|  |
| --- |
| Project:Possibility |
| SunSPOT Framework for Language and Action Recognition (SUNFLARE) |
| A Developer’s Guide |

|  |
| --- |
| Winnie Yip  7/19/2008 |

A Developer’s Guide to

SunSPOT Framework and Language for Action Recognition (SunFLARE)

Table of Contents

[History 3](#_Toc204287819)

[Project Description 3](#_Toc204287820)

[Step 0 4](#_Toc204287821)

[SunSPOT Basics 4](#_Toc204287822)

[Getting the files from repository 4](#_Toc204287823)

[Setting up the projects in NetBeans, Compile and Run 4](#_Toc204287824)

[Setup Hibernate (Persistence) 5](#_Toc204287825)

[How to use the software 5](#_Toc204287826)

[Packages 5](#_Toc204287827)

[Server 5](#_Toc204287828)

[Getting Accelerometer Data 5](#_Toc204287829)

[Recognizing a BasicGesture 5](#_Toc204287830)

[Classifying a BasicGesture 6](#_Toc204287831)

[Classifying a Gesture 6](#_Toc204287832)

[System state control 6](#_Toc204287833)

[Plugin 6](#_Toc204287834)

[Gui 6](#_Toc204287835)

[Persistence 6](#_Toc204287836)

[Service 6](#_Toc204287837)

[Utils 7](#_Toc204287838)

[Future Tasks 7](#_Toc204287839)

[Bug List 7](#_Toc204287840)

# History

The purpose of this document is to provide a roadmap for developers who will be continuing our effort in the SunFLARE project. SunFLARE was one of the five projects that were developed in the Semester Project Pilot Program in Spring 2008. We had 6 team members: David Woollard (Team Lead), Irina Abramova, Praveen Kansara, Sean Bachelder, Nikhilesh Kruthiventi and Winnie Yip. The team delivered a full-functional, working piece of software by the end of the semester. There are a lot of improvements can be made, and this document will guide you through the design and implementation of SunFLARE.

# Project Description

Our project concept is to use the wireless capabilities and accelerometers of the SunSPOT platform to record and recognize human gestures for command and control applications. Users with limited mobility often have difficulty using traditional computer input devices such as keyboards and mice. Our gesture recognition framework, which we call the SunSPOT Framework and Language for Action Recognition, or SunFLARE, will allow users to dynamically capture movements that they are capable of making (tailoring to a number of physical disabilities) and associate these movements with actions.

We envision SunFLARE as an extensible plug-in framework that will allow developers to add control functionality particular to applications. Examples of the extensible applications that could take advantage of our framework (some of which we plan to develop as part of the semester project) include: manipulation of an on-screen keyboard, development of limited mobility games, and even motor control/physical therapy applications.

Not only will this project benefit the disabled community, but also it can benefit general users interested in associating a particular movement with action. Though out of the scope of our semester project, our gesture recording and plug-in framework could be used to rapidly develop motion response applications for SunSPOTS.

# Step 0

## SunSPOT Basics

1. Install Netbeans IDE 6.0 or later, JDK, all that good stuff
2. Download Tortoise SVN
3. Install the SunSPOT SDK from the CD-ROM that came with the kit. The instruction manual is a good source of troubleshooting installation problem. It will require you to install ANT as well. You should also have the SunSPOT manager installed. <http://www.sunspotworld.com/SPOTManager/>
4. Hook up your SunSPOT base station to your computer. Launch SunSPOT manager and it should recognize the base station. There’s a drop down at the top where you can select which SunSPOT to talk to in case you have multiple of them connected. The dialog box in the manager tool gives you a detailed walk through of how to use the tool. For your base station, after you click on the ‘Upgrade’ button, you should also click on ‘Install Base Station’. You will only need to do the Upgrade step for your wireless SunSPOTs.
5. If you click on the SDKs tab at the top, you can get the latest version of SunSPOT SDK. We used v3.0 Purple. If you have an older SDK installed, it will automatically moves everything from that folder to a sdk-old folder.
6. You will notice there’s a “Demos” button at the bottom right under the SDKs tab. If it is disabled, you need to do a trick to make it enabled. You can click on the Preference tab, change the Network Timeout to some other hours, then navigate back to the SDKs tab. The Demos button should be enabled now and you can download updated demo code.

## Getting the files from repository

1. Our repository is currently in http://www.projectpossibility.org/svn/SunSPOT/
2. Make a folder on your desktop, call it SunFLARE, right click on the folder and click SVNCheckout. Copy and paste the link to the repository to the “URL of repository” field in the dialog box. Click OK. It will start downloading the entire source tree.

## Setting up the projects in NetBeans, Compile and Run

1. Open NetBeans. File->Open Project-> Select C:\Program Files\Sun\SunSPOT\Demos\TelemetryDemo\SunFLARE\Telemetry-onSpot
2. File->Open Project-> click on the SunFLARE folder that you just created. It will show up as “SunFLARE-GestureCreator” in your Projects window.
3. Compile both projects.
4. Connect a wireless SunSPOT to your computer. Make sure you have upgraded the firmware on it using the Manager Tool.s
5. Right click on the build.xml of the Telemetry-onSpot project -> Run Target -> deploy
6. Unplug your wireless SunSPOT and hook up the base station. Right click on the build.xml of the SunFLARE-GestureCreator project -> Run Target -> host-run

## Setup Hibernate (Persistence)

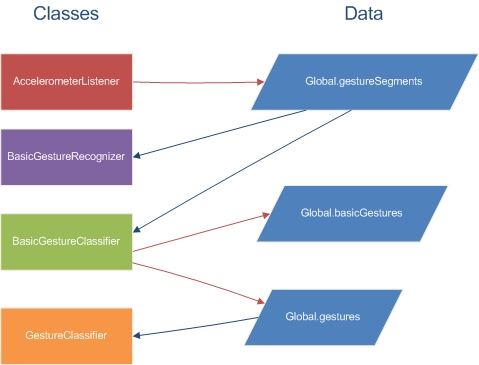
1. Put hibernate3.jar in your C:\Program Files\Java\jdk1.6.0\_05\jre\lib\ext

## How to use the software

[How to use the Gesture Creator GUI, to be added]

# Packages

## Server



### Getting Accelerometer Data

The remote SPOT sends packets of accelerometer data to the base station. The accelerometer listener has a recognize() method, whose purpose is to parse the data into useful segments. The dataset vector in AccelerometerListener.java is a placeholder for the data collected during the period of time when the wireless SPOT is moving. The recognize method uses the totalGThrehold to filter out the noise and just put the useful datasets to Global.gestureSegments vector. The Global.gestureSegments vector has all the raw data that you’d need for analyzing what gestures are being performed.

### Recognizing a BasicGesture

The BasicGestureRecognizer looks at Global.gestureSegments vector and sees if there’s any new dataset that has not been processed. Its recognizer() method creates a BasicGesture out of a dataset and puts it in Global.basicGestures vector. Note that we temporarily treat each dataset as an individual BasicGesture for now, but it is usually not the case because we cannot rely on the totalGThreshold that we put in our AccelerometerListener 100%. We need to check the timestamp of the last data element in each dataset. If it is close enough to the timestamp of the last BasicGesture, we need to combine the two BasicGestures. BasicGestureRecognizer also calculates which one is the active axis. If there’s a lot of activity in all 3 axes, we mark it as a Shaking gesture. But if there’s more activity on one axis than the other two, we’d mark it as the active axis. For example, if the user is moving the SunSPOT to the left, the sum of acceleration detected in the x-axis will be a lot greater than that in the y or z-axis.

### Classifying a BasicGesture

After a BasicGesture is created, we need to identify what movement the BasicGesture represents. The BasicGestureClassifier serves this purpose. The classifier() method determines if a BasicGesture in Global.basicGestures vector is ready to be classified. The reason of having this method is there’s a chance that the BasicGesture is not finished, it needs to be combined with the later BasicGesture (This process is described in the Recognizing a BasicGesture section). The BasicGestureClassifier decides what a BasicGesture represent (Left, Right, Up, Down, Forward, Backward, Shake). The classifyBasicGesture() method either creates a new Gesture object and adds it to Global.gestures vector or it appends the BasicGesture to the last Gesture in Global.gestures.

### Classifying a Gesture

Similar to the strategy we used in classifying a BasicGesture, the GestureClassifier class looks at Global.gestures and sees if there’s any Gesture that is ready to be interpreted by using the timestamp threshold and the preset time allowance in between Gestures. It makes a call to the Global.gestureDB.search() which searches the local database for a match. If SunFLARE is run in service mode, it will execute the plugin that is associated with this particular Gesture.

### System state control

This is done by the Controller class. We need a middle layer between the GUI and the server to keep track of system states (SYS\_RECORDING\_MODE, SYS\_GESTURE\_RECORDED, SYS\_TEST\_GESTURE, SYS\_SAVE\_GESTURE, SYS\_IDLE). The GUI can call the Controller methods to notify the backend what state is the system in based on user input to the GUI, and the Controller takes care of all the cases. For example, if the state is SYS\_SAVE\_GESTURE, the Controller calls a method of the Global.gestureDB to add the recorded Gesture to the database.

## Plugin

[To be added]

## Gui

[To be added]

## Persistence

[To be added]

## Service

[To be added]

## Utils

[To be added]

# Future Tasks

1. Caching the database. Currently dipping into the database is quite costly, and preloading some if not all Gestures when the third party application starts up is definitely something that we should do. This should be pretty easy to do.
2. Calibration. There are some predefined thresholds and those should be configurable.
3. A better Gesture Recognition algorithm. Right now it is quite primitive, it only finds the axis of with most activity, then look at whether the absolute max occurs before or after the absolute min. You will get a better understanding if you observe the pattern of g’s by running Telemetry-onDesktop demo.

# Bug List

It’s been a while since I ran SunFLARE, and I remember it was stable enough to run a demo. However, I believe there’s a glitch somewhere – when we started the demo fresh and recorded a gesture, it said the gesture already exists. I believe it has to do with the persistence. We did not try to reproduce the bug.