Sensei: Sports Assistant

Daniel Jang, Debosmit Ray, Jin Choe, Matthew Reynolds

CSE 477, Spring 2015

**ABSTRACT**

.

**Author Keywords**

.

**ACM Classification Keywords**

.

**General Terms**

.

# INTRODUCTION

People play sports everyday and is a healthy part of your lifestyle. Our wearable devices will help you in sports. Senseiii is a device that will quantify your motion during a sports session. Whether it be basketball, tennis, golf, or badminton, Senseiii can help make your sports life easier. Senseiii is designed to measure your consistency, record the good shots and bad shots. It will deliver then a statistics to show the user how they are performing for a particular stroke. It also can be used as a scorekeeper for your sports. Although it is important for sports players to keep track of the score, people often forget the score of their game. Senseiii can do that for you so that you can focus on the game!

# BACKGROUND

The core of Senseiii is essentially made up of three components: the IMU (inertial measurement unit), RFduino, and Android application.

## RFduino

An RFduino is a finger-tip sized, Arduino compatible, wireless enabled microcontroller. It sports a ARM Cortex-M0 processor and has a built in Bluetooth 4.0 Low Energy module.

## Inertial measurement unit

The IMU (inertial measurement unit) is commonly used to measure the kinematics of a device. A 6-axis IMU uses a 3-axis accelerometer and a 3-axis gyroscope. The 3-axis accelerometer is a sensor that outputs the acceleration in the x, y, and z direction. Similarly the 3-axis gyroscope is a sensor that outputs the angular speed in the x, y, and z direction.

## Android

Android is a mobile operating system developed by Google. Android is used to provide interaction Senseiii. In addition with the mobile phone, it is used for its processing power. The android phone processes the raw values from the IMU and performs various algorithms to compute the results.

# IMPLEMENTATION METHOD

Sensei has three main components: the RFduino, IMU, and Android application. The RFduino communicates with the IMU via I2C. The RFduino communicates with the Android phone via Bluetooth. Therefore the system is connected from the IMU to the Android phone.

## Hardware

To measure the acceleration and rotation of the device we use the IMU and an RFduino. The IMU uses 16-bit ADC per axis of measurement. In its default configuration, the accelerometer has a scale of +/- 2g. For our intent, we need this value to be much higher. The trade off is precision. If we increase the accelerometer range, then we are essentially decreasing the precision of our accelerometer. The largest range that the IMU supports is +/- 16g. In order for this we need to set the AFS\_SEL for each axis to be the value 3. As shown in the figure X. According to the datasheet, values from -16g to 16g are stored as a 16 bit 2’s complement value. Its full scale is +/- 16g with a LSB sensitivity of 2048/g. This means that 2048 digital output is a unit of 1 g. For a 16 bit 2’s complement, the range is -2^15 to 2^15-1 or -32,768 to 32767. A value from the sensor can be directly translated to units of g using the unit conversion of 2048/g. So for example, if the sensor reads accelX = 6032, accelY = 8382, and accelZ = 2020, we can divide values by the LSB sensitivity constant 2048 to get the acceleration in each axis in units of g. accelX = 6032/2048 = 2.95g, accelY = 8382/2048 = 4.09g, and accelZ = 2020/2048 = 0.99 g. Since we are using 2 bytes per axis of acceleration, we will need 6 bytes of data from the IMU.

The IMU is paired with an RFduino. The main job of the RFduino is to read the raw values from the sensor, and encode the 6 bytes of acceleration data to a string and send it to the Android via Bluetooth. We obtained the data from the IMU via I2C. After successfully reading the data we verified to make sure it was correct. We graphed the values coming from the sensor and realized a potential problem. Speed was an issue. The frequency at which we are sending data severely impacts our ability to determine whether a frame of data is consistent to another frame. At first without any delays we were getting a frequency of about 25 Hz. In other words we were sending raw data 25 times per second. This wasn’t sufficient because if we are moving the device very fast, then the capturing of motion is not as smooth. We later optimized our RFduino code’s efficiency. With this change, we were able to send raw data at a rate of 80 Hz. This helped smooth our data graphs and improved our precision.

## Software

The Android phone handles most of the processing. When the motion button is pressed on the RFduino, the Android process the data until the button is pressed again. It decodes the data from the RFduino and stores it into a Motion Object. When Motion data is made up of an array of data coming from the accelerometer. Once the user has done a variety of motions, when the user swipes to the consistency tab, it calculates the score by comparing the motion objects that were created. The biggest problem with this is the method of comparing. How should you compare sets of data? Without some measure of warping the data, it is very hard because the data has a component of time that we do not care too much about during comparisons. A stroke or motion that is similar but starts later in time should be treated without factoring the time. The solution to this problem was dynamic time warping.

# RESULTS

# IMPROVEMENTS

# CONCLUSION

# REFERENCES

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Conference’10*, Month 1–2, 2010, City, State, Country.

Copyright 2010 ACM 1-58113-000-0/00/0010 …$15.00.

1. Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= <http://doi.acm.org/10.1145/161468.16147>.
2. Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. University of Maryland at College Park.
3. Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM, New York, NY, 526-531. DOI= <http://doi.acm.org/10.1145/332040.332491>.
4. Tavel, P. 2007. *Modeling and Simulation Design*. AK Peters Ltd., Natick, MA.
5. Sannella, M. J. 1994. *Constraint Satisfaction and Debugging for Interactive User Interfaces*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.