**< WEEKLY REPORT FOR WEEK 8 >**

Name: Oh SuJin

Project: Multimodal Sensor Interfacing, Acquisition and Visualization

**I) Project Work Summary**

**Finished:**

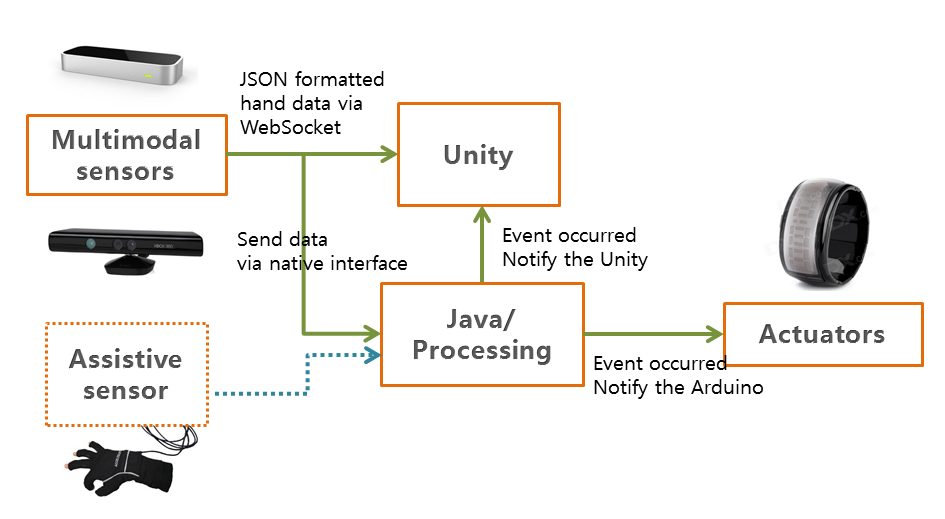
* Understanding System architecture of iFarm
* Understanding difference between Websocket and native interface communication
* Installing RabbitMQ for java-unity integration
* Adding more exercise & difficulty levels to simple iFarm GUI.

**II) Tasks Assigned**

**III) Detailed Activities / Accomplishments**

**DAY1&2**

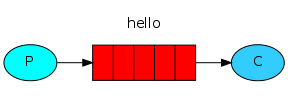
* System architecture of iFarm



* Understanding communication and its difference between Java-leap motion and 3D unity-leap motion.
* Leap motion controller provides tracking data not only by its native interface but also 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 (Leap motion team). In this project, Unity will directly access the Leap motion to obtain hand data for 3D visualization.
* At the same time, Leap Motion also feeds Java with its raw tracking data. DAQ, Feature Extraction and Assessment units running on Java platform will then process such data to notify both Unity and Arduino IDE for respective biofeedback.

DAY 3

* Java-unity integration
* For Java to notify Unity (C#) for every event occurred, a middleware which is able to connect each application as components of a larger application is required.
* **RabbitMQ** is a message-oriented middleware implementing the Advanced Message Queuing Protocol which enables messaging across different platforms (Pivotal). RabbitMQ is therefore chosen for the integration.



Once RabbitMQ is installed and libraries are imported (<http://www.rabbitmq.com/download.html>), it can accept and forward messages to the client.

**Producer (P)** here is Java, where strings of message are passed over to the queue whenever conditions are met.

In java, **a queue** is created which lives inside RabbitMQ. A queue is not bound by any limits, it can store as many messages as you wish functioning like an infinite buffer. Many producers can send messages that go to one queue, many consumers can try to receive data from one queue. However in this project, a simple one-to-one case is only considered.

**Consumer (C)** in this case is the Unity which waits to receive messages.

DAY 4&5

* Improving simple GUI created in week 7 by adding different types of activities and difficulty levels for respective exercise.
* **Activities of Daily Living (ADL):** An individual's ability to perform ADLs is important for determining what type of long-term care (e.g. nursing-home care or home care) and coverage the individual needs (i.e. Medicare, Medicaid or long-term care insurance) One of the key advantages iFarm has over other rehab applications is that it offers fun and complex daily activities such as cooking and brushing teeth as its ADL exercises. This provides therapists more concrete data to assess patient’s progress of hand recovery. ADL exercise will be visually simulated by 3D Unity, making physiotherapy routine more fun and practical.
* **Traditional hand rehab exercises:** Besides ADL exercises, iFarm also provides traditional grabbing, stretching and pinching exercises.
* Difficulty level for each exercise can be set from Beginner to Intermediate to Advanced. This is based on simple algorithm of calculating relevant fingertip speed and hence setting different speed limit for each difficulty level. If advanced is selected, higher minimum speed limit is set and hence one should carry out exercise above this speed.