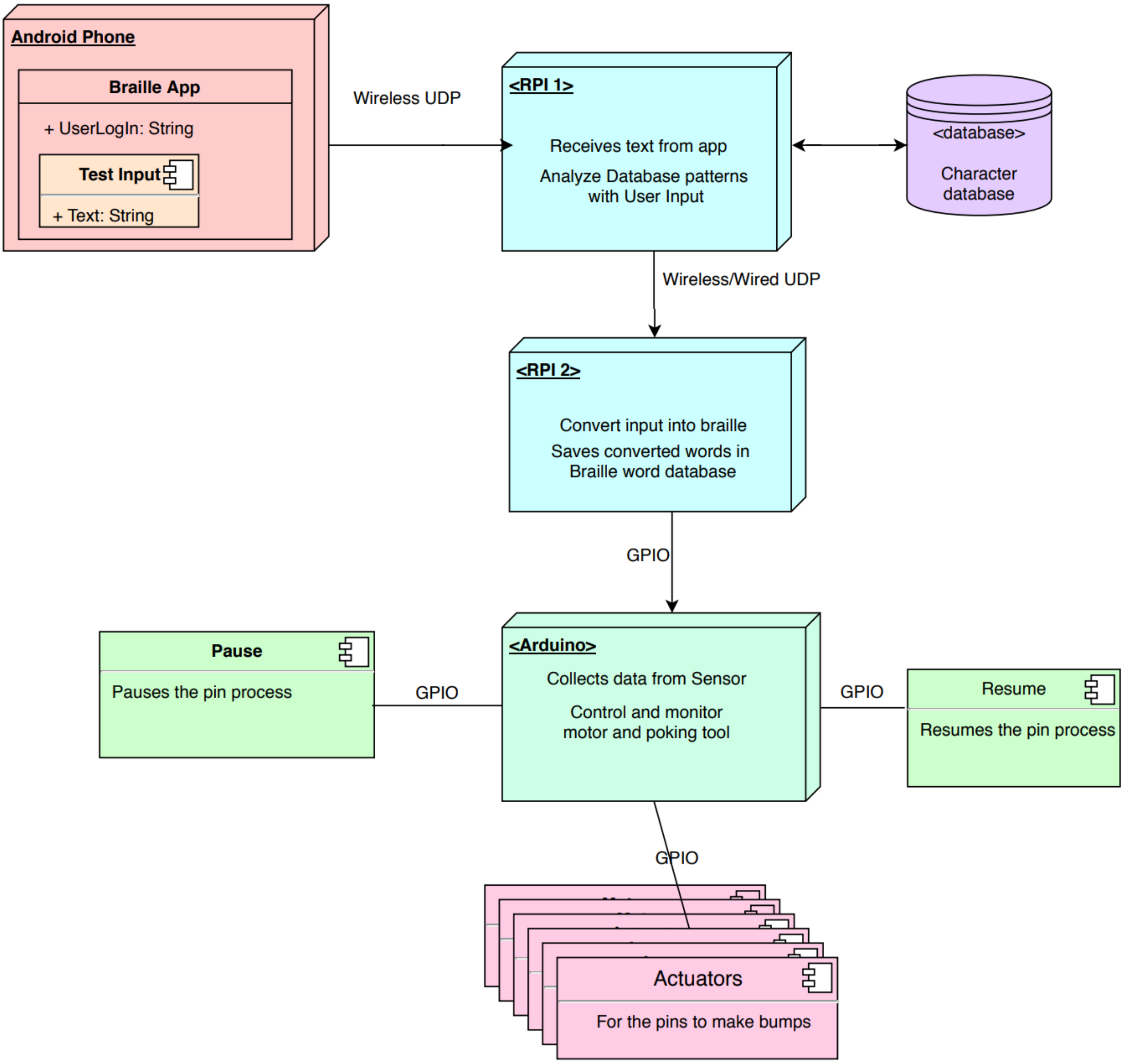


Figure1. System Architecture

Communication Protocols

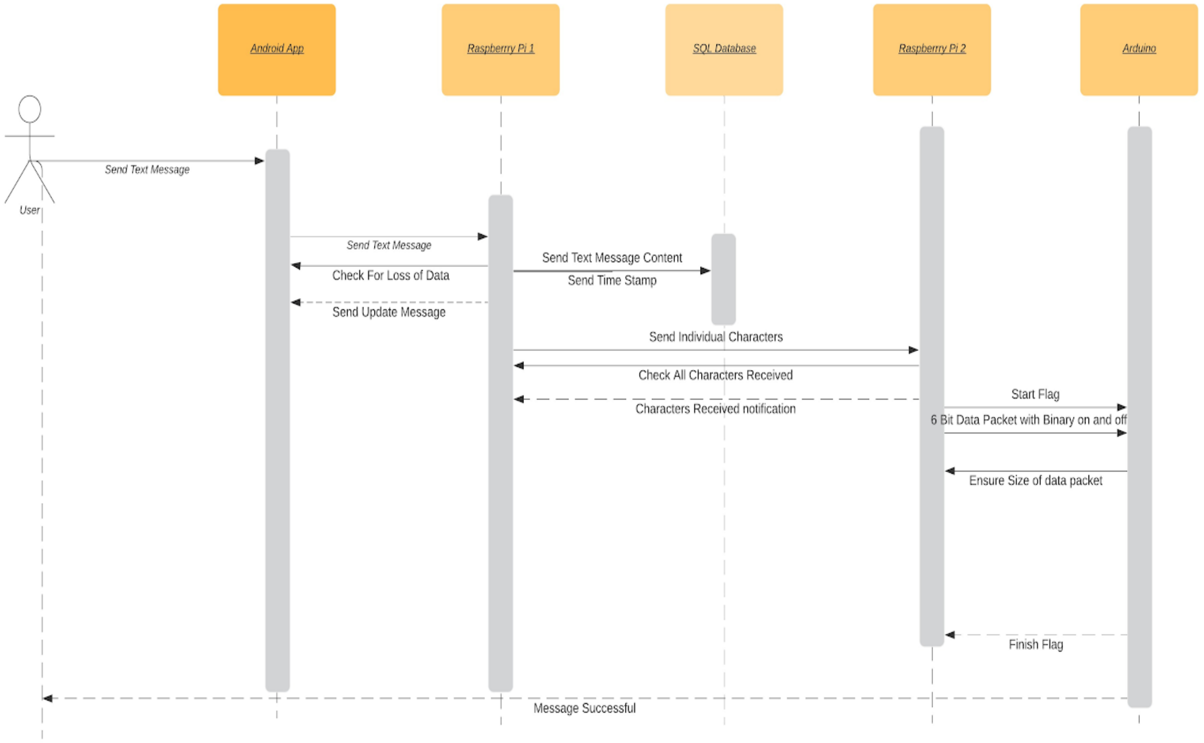


Figure2. Communication Protocols between App, Raspberry 1 & 2 and Arduino

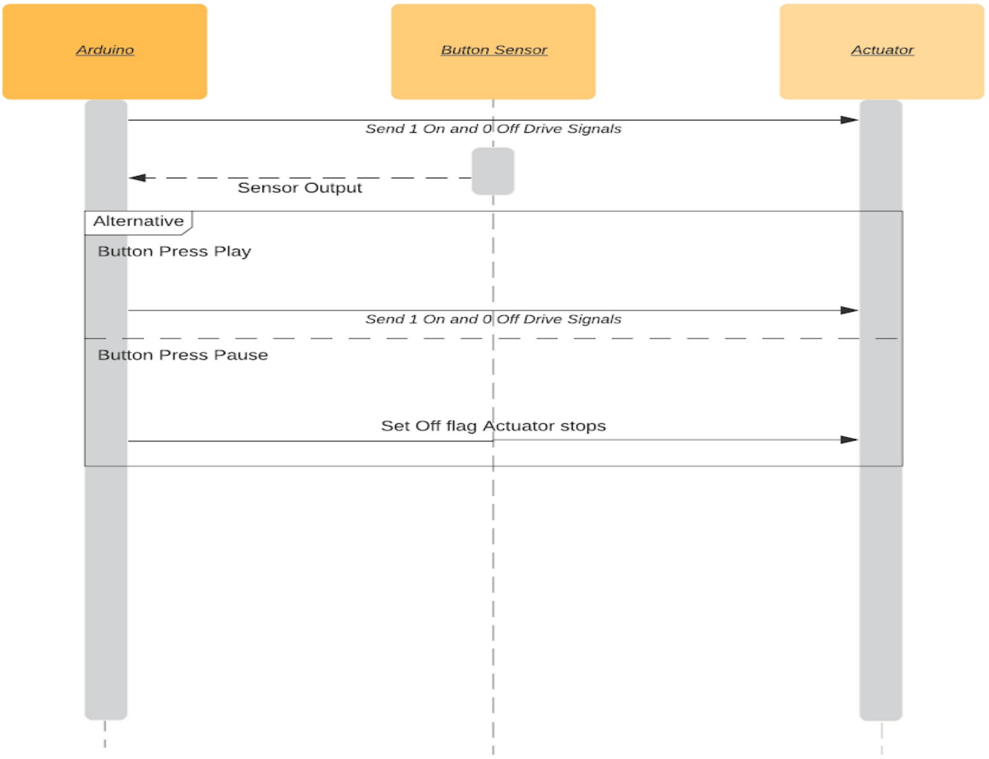

Arduino
Button Sensor
Actuator
Send 1 On and 0 Off Drive Signals
Sensor Output
Alternative
Button Press Play
Send 1 On and 0 Off Drive Signals
Button Press Pause
Set Off flag Actuator stops

Figure3. Communication protocol between arduino and the sensor

Sender	Receiver	Packet Format
User	Android App (Receiver)	[max one-sentence text format]
Android App (Sender)	Rpi 1	[max one-sentence text format]
Rpi 1	SQL Database	[text, text received timestamp (s)]
Rpi 1	Rpi 2	[characters of received text]
Rpi 2	Arduino	[0 and 1 bits to set on and off]
Arduino	Sensor	[0 and 1 bits to set on and off]
Sensor	Arduino	[0 and 1 bits to set on and off]
Arduino	6 Solenoids	[0 and 1 bits to set on and off]

UDP Protocol:
The communication protocol is used throughout the communication of the project. UDP protocol is similar to TCP protocol both protocols are built on top of the IP protocol. The difference is that over TCP, UDP a connectionless protocol provides a simpler and faster connection. UDP protocol does not check to see whether each packet has been received and does not error check, allowing the loss of certain information. The use of UDP is better for the Braille Box because it will provide a faster retrieval of data and to account for the loss of any data, the software design will account for any lost data within receiving code.

Communication Protocol Steps:

- User to Android App
    - Send Text Message to the individual, received through android app
- Android App to Raspberry Pi 1
    - Send Text Message to Raspberry Pi 1

    - Check to make sure no data loss
- Raspberry Pi 1 to Android App
    - Send update message of data received or data missing
- Raspberry Pi 1 to SQL Database
    - Sends the complete message that was inputted along with a proper timestamp
- Raspberry Pi 1 to Raspberry Pi 2
    - Packet contains a collection of characters of input text
    - Ensure collection received has the same amount of characters as the original text
- Raspberry Pi 2 to Raspberry Pi 1
    - Send notification that all characters were received or that characters are missing
- Raspberry Pi 2 to Arduino
    - Set Start Flag
    - Packets contain 6 bits of data **exactly** (Each bit is the on/off status of the 6 positions of the braille representation of a character)
- Arduino to Raspberry Pi 2
    - Arduino ensures the size of packet received is 6 bits
    - Send Finish Flag
- Ardiuno to Motors
    - Wired communication
    - A bit with 1 means motor is in “up” position, 0 means down
- Buttons to Arduino
    - Send 1 flag for On when play is pressed
    - Send 0 flag for Off when pause is pressed

Database

The database will have three three tables. First table will be the Message Table which will have Message of type string and a Message Id which is type int to give each message a way to identify them. The Message Id will be the primary key connecting to Message container where its foregin key will be the Message Id. The Message Container will have 5 components, the length of the message which will be type int, A timestamp of when the input is sent and a timestamp of when the message is done outputting both being type date, A user id which will be the primary key of this table and is type int, and the Message Id to connect the first two tables together. The user id was set as the primary key to be able to connect the user identifier to the user info table which contains the identifier number of type int, first name and last name of type string. The user id in the user info table is the foreign key allowing the connection to the message container table.

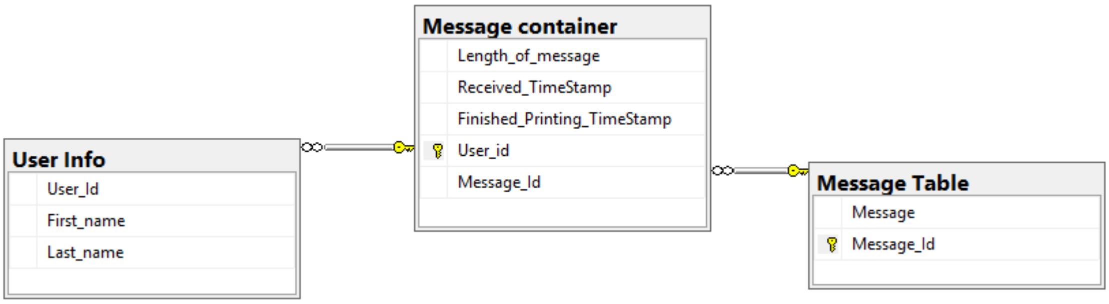


Figure4. The database diagram showing tables required for the Braille Box

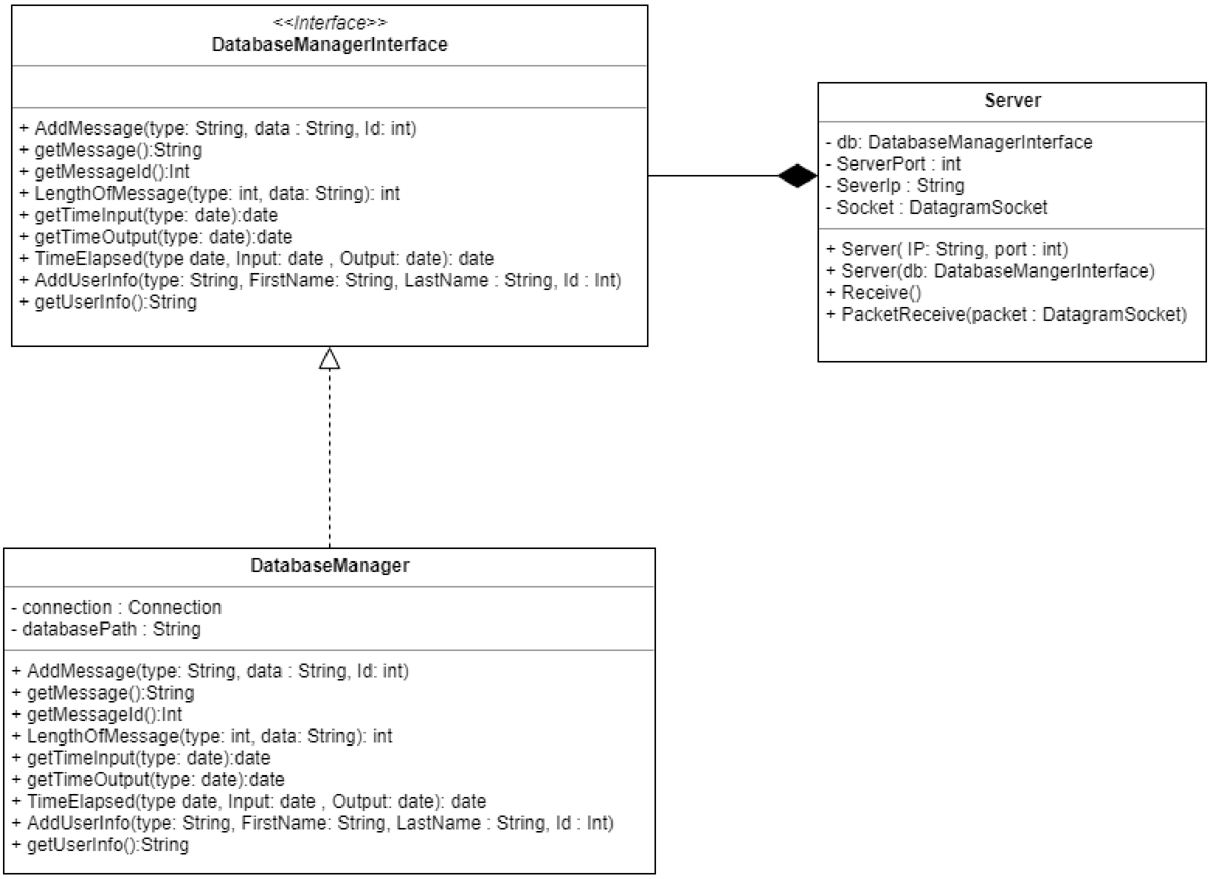

<<Interface>>
DatabaseManagerInterface
+ AddMessage(type: String, data : String, Id: int)
+ getMessage():String
+ getMessageId():Int
+ LengthOfMessage(type: int, data: String): int
+ getTimeInput(type: date):date
+ getTimeOutput(type: date):date
+ TimeElapsed(type date, Input: date , Output: date): date
+ AddUserInfo(type: String, FirstName: String, LastName : String, Id : Int)
+ getUserInfo():String
Server
- db: DatabaseManagerInterface
- ServerPort : int
- SeverIp : String
- Socket : DatagramSocket
+ Server(IP: String, port : int)
+ Server(db: DatabaseMangerInterface)
+ Receive()
+ PacketReceive(packet : DatagramSocket)
DatabaseManager
- connection : Connection
- databasePath : String
+ AddMessage(type: String, data : String, Id: int)
+ getMessage():String
+ getMessageId():Int
+ LengthOfMessage(type: int, data: String): int
+ getTimeInput(type: date):date
+ getTimeOutput(type: date):date
+ TimeElapsed(type date, Input: date , Output: date): date
+ AddUserInfo(type: String, FirstName: String, LastName : String, Id : Int)
+ getUserInfo():String

Figure5. Database Uml Model to show connection

Hardware Design

The hardware of the Braille Box consists of a small thumb-sized box where the person places their thumb on the box to feel the braille letters. The letters are displayed using small metal strips that are controlled by solenoids. The person would be able to feel the words inputted by moving through the letters using two buttons.The button on the left is used to move backwards and the right button moves forwards. The person can then move to the next letter if they got the first one, or move back to the previous letter if they weren't sure they got it.

The main centerpiece essentially looks like a one-sided die where instead of the pips (black dots) on the face of the die, there are holes where the pins move up and down denoting the braille letter. The centerpiece is connected to 6 other pieces that hold the solenoids.

The 6 extensions from the centerpiece each connect to the main mechanism required to move the pins. Each pin is placed under a hole in the centerpiece, which is connected to a seesaw bearing that is moved by the solenoid. When a pin is selected to move based on the braille letter, the solenoid for that specific pin is activated and pokes the seesaw downwards from one side which sends the other side which is connected to the pin upwards; displaying the braille letter.

The solenoids are connected to the power pin on the arduino that can be controlled give power; which will activate and deactivate the solenoids. The power will be distributed to all solenoids through a breadboard which will also include both buttons. The buttons will be connected to the second Raspberry Pi which is the Pi that uses the arduino. When the person feeling the braille letters would want to move forwards or backwards, they press the button and then the signal is sent to the second Raspberry pi; then the Pi sends the signal to move to the arduino.

Software Design

Figure6.Android App:

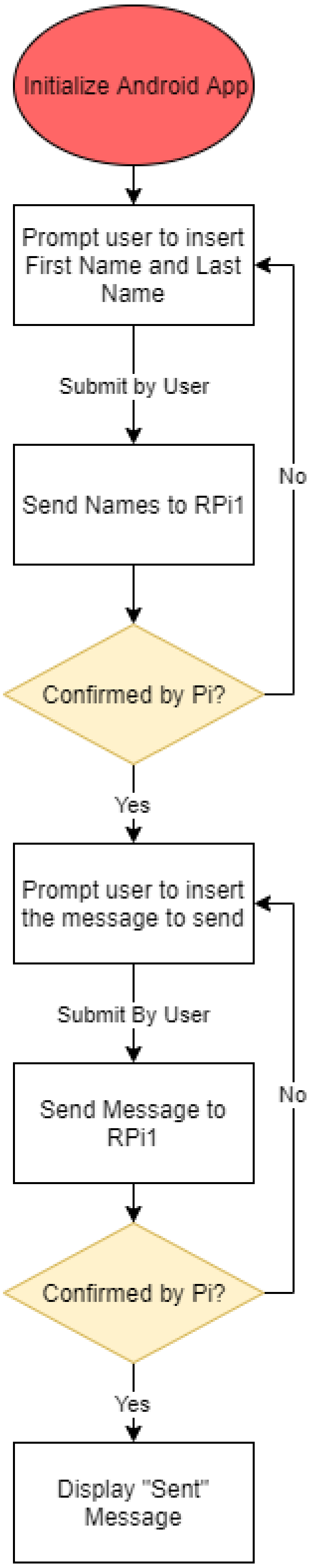


Figure7. Raspberry Pi 1:

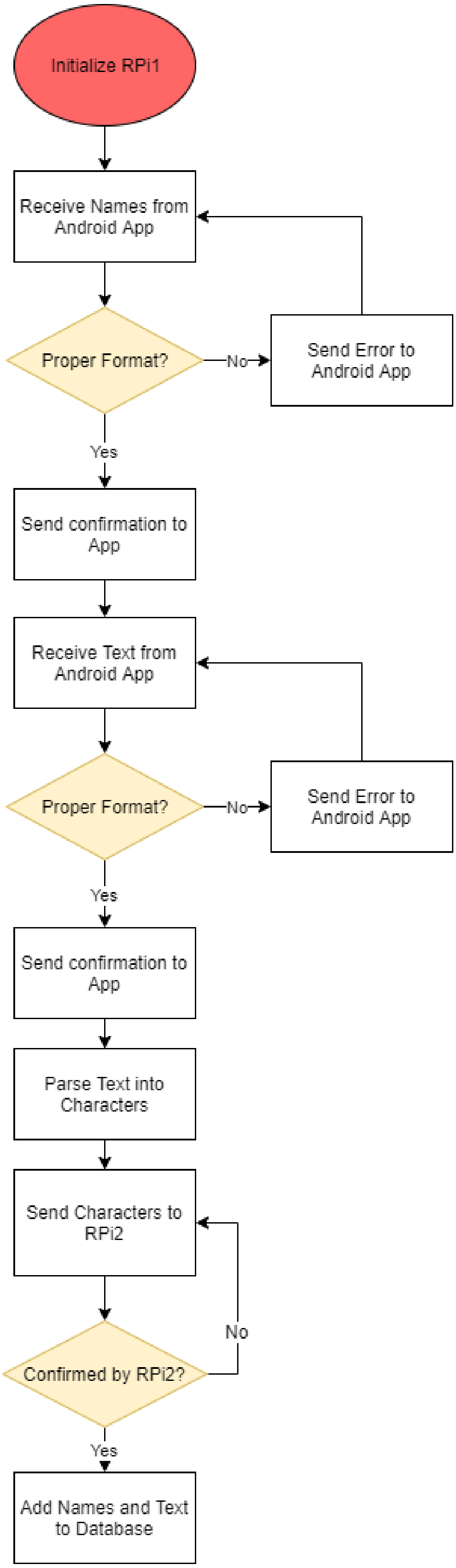

Initialize RPi1
Receive Names from Android App
Proper Format?
No
Send Error to Android App
Yes
Send confirmation to App
Receive Text from Android App
Proper Format?
No
Send Error to Android App
Yes
Send confirmation to App
Parse Text into Characters
Send Characters to RPi2
Confirmed by RPi2?
No
Yes
Add Names and Text to Database

Figure8. Raspberry Pi 2:

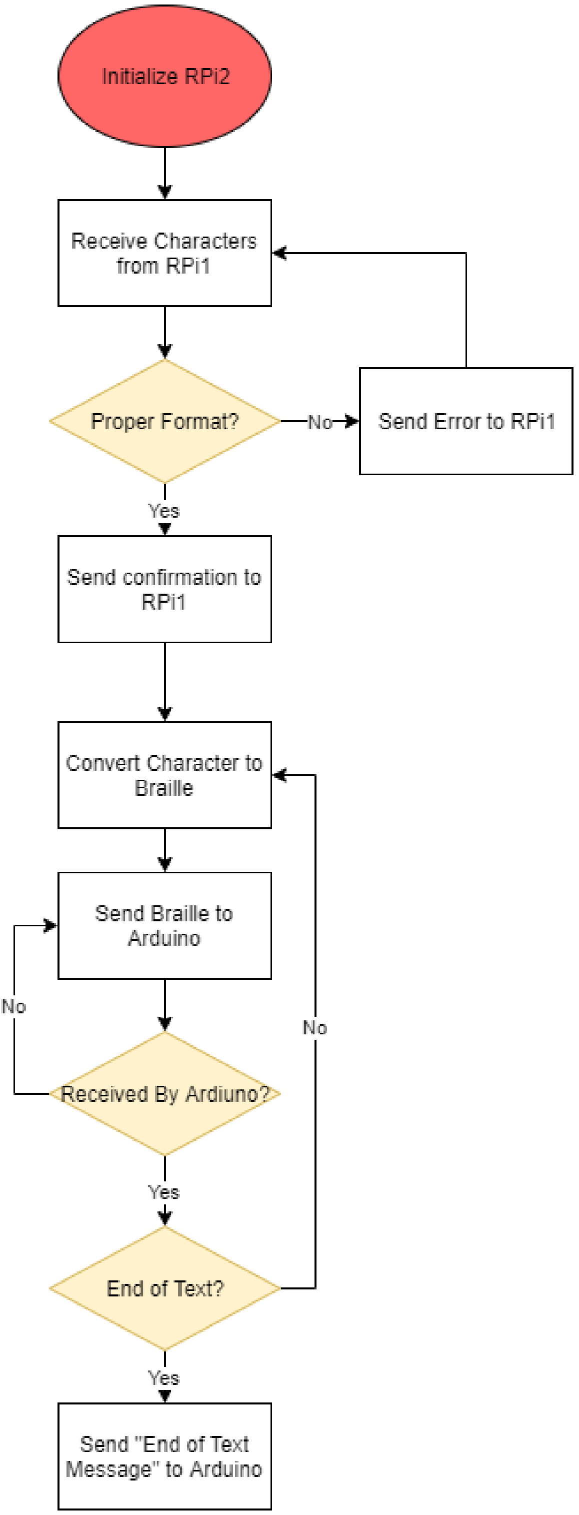
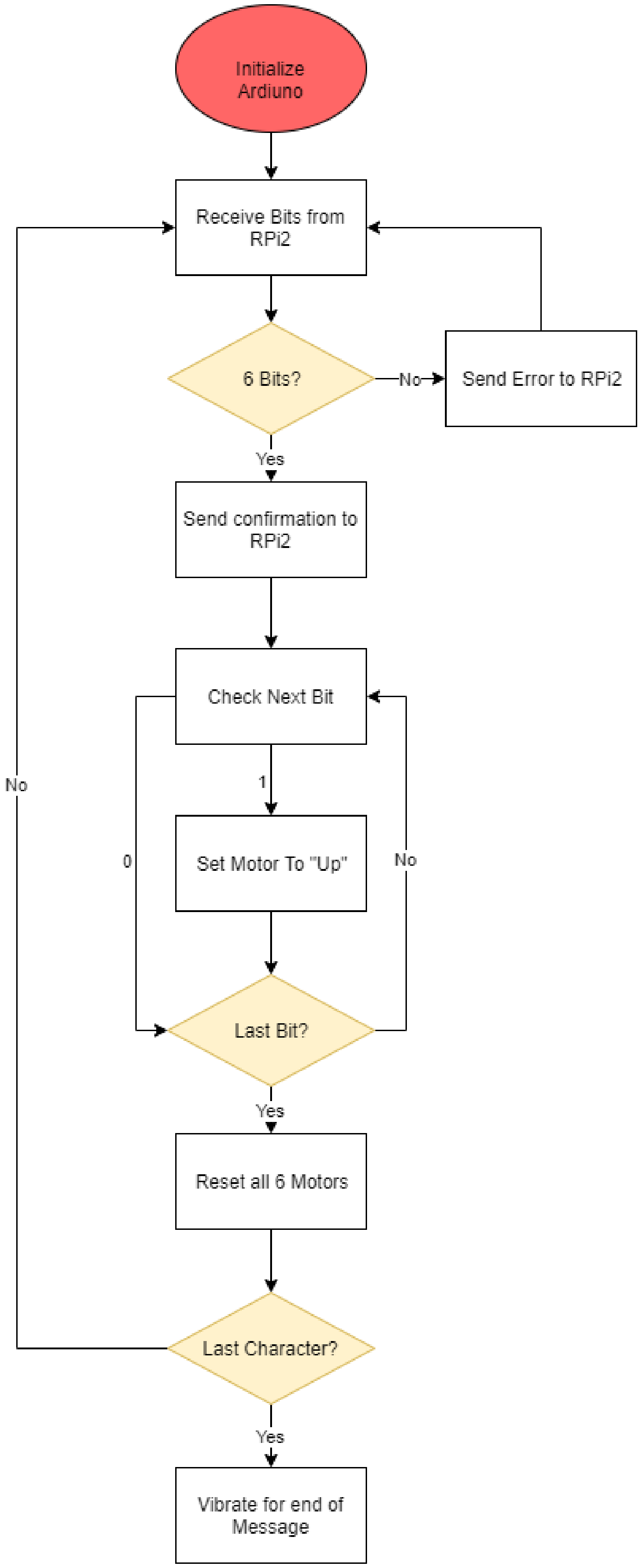

Initialize RPi2
Receive Characters from RPi1
Proper Format?
No
Send Error to RPi1
Yes
Send confirmation to RPi1
Convert Character to Braille
Send Braille to Arduino
No
Received By Ardiuno?
No
Yes
End of Text?
Yes
Send "End of Text Message" to Arduino

Figure9. Arduino:


Initialize Ardiuno
Receive Bits from RPi2
6 Bits?
No
Send Error to RPi2
Yes
Send confirmation to RPi2
Check Next Bit
1
Set Motor To "Up"
0
No
No
Last Bit?
Yes
Reset all 6 Motors
Last Character?
Yes
Vibrate for end of Message