

Status Update 3/26

SunFLARE Project

Architecture

The final architecture for the SunFlare system is a client-server architecture; an architecture dictated by the relationship between the SunSpot sensor platform (host of the client) and the PC with base station (host of the server). The server software is a classic three-layer architecture in the vain of Model-View-Controller (see Figure 1).

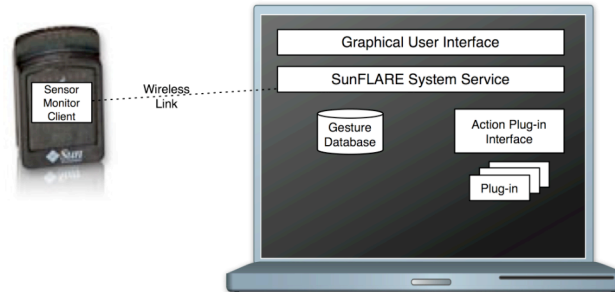


Figure 1: SunFLARE High Level Architecture.

Interaction between 3rd party applications and the SunFLARE service is captured in Figure 2. The Sequence of events for this figure is as follows: For start-up, (1) a 3rd Party application instantiates the SunFLARE System Service (possibly by passing a listener); (2) the System Service gets registers call back class names from the DB; (3) the System Service uses a factory to load call-back classes. For IDing gestures, (4) sensor data is returned from the SunSPOT; (5) the System Service queries the DB to identify gesture and get call-back info, and; (6) the SPI executes the call-back, triggering a 3rd Party application event.

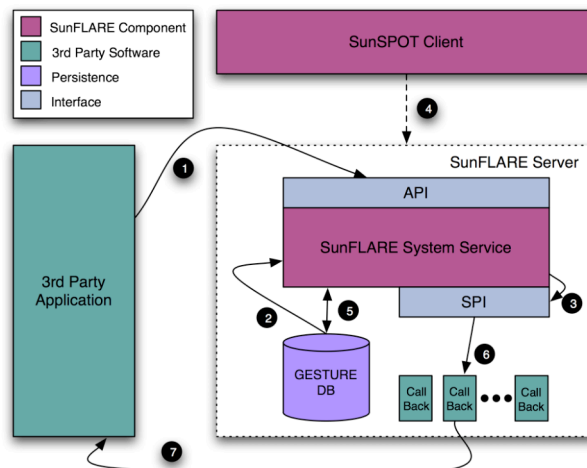


Figure 2: Dataflow diagram.

Development Status

Gesture DB

Hibernate has been set up to use Apache Derby to persist objects. This approach emerged as the most flexible of possible persistence strategies from the trade study (see our Wiki). The benefits of this approach are that Derby can be set up to run in the same JVM as the SunFLARE system service (making a small impact on setup and instantiation for its DB capabilities), while Hibernate allows us to annotate objects at the language level without concern for the mechanics of DB access. The process for setting this subsystem up will be published to our Wiki in order to capture necessary information of the packaging of the eventual SunFLARE system.

GUI

We have moved forward to develop base GUI functionality. This includes the windows for which we presented mockups in PDR. Currently, we are working to develop both the basic screens as well as interface the prototype gesture recognition algorithm to the GUI in order to display recognized gestures for prototyping purposes.

Plug-In Architecture

We have developed the base code for the system-level interfaces (the interfaces between each of the drawn components in Figure 2) as an initial task on this subsystem. Additionally, we are in the process of refactoring the SS12 game to include a SunFLARE listener as part of its user input classes. We plan to test the game with simulated data this week.

Algorithm

We have worked to identify and reconcile two major questions regarding the gesture recognition algorithm. The first is an issue of coordinate systems. While the SunSPOT uses a local coordinate system, this system might not match the work coordinate system. This might be an issue in the event that a change in SunSPOT orientation might alter gesture recognition. We don't think false positives are likely considering gravity will always act in the world coordinate system, though false negatives are possible. For the time being, we decided that we would address the issue as it came up and our first-line mitigation strategy would be to instruct the user to hold the SunSPOT in a particular manner (as time permits, we may be able to relax this assumption).

The second question is a matter of knowing the starting and stopping point of gestures. We have developed an algorithm for windowing gesture information and have further reduced the signal information through the use of the first derivative as a metric of signal areas of interest. We plan to show a demo of simple gesture recognition using this approach at CDR along with the first iteration of the GUI.