FIT3140: Advance Programming

Spiking and Analysis of Alternatives

Morse Code Decoder: The server reads form the motion sensor and sends the data to clients that should interpret the received data into alphabets

Team Member 1: Melissa Anne Pereira

Team Member 2: Jacky Lai Choon Huey

Contents

[Background Information 3](#_Toc480736224)

[Spikes Analysed 3](#_Toc480736225)

[Background Information: 3](#_Toc480736226)

[Analysis Criteria: 3](#_Toc480736227)

[Spike Plan 1 4](#_Toc480736228)

[Spike Outcome 1 5](#_Toc480736229)

[Spike Plan 2 6](#_Toc480736230)

[Spike Outcome 2 7](#_Toc480736231)

[Analysis of Alternatives Used 8](#_Toc480736232)

[Recommendation 9](#_Toc480736233)

[Bibliography 9](#_Toc480736234)

# Background Information

## Spikes Analysed

1. Johnny-Five and Socket.io using Node.js
2. BoneScript and Google Firebase using Node.js

## Background Information:

Bonescript: A library of Node.js specifically for BeagleBone. It provides functions for hardware interaction. (beaglebone.org, 2016)

Johnny-Five: A JavaScript library for the Robotics and IoT platform. (Bailey, 2013) It is mainly developed for Arduino but has io packages that make it compatible with other microcontrollers.

Socket.io: A JavaScript library that is an event based bi directional communication layer for web applications. (Kelleher, 2014)

Google Firebase: Firebase is a real time, cloud hosted database. The data is stored as a JSON and is synchronized to every connected client. (Google Firebase, 2017)

## Analysis Criteria:

1. Latency of connection between client and serve
2. Learning curve for the toolchain
3. Support and documentation
4. Interoperability between toolchains

# Spike Plan 1

**Name:** Johnny-Five and Socket.io for real-time bi-directional client server communication using Node.js

**Context:**

This spike is conducted to determine if it is possible for our team to implement real-time bi-directional client server communication using Johnny-Five and Socket.io using Node.js

**Gap:**

This spike will be conducted to mitigate the risk of not being able to develop the project due to a steep learning curve and integration of technologies. Our team has had no experience with Johnny-Five therefore we will conduct the spike to determine whether it would work seamlessly with the BeagleBone black so that we would mitigate the risk of encountering compatibility issues later in the project. We would also investigate the usage of socket.io for client-server communication to understand how it would interact with other packages required for connection to reduce the risk of our team not being able to continue with the toolchain due to a lack of understanding.

**Goals/Deliverables:**

* To determine whether Johnny-Five can be used with BeagleBone Black
* To determine whether there is sufficient support for Johnny-Five and Socket.io
* To determine the latency between the client-server connection
* To determine whether our team can learn the toolchain

**Planned start date**: 14/April/2017

**Deadline:** 17/April/2017

**Planning notes:**

First, the plan would be to view sample implementation codes of socket.io over the internet and material prepared for us on Moodle. Next, we would write simple code to understand basic concepts and try to use socket.io with sample data. After that, we will explore the option that Johnny-Five would offer us and implement it in our spike. Throughout all this, we will use the documentation provided by both Socket.io and Johnny-Five to determine their helpfulness. We will divide our responsibility by having one of us find a good reference and sample code as guidance and the other person will write the code and test whether it works or not.

# Spike Outcome 1

**Name:** Johnny-Five and Socket.io for real-time bi-directional client server communication using Node.js

**Goals:** Real-time bi-directional client server communication and to see if there are any constraints in the process

**Personnel:** Jacky Lai Choon Huey, Melissa Anne Pereira

**Technologies, Tools, and Resources used:**

Website such as stackoverflow.com, GitHub and sample code from Moodle/resource kit from Monash.

**Tasks undertaken:**

First, we view sample implementation codes of socket.io over the internet. Then, we wrote a simple code to understand its basic concepts and have a simple Hello World communication between client and server. After that, we use more advance technique and packages like Express to test out the functionality of these packages. Next, we try to implement the said functionality using Johnny-Five’s IoT platform.

**What we found out:**

As expected, socket.io was a steep learning curve because we did not understand their purpose enough but we manage to learn it core functionality during this spike. Besides that, what we found out for Johnny-Five was that the sources code and documentation we can find is mostly Arduino based platform so this prove to be tough when we are developing for BeagleBone platform.

**Open issues/risks:**

We discovered it was not easy at the beginning because we lack the knowledge in using socket.io and the fact that Johnny-Five was mainly documented in Arduino platform cause some confusion in our team. Nevertheless, with some guidance from the lecture and online research we manage to do this task.

**Conclusion:**

We feel that socket.io and Johnny-Five is not too complicated and the code we wrote could be used in our future projects. Besides that, we feel that Johnny-Five could provide the scalability the project is looking for. We don’t feel more spikes are needed for this functionality. Our team prefers this over the second spike.

# Spike Plan 2

**Name:** BoneScript (Node.js library) and Google Firebase for real-time bi-directional client server and database communication.

**Context:**

This spike is conducted to determine if it is possible for our team to implement real-time bi-directional client server communication using BoneScript as the programming language and Google Firebase as the cloud database.

**Gap:**

This spike will be conducted to mitigate the risk of not being able to develop the project due to the toolchains requiring tedious work to implement, integration of the toolchains and a steep learning curve. Our team has had previous experience with Google Firebase and BoneScript but we are trying to ensure that we will not face too many problems before we develop the larger project namely the Morse Code Decoder. Therefore, we would conduct the spike to determine whether Google Firebase would work seamlessly with the BeagleBone Black so that we would mitigate the risk of encountering compatibility issues later in the project. We would also investigate the usage of Google Firebase for client-server communication as well as how the database should be structured as Google Firebase does not follow the conventional database structure. This will greatly reduce the risk of our team not being able to continue with the toolchain due to a lack of understanding.

**Goals/Deliverables:**

* To determine whether BoneScript is sufficient to develop the project Morse Code Decoder using BeagleBone Black.
* To determine whether there is sufficient support for BoneScript and Google Firebase
* To determine the latency between the client-server connection
* To determine whether our team can learn the toolchain using these two technologies in particular

**Planned start date**: 18/April/2017

**Deadline:** 22/April/2017

**Planning notes:**

First, the plan would be to review the BoneScript implementation in our previous work as well as the samples found online. Next, we would write simple code to understand basic concepts and try to use BoneScript for simple tasks. After that, we will make use of Google Firebase as the database and communication platform. Throughout this, we will explore the option that Google Firebase would offer us and implement it in our spike. Besides that, we will use the documentation provided by both Google Firebase and BoneScript to determine their helpfulness. We will divide our responsibility by having one of us find a good reference and sample code as guidance and the other person will write the code and test whether it works or not.

# Spike Outcome 2

**Name:** BoneScript (Node.js library) and Google Firebase for real-time bi-directional client server and database communication.

**Goals:** Using Google Firebase as the database and communication platform implemented using BoneScript a Node.js library created for BeagleBone IoT.

**Personnel:** Jacky Lai Choon Huey, Melissa Anne Pereira

**Technologies, Tools, and Resources used:**

Website such as stackoverflow.com, GitHub and sample code from Moodle/resource kit from Monash.

**Tasks undertaken:**

First, we view sample implementation codes of BoneScript from our previous classes and over the internet. Then, we wrote a simple code to understand its basic concepts and have a simple Hello World communication between client and server and try to store certain data in every communication. After that, we use more advance technique and try to make use of the function Google Firebase had to offer. All of this are to be implement using BoneScript.

**What we found out:**

Google Firebase was not a hard platform to implement because of the huge amount of documentation and support from the community but there are certain disadvantages when developed using BoneScript. For example, errors are not shown properly to allow smooth debugging process. Besides that, what we found out for BoneScript was that the library is not high level enough to be used as a future proof platform but it was easy because it was developed for the hardware we are specifically using.

**Open issues/risks:**

We discovered it was not easy at the beginning because we lack the knowledge in using both technologies together and being newly introduce into these technologies cause some confusion in our team. Nevertheless, with some guidance from the lecture and online research we manage to do this task.

**Conclusion:**

We feel that Google Firebase and Bonescript is not too complicated and the code we wrote could be used in our future projects but we feel that we should replace Bonescript with Johnny-Five because the functionality that BoneScript provide is simply too basic for future implementation. Finally, we don’t feel more spikes are needed for this functionality.

# Analysis of Alternatives Used

Bonescript:

* The learning curve for this toolchain was relatively low as it uses simple concepts to deal with the hardware. The functions are also named in such a way that they are intuitive and can be easily understood, for example, digitalWrite()
* The documentation for BoneScript is very complete and has many examples which makes it easy to use.
* BoneScript being built specifically for BeagleBone, has no issues with the BeagleBone. It also has no known compatibility issues with Firebase or Socket.io.

Johnny-Five:

* The learning curve for this toolchain was relatively steep as it required the understanding of event based programming but seeing as our team has this understanding, it was not too difficult.
* The documentation for Johnny-Five and the examples given were mostly written for Arduino making it slightly more tedious as one must translate that for the BeagleBone. Certain functions were also deprecated but still appear in the examples which could cause confusion.
* Using the beaglebone.io made Johnny-Five compatible with the BeagleBone and there are no known compatibility issues with Socket.io or Firebase.

Socket.io:

* The latency using Socket.io was lesser compared to when using Firebase. Although it was a negligible amount.
* The learning curve for this toolchain was steep as our team did not understand the purpose of socket.io. However, after understanding its purpose, implementation was relatively easy.
* Support for socket.io is average as the documentation is complete but there are few examples that can be used as reference.
* There are no known compatibility issues with Johnny-Five or BoneScript.

Google Firebase:

* The latency using Firebase was slightly more compared to socket.io presumably because the data had to be updated to the database and only then could the client read. However, the difference is of a negligible amount.
* The learning curve for this toolchain was not too steep as the functions were very intuitively named. The structure for the database however is a JSON structure which would require some research for inexperienced users.
* The documentation for Firebase is the best out of all the toolchains explored as not only do they offer sample code and documentation but they also offer short videos explaining the samples of code.
* There are no known compatibility issues with Johnny-Five and BoneScript.

# Recommendation

Our team would recommend using a combination of Johnny-Five for hardware communication as the high-level nature of Johnny-Five would make it more usable for larger projects such as the Morse code decoder. Since our team found it relatively simple to translate the examples for Arduino to BeagleBone, we found that the current support is sufficient. As there are no known compatibility issues, it would work well with either Firebase or Socket.io.

We selected Google Firebase for our client-server communication because should the project require scalability, Firebase would be able to provide those services without the developers having to program everything from scratch. The documentation for Firebase being so complete, would be useful when developing the project as it would aid our team in learning new functions to implement our project.

Node.js is the preferred language as both our team members have worked with JavaScript in our past few projects, this makes the language familiar and still recent in our minds which would further reduce the learning curve when developing the next project.

# Bibliography

Bailey, D. (2013, July 16). *JavaScript Powered Arduino with Johnny-Five*. Retrieved from Safari: https://www.safaribooksonline.com/blog/2013/07/16/javascript-powered-arduino-with-johnny-five/

beaglebone.org. (2016, February 10). *Bonescript*. Retrieved from beaglebone.org: http://beagleboard.org/Support/BoneScript

Google Firebase. (2017, April 13). *Firebase Realtime Database*. Retrieved from Firebase: https://firebase.google.com/docs/database/

Kelleher, F. (2014, August 11). *Understanding Socket.IO*. Retrieved from NodeSource: https://nodesource.com/blog/understanding-socketio/