Spike Plan 1
Name: Firebase/SerialPort based communication spike.

Context:
For the upcoming project to create a morse code decoder, we need to decide which frameworks for communication are appropriate on both the server-client and server-IoT directions.

Gap:
We hope to understand the consequences of our choice of client-server communication method, especially one where another intermediary server is involved (Firebase). We should take care to see how much of a delay this server introduces, and the reliability of the server (making sure that no data is lost, etc.). We should also make sure that coding the IoT-server connection at a lower-level using SerialPort allows us some clear advantage (lower latency, for example) over using a higher-level library (such as JohnnyFive). We should make sure that using SerialPort results in relatively stable, easy to understand code.

Goals/Deliverables:
The main goal here is to discover how difficult creating each of the two communication directions (server-client and server-IoT) is, and how difficult it is to integrate the two.

- A `server.js` file that will simply push the time of motion start and end events to firebase. It will determine when motion starts and ends using SerialPort to communicate with the IoT device.
- A `client.js` file that will read from firebase and output the latency between actual motion event and display time in the client.

Planned start date: Apr 8, 2017
Deadline: Apr 21, 2017

Planning notes:
First, the server-IoT communication will be established using SerialPort. Once this is working and the server is capable of printing the motion on and off events to the console, this information will simply be sent over firebase to the client. Finally, the client side code will be written to display the event, including the latency.