50.001 - Introduction to Information Systems

POLARIS
ALWAYS WITH YOU

CI01 Group 4 - POLARIS Smart Lamp

Group Members:
Ainul Mardhiyyah
Bhavy Mital
Edric
Kwon Inhyuk
Wei Letong
Zhao Lutong

Table of Contents

Background & Problem Statement

Background

For many of us living or working far away from family, even trivial conversation may seem impersonal through a screen. As a result, we may feel obligated to compensate for the physical separation by packing our messages with information likes videos and emojis. More often than not, even when we are mentally exhausted and just want to mention something quickly, we end up “stuck” on long calls to family.

All in all, what does this all mean for our growing digital community. There are times when we just want to send a trivial and simple message, but the physical separation between us and the recipients makes it awkward to do so. How then can we send simpler messages while also capturing a good amount of personal value in them?

Problem statement

Our group envisions a way to send even the simplest of messages that is both meaningful and personal.

As our world inevitably becomes more and more digital, the personal value in our communications may appear to drop. When we are mentally exhausted from all our earlier focus on digital devices, we may unintentionally upset our loved ones when they are unable to communicate with us, or wear ourselves out even more trying to keep up with their need to catch up with our lives.

Our solution is a customisable Smart Lamp called **POLARIS**, accompanied by an app. Through the simplified in-app chat and the ability to remotely control the recipient’s lamp, we can convey key yet trivial information without uttering a word or being physically present.

System Design & Implementation

System Design

Our idea is to design a system where users can use an app to send short text messages and remotely change the colour of light and blinking pattern displayed on a physical lamp in real-time. Therefore, our solution would consist of the following:

● Smart Lamp (Raspberry Pi):	Displays different colours of light and different light blinking patterns. RGB codes of light colours and pattern types will be extracted from a cloud database
● Cloud database (Firebase)	Stores RGB codes and text messages sent by the app to lamp.
● Companion app	Displays an in-app messaging chat, and options to set light colours and blinking patterns to be shown on lamp

User Journey

- Personal: Users can access the **"Set My Lamp"** menu to set the light colour and blinking pattern to be displayed on **their own lamp**
- User-to-User: Users can also access the "See My Messages" menu to (1) send text messages to another user, (2) set another user's lamp to display preset colours according to certain emotions, (3) set another user's lamp to self-defined colours

Back-end

Users interact with the app to send messages, blinking patterns and colour codes to the lamp, and the recipient's app and lamp will be updated in real-time. This is done by connecting user selections within the app to a Firebase database, and also connecting the same database to the Raspberry Pi of the lamp. For our two-user implementation, users also share and access the same database.

When the user confirms their selection of colour and blinking pattern, this data is sent to the database where it will be constantly be checked by the lamp Raspberry Pi for new data. Once new data is received, the Raspberry Pi will process the data and display it almost instantly on the receiving lamp. A similar process occurs for sending of text messages between users, where the strings would also be sent to the Firebase and the in-app chat history will be updated almost immediately.

Discussion & Lessons learnt

- **.Amazon Web Services vs Firebase**
  One such setback is our attempt to use an Amazon Web Services Bucket as our database. Since it could only store file objects, using AWS bucket as our database would complicate the back-end of the app and lamp since integers and strings would have to be inserted into a text file, and access textfiles would have to be parsed through before the update to the recipient can occur.
  Firebase, on the other hand, could already take string and integer inputs, skipping the file processing step required in our previous AWS implementation. This means that it would be easier to use Firebase over AWS in our app from the beginning, and would have saved us more time that can be used to implement more complex features.

- **User Experience on Transitions between different Activities**
  After the first drafts of different activities were created, we realized that the transition between different activities should be arranged in a way so that it does not show the same activity repetitively. For instance, after the user set the username when first opening the app, the setting username activity should not be shown again unless the user wants to switch user. Thus, we saved the username into SharedPreference and check whether it is empty string before going into name setting activity. Also, after switching user, the app goes to username setting when the username saved in SharedPreference is empty so that there will not be additional need for manually switch between activities.

- **Java on Raspberry Pi**
  We tried to use Java on Raspberry Pi to run our lamp, but due to a lot limiting factors, we fall back to python. After doing some research we realized that java on RPi is still fairly new and there's not much tutorials and guides available online. In fact with java there are a lot of limitation on the GPIO pins which are not there in python. With java, there are only 4 GPIO pins that can be used for PWM but using python, that limitation does not exist. But since we only needed 3 GPIO pins for your lamp, this limitation didn't really stop us. But we faced another problem, which was connect to Firebase using java on RPi.

- **Java and Firebase**
  In android, it's an easy process to connect to firebase and on firebase's website we can find all the steps and tutorial in detail to do so. But when we are not using Android studio, it's a nightmare, especially for java. Although they have a third party library available to connect to firebase to Java, its not well documented because of which we had look for alternatives to use Java on RPi.

- **Other OS on Raspberry Pi**
  Since we were not able to use java on the Raspbian OS installed on our RPi, we thought maybe we could using some other OS which may resemble our PC's OS or which will allow us to use Android Studio. We start looking in Windows IoT core and Android Things. Both of them are available for RPi but in the end, they had their own set of limitations. Windows IoT core does not support Java (at least not when we were working on our prototype) and Android Things only support one GPIO pin to be used for PWM. So in the end we fall back the original setup which was Python > Raspbian OS > RPi.

On a whole, our group could have done more research earlier about the different features we would like to implement and the technologies we were using, as it is one of the main reasons why we have quite a number of setbacks

Future Works & Conclusion

Future Works

Despite our combined efforts, our group was unable to implement all that we envisioned for POLARIS. Given more time and budget, we could have explored:

1. Sending and playback of short audio messages, which we feel has even more personal value than just text messaging and real-time colour display changes.
2. Integrating a touch screen on our Lamp which would allow user to interact in many more ways.
3. Implement a text-to-speech engine in app to convert typed messages to audio to be played on the recipient lamp.
4. Designing the lamp in a way so that it can also be used in communication with children with special needs.

Conclusion

Our project idea has all begun from the problems just around us and problem might have been neglected and ignored by most of us for some reasons. Although our prototype still needs further development and improvement before it gets introduced to the market, we believe that we have definitely made progress as to how we could address the real world problem. We also learned that JAVA, the object-oriented programming language, is so much of a great potential and help to structure and put up a sophisticated design of our system. By extensively making use of Raspberry Pi and Firebase, we could bring in synergy with our JAVA-based app design. At the end of the day, we hope that POLARIS would be complete enough to not just be ready for show but also to connect all of us to the people we care the most.