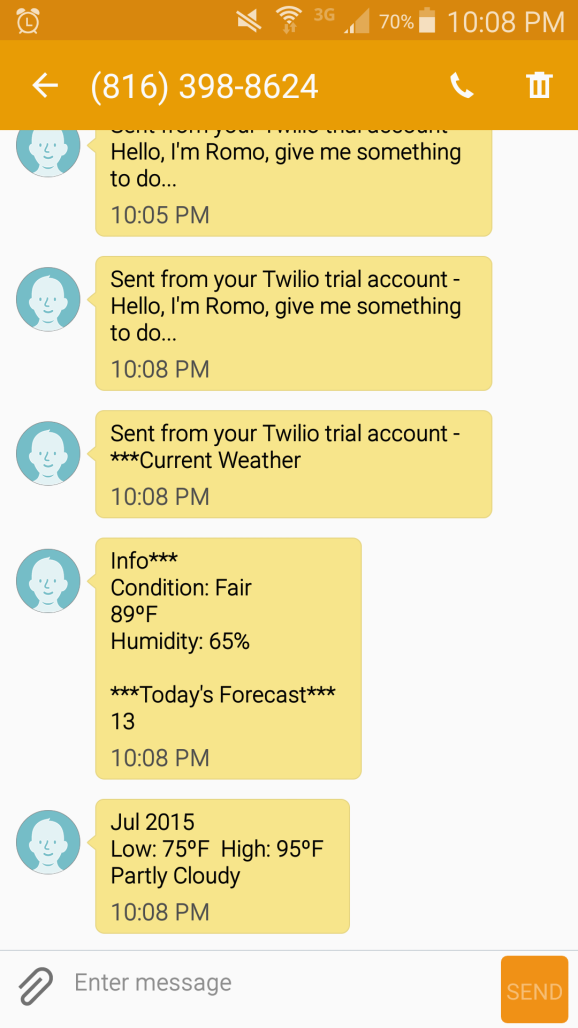
# Project Increment 2

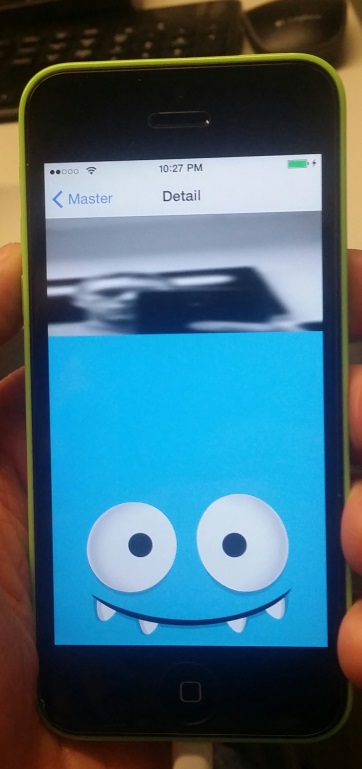
GitHub Repository: <https://github.com/jeff-umkc/PG6_Iteration2>

For the second iteration we wanted to implement the following: speech out, enhanced movement, updated Romo UI, and integrate Big Data into the project. In alignment with NightBot’s purpose, the Yahoo! Weather API was added to satisfy the speech out requirement. We believe that imminent weather can be a safety concern. From the Android client, the user is capable of saying “weather” and Romo will reply by sending the user a text message via the Twilio API. The text message will contain the current weather conditions as well as the daily forecast. There are 49 weather codes such as windy, fair, and heavy snow to give the user a general idea of the current weather outlook.



**Speech Out**

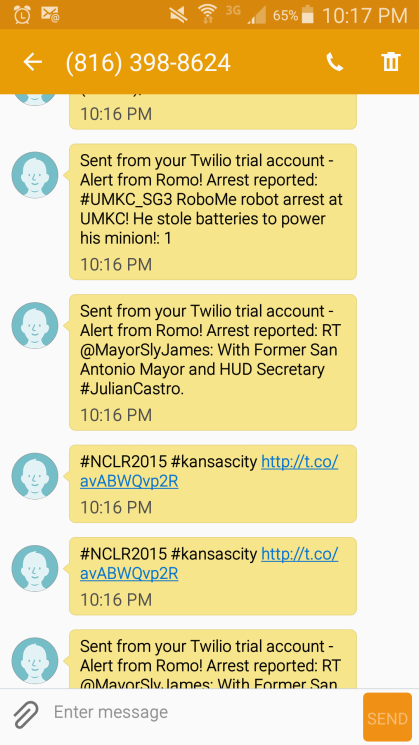
Previously, our Romo maintained the UI from the stock Romo robot. However, we thought it would be beneficial to split the view. We wanted to do this to get a better idea of what the camera sensors were looking at for object detection. Otherwise, we may not know precisely if the object that we are trying to track is in the frame.



**Romo UI**

In iteration 1 we had simple controls for our Romo using the Android client. The teaching assistants have provided the class with an accelerometer based control using the Android client. This has been consolidated into NightBot. Ideally, this will be much more responsive. We had aspirations and the Romo performing line-following based on object detection, but further research will be required to get it working.

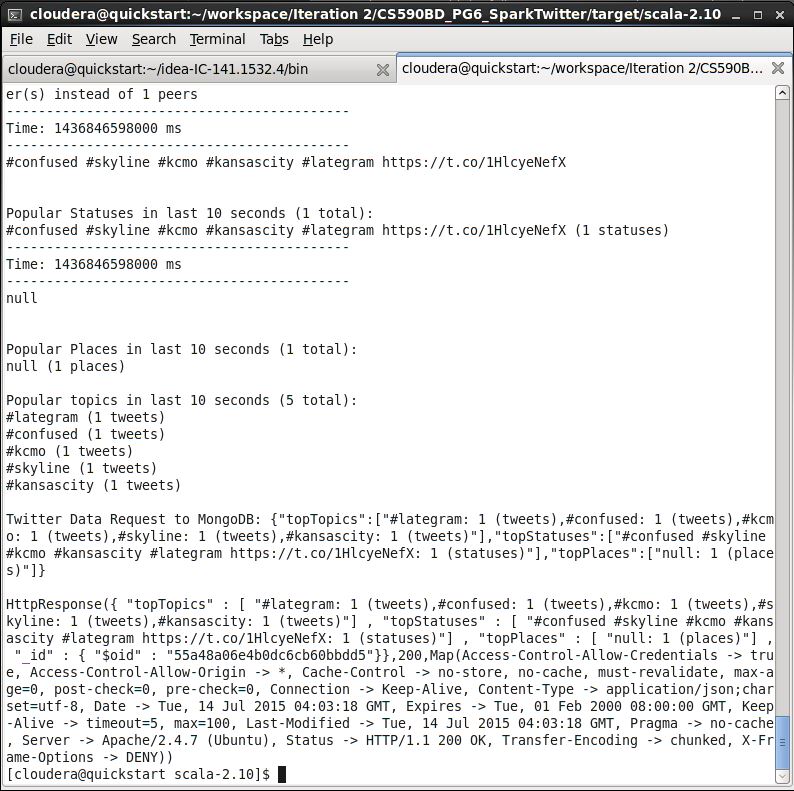
Finally, for the Big Data portion, we thought it would be best to provide the user with crime information from Twitter. #kcpolice, #kcpd, #kcmo and other hash tags were utilized to find relevant information on Twitter. In Spark, a MapReduce job was created with Scala to get the top 10 tweets, places, and hashtags. All of which were filtered on keywords related to Kansas City and crime. To accomplish this, the streaming API for Twitter was used and it continuously retrieves data every two seconds to stay current. Next, we created a MongoDB and stored the data in JSON format. Finally, the data was sent to the iPhone using a REST API. The information is communicated to the user via text message and the Twilio API.



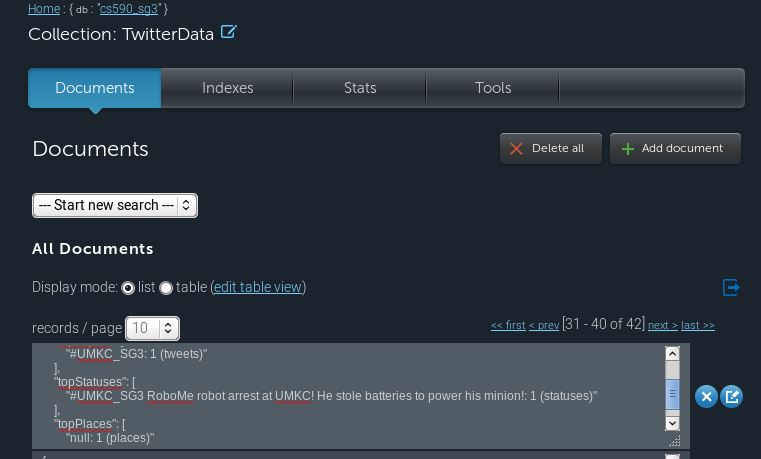
**Twitter Output**



**Spark (picture 1 of 2)**



**Spark (picture 2 of 2)**



**MongoDB**