**< WEEKLY REPORT FOR WEEK 2 >**

Name: Oh SuJin

Project: Multimodal Sensor Interfacing, Acquisition and Visualization

Duration: 26/5/2014 – 30/5/2014 (Week 2)

**I) Project Work Summary**

**Finished:**

* Studied different hardware from Kinect to Leap Motion and SDKs provided. Build simple programs to test various samples provided in SDKs.
* Familiarized with Eclipse environment.
* Studied various Kinect library and tools to come up with framework suitable for Kinect-based rehab application.
* Review on past IO reports of Ericko; read through his findings about AcceleGlove and its limitations.
* Explored the possibilities of using MS Kinect-OpenNI bridge.

**Ongoing:**

* Data acquisition from leap motion; what types of data are available in what languages and which one to choose for iFarm.
* How to store such data, in what format
* Study available finger/hand exercise and try to record using existing leap motion data recording application to determine whether it's possible to detect and recognize those exercises.
* Communication between Leap motion and Arduino: WebSocket vs Serial communication

**II) Tasks Assigned**

* Study MS Kinect-OpenNI bridge and explore its usefulness and applications in the scope of our project.
* Study Ericko’s report especially on node.js and AcceleGlove.
* Research on node.js / find out how to send data from Leap motion to Arduino
* Study WebSocket communication for Leap Motion.
* Data acquisition from leap motion; what types of data are available in what languages and which one is most appropriate for our project.
* Become more familiar with new languages and IDEs required: Javascript, C#, Visual Studio, etc.

**III) Detailed Activities / Accomplishments**

**DAY 1:**

* Learn AcceleGlove SDK Version 1.0.1 which provides the necessary tool to interface with the AcceleGlove. The AcceleGlove SDK provides two Java API:
* Raw DataStream API that accesses the X, Y, Z axis values
* Gestures API that uses the raw data stream to create and access gestures.

Similar to Leap Motion, AcceleGlove SDK comes with a predefined library of functions for Java environment. I can capture

* Understand how to make Leap Motion and Arduino talk using Bluetooth. In order to do so,
* Bluetooth serial port must be identified
* LeapJava.jar (and also the DLLs) and RXTXcomm.jar must be added to the Eclipse project

References:

[Two-way communcation with the serial port](https://web.archive.org/web/20120820200551/http:/rxtx.qbang.org/wiki/index.php/Two_way_communcation_with_the_serial_port)

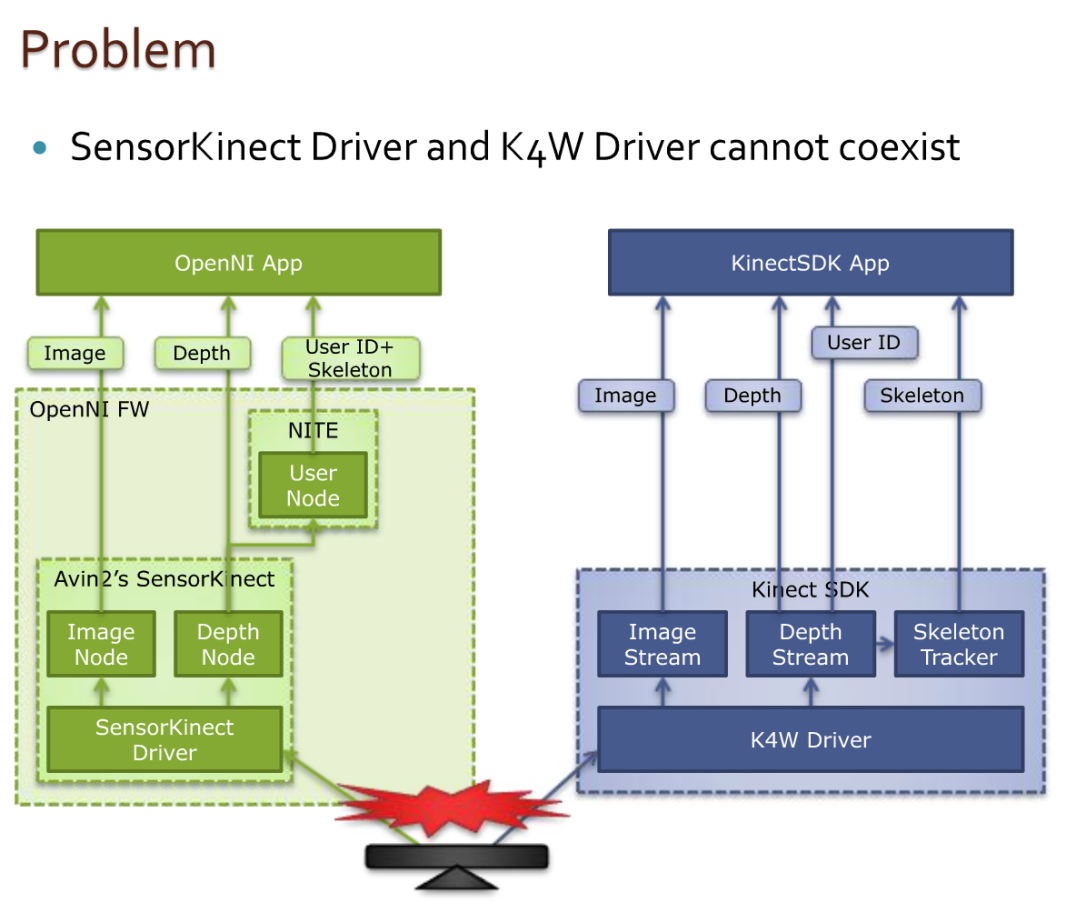
[RXTX for Java](http://mfizz.com/oss/rxtx-for-java) (DLLs for 64-bit machine)

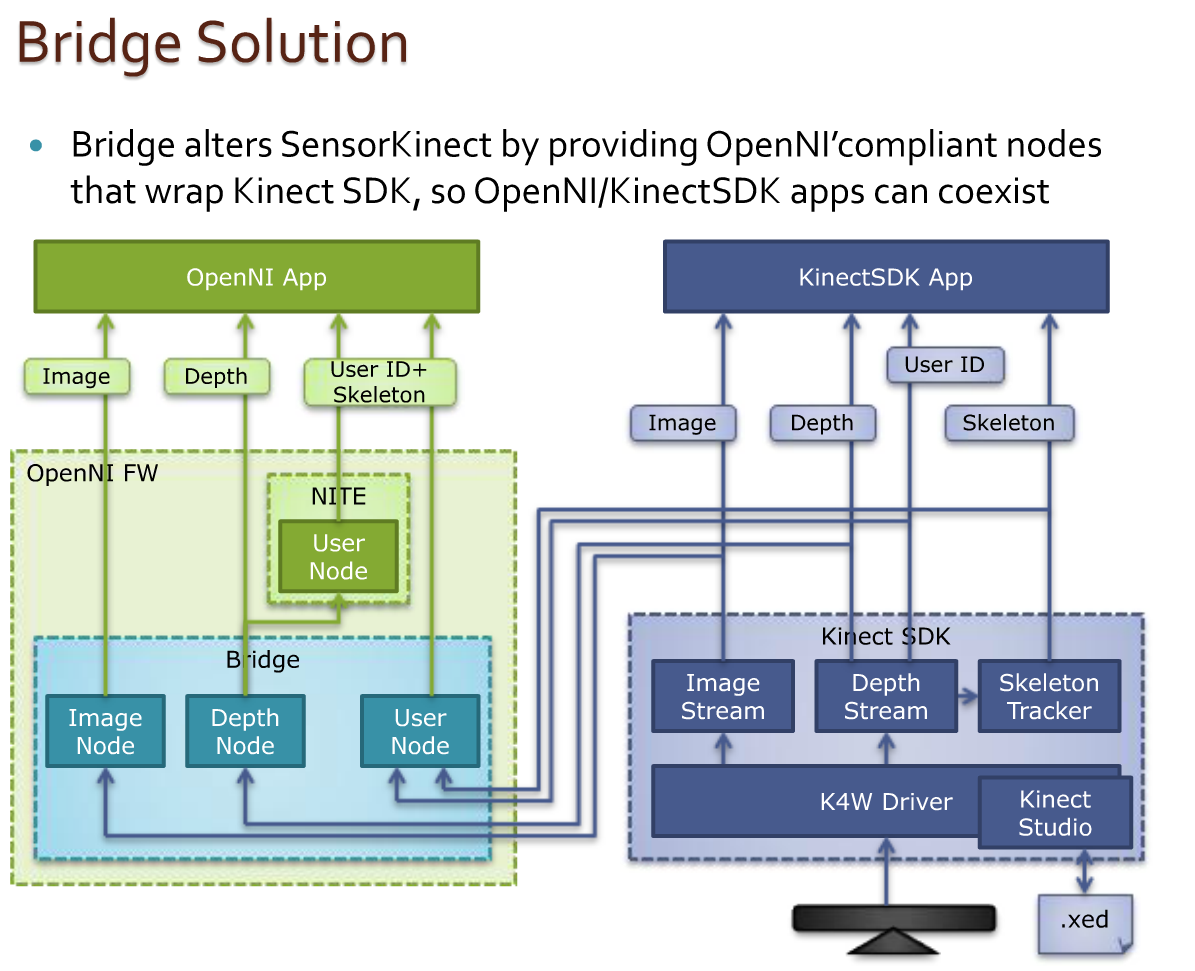
[Interfacing Arduino and Java](http://playground.arduino.cc/Interfacing/Java#.UwdVtfldVIE)

[Leap Motion Java Example from SDK](https://developer.leapmotion.com/documentation/java/devguide/Sample_Tutorial.html)

**DAY 2:**

* Further research on various Kinect development software. I also studied **OpenKinect’s Libfreenect** as this supports Processing IDE as well, which I am familiar with. However Libfreenect is mainly a driver which exposes the Kinect device's features like depth stream/IR stream/color(RGB) stream/motor control/LED control/accelerometer. It does not provide advanced processing features like scene segmentation, skeleton tracking, etc.
* As our iFarm application mostly requires high level abstraction like face tracking/eye tracking/voice recognition, Libfreenect is not in the scope of our interest.
* According to my findings on week1 day 1~2, OpenNI and MS Kinect SDK both have its own pros and cons. To maximize the benefits of both, using **MS Kinect-OpenNI bridge** was considered.
* A bridge called ‘**kinect-mssdk-openni-bridge**’ was developed in 2012.
* How the bridge works:



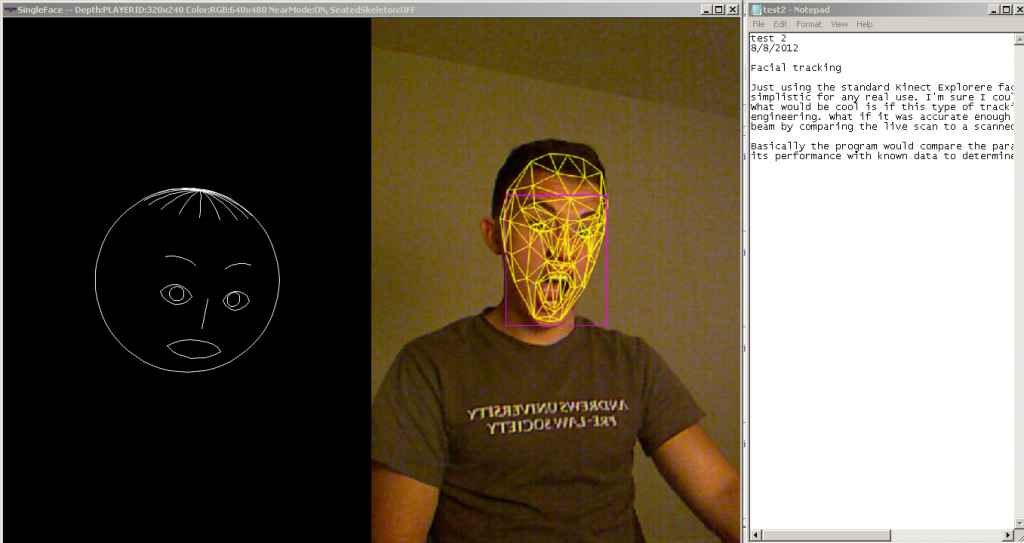


* With the help of this bridge, app can apply NITE’s algorithms (e.g. User node) upon Bridge’s Depth node. At the same time, app can also use Kinect SDK’s skeleton tracker through the User node bridge provides
* Use “query” on User node creation
* Bridge’s User node works by itself but requires no depth node
* App can use both User nodes at the same time if needed
* Compatible with recording/playback with Kinect Studio as Kinect Studio is totally transparent within Kinect SDK.

**DAY 3:**

* ‘Kinect-mssdk-openni-bridge-v1.6.0.0-for-1.5.2.23’ turned out not working on my PC. The installation failed due to some suspected reasons as follows.
* The bridge is for Kinect for Window, not Kinect for Xbox360. The one I tried with is Kinect for Xbox.
* The bridge was developed for OpenNI version 1.5.2.23 and it might not support the current version of OpenNI (which is v2.2). Most of articles and documents about this bridge are outdated (mostly created in 2012) and the developer seems stops updating his bridge since 2012.
* Learned interfaces of MS visual studio 2010 and ran some MS Kinect SDK examples in VS environment. It was to become more familiar with various APIs which MS Kinect SDK provides.
* MS Kinect SDK definitely overrides OpenNI+NITE in handling audio data, such as detecting audio angle and speech recognition. **Kinect for Windows Developer Toolkit v1.8.0** currently offers ample resources and samples of Kinect application. It provides face tracking capabilities, which is able to locate the position of eyes, nose and mouth and capture facial details like whether the mouth is open/closed and movement of eyebrows. However it does not detect the movement of eyelid and hence tell users whether the eyes are open or closed.





* Furthermore Kinect Fusion, one of the latest samples, offers various APIs to capture and reconstruct 3D model of any objects. However, my PC seems not meeting the minimum requirement of this application.



**DAY 4:**

* Had in-depth discussion with my supervisor about my findings so far.
* MS Kinect SDK seems to have much more resources and examples that I can refer to. Furthermore its **face tracking capabilities** and **voice recognition** can be incorporated with the iFarm application. Face tracking capabilities could tell whether the user is ready for the rehab exercise by looking straight at the screen. (How to determine whether user is looking straight at the screen, based on the position of eyes and face width need to be studied further soon) For voice recognition, it can be developed to listen to user’s command to start and stop the exercise, as well as move to the next exercise.
* Therefore I was advised to use MS Kinect SDK instead of OpenNI+NITE as this fits into our application better.
* I was hence required to get myself familiar with the interface of Visual Studio as well as C# language.
* Although it is possible to make leap motion communicate with Arduino in java, I was advised to use **node.js (javascript)** for several reasons.

1. This is one of the popular platforms and hence the application would be more scalable and easy for further improvement in future.
2. Ericko, the previous IO student also worked on similar project, making use of node.js server. Once the data is sent to the server it is saved in MongoDB. By using node.js I could make good use of what is already built and researched by Ericko.
3. Most of Leap Motion examples and resources are written in Javascript.

I hence decided to get familiar with Javascripte language, as well as Web socket communication.

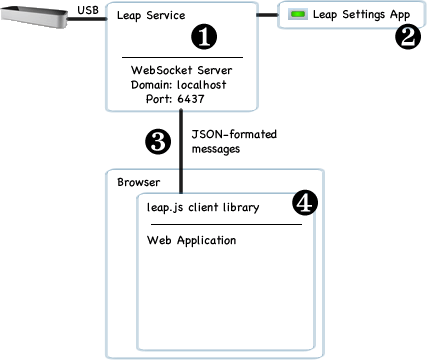
* In order to directly talk to the Arduino from Node.js, we need to upload the **Standard Firmata** as well as **johnny-five library.**
* Firmata is a generic protocol for communicating with microcontrollers from software on a host computer. The aim is to allow people to completely control the Arduino from software on the host computer.
* Johnny-Five is a library of JavaScript components that know how to talk to an Arduino through the “Firmata” protocol. To install it, open a command prompt or terminal window in the folder where you will keep your project code. Run the command

npm install johnny-five.

This will install the NodeJS package for Johnny-Five, making it available for any NodeJS files in that folder.

**DAY 5:**

* The Leap Motion software installed on any computer with a Leap Motion controller provides **tracking data through a WebSocket server**. The WebSocket server listens to port 6347 on the localhost domain ([http://127.0.0.1:6437](https://developer.leapmotion.com/documentation/skeletal/java/_static/JSONViewer.html)). Any client application, including Web clients, that can make a WebSocket connection can access the Leap Motion tracking data in the form of JSON-formatted messages. The WebSocket server is provided by the leapd process, which runs as a service on Windows and a daemon on OS X and Linux.



**IV) Future Works**

* Data acquisition from Leap motion
* Communication between Leap motion and Arduino
* Clear overview/architecture of iFarm application