**CS4560 Fall 2016**

**Project Name: RoboCat**

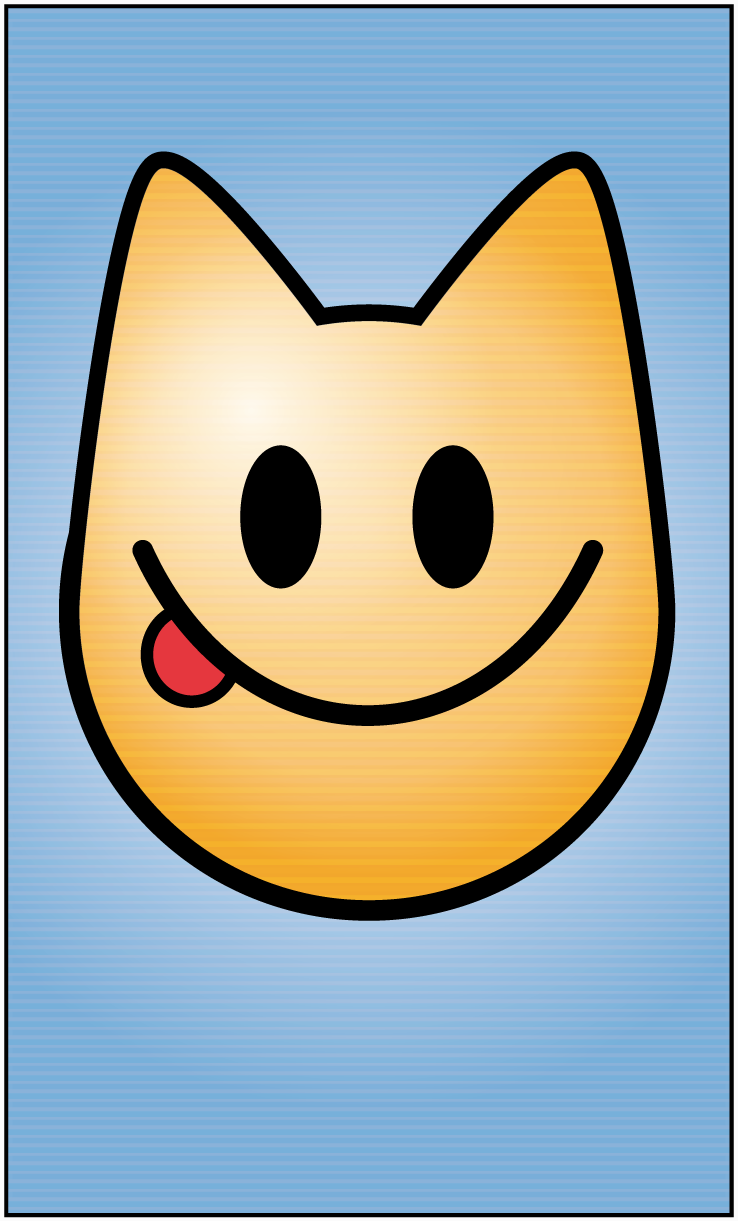
**Team Name: AKUnited**

**Team Members:**

Alex Vild, Luke Shays, Quintin Fettes, Seattle Ruzek, Zach Taylor, Kurtis Davis

**Clients:** Dr. Jim Zhu and Miguel Sempertegui Sosa

**Instructor:** Dr. Chang Liu

**TA:** Yunyi Feng

**Content**

1. Project Requirement
   1. Overview
   2. Use Cases
   3. Actors
   4. Use Case Diagram
   5. Use Case Details
2. Technology Stack
   1. Diagram
   2. Trade Offs
3. Design
   1. Purpose-Concept Map
4. System Architecture
5. Implementation Notes
   1. Source Code Documentation
6. Known Issues
7. Links for Project
8. Appendix (Initial Informal Client Requirements)
9. **Project Requirement**

*Overview:*

RoboCat is a biomimetic robotic cat controlled by RoboApp running on an Android smartphone. When RoboCat is in use, meaning that the phone is running RoboApp and is attached to the body of the RoboCat, he should be able to respond to various external stimuli. He is able to understand commands through active listening, and is able to follow faces through facial recognition. He responds to voice commands using natural language processing by the image of the cat on the Android phone screen changing emotions (smiling, frowning, etc).

*Use Cases:*

1. **Listening**: RoboCat is being spoken to and is actively listening for a command.
2. **Searching**: RoboCat is looking for a point of interest (either a person or a color).

*Actors:*

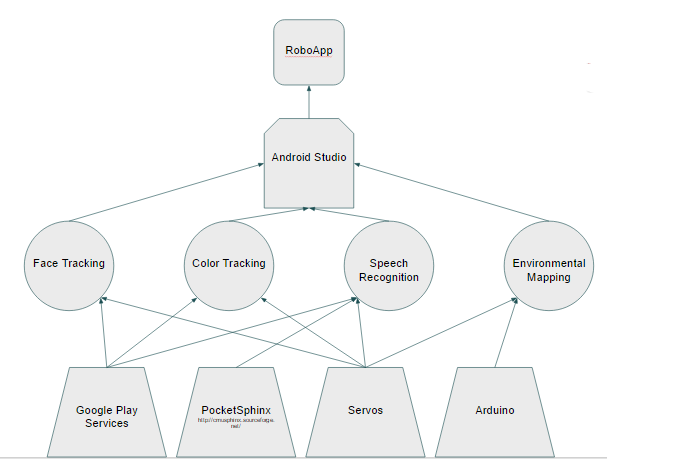
1. RoboCat
2. User

*Use Case Diagram:* 

User RoboCat

*Use Case Details:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Participating Actors** | **Pre Condition** | **Flow of Events** | **Post Condition** |
| Listening | Robocat, User | User gives RoboCat a command. | PocketSphinx interprets the verbal command into a string and determines the appropriate action. | RoboCat responds to the command. |
| Searching | Robocat | None | The facial or color tracking technology in RoboCat detects a face or color in his environment. | RoboCat follows the face or color using the servos motors that control his neck. |

**II. Technology Stack** 

The entire project is controlled by RoboApp running on an Android phone [<https://www.android.com> ]. **Android Studio** is used to write the code for all of the functionalities of RoboApp [<https://developer.android.com/studio/index.html> ].

Face tracking is a key part of RoboCat because it allows him to recognize and respond to the presence of people, similar to a real cat. Google play services are used to control Face Tracking, specifically the package **com.google.android.gms.vision** [<https://developers.google.com/android/reference/com/google/android/gms/vision/package-summary> ], and the Servos motors are used to turn his head to follow faces as he finds them.

Color tracking is done in a similar way, Google play services are used to implement color tracking, **android.hardware.Camera** [<https://developer.android.com/reference/android/hardware/Camera.html> ].

Active listening for speech recognition is implemented with **PocketSphinx**. PocketSphinx is a technology that is used to replace the Google speech command button in order to implement active listening [ <http://www.speech.cs.cmu.edu/pocketsphinx/> ].

Environmental mapping will be implemented using the **Arduino** board [<https://www.arduino.cc/en/Main/ArduinoBoardMega2560> ] which will be connected to the **ultrasonic sensor**, called Ping [<https://www.arduino.cc/en/Tutorial/Ping> ].

We traded OpenCV [<http://opencv.org> ] for the Google package com.google.android.gms.vision, this package simplifies facial tracking and recognition in a way that wasn’t possible with OpenCV. Another trade off we made was replacing the Google command button with PocketSphinx to implement active listening.

**III. Design**

*Purpose Concept Map:*

Ovals represent the purposes and rectangles represent the concepts (hierarchical).



*Narrative:*

The purpose of RoboCat is to behave like a regular cat would in different circumstances. We use the three main concepts described above to complete this goal: facial recognition, speech recognition, and environmental mapping.

**Key Concepts**: Interaction based on voice commands, movements, and the environment.

*Pre-existing concepts:*

* Facial recognition
* Responding to voice commands
* Reacting to environment through color tracking

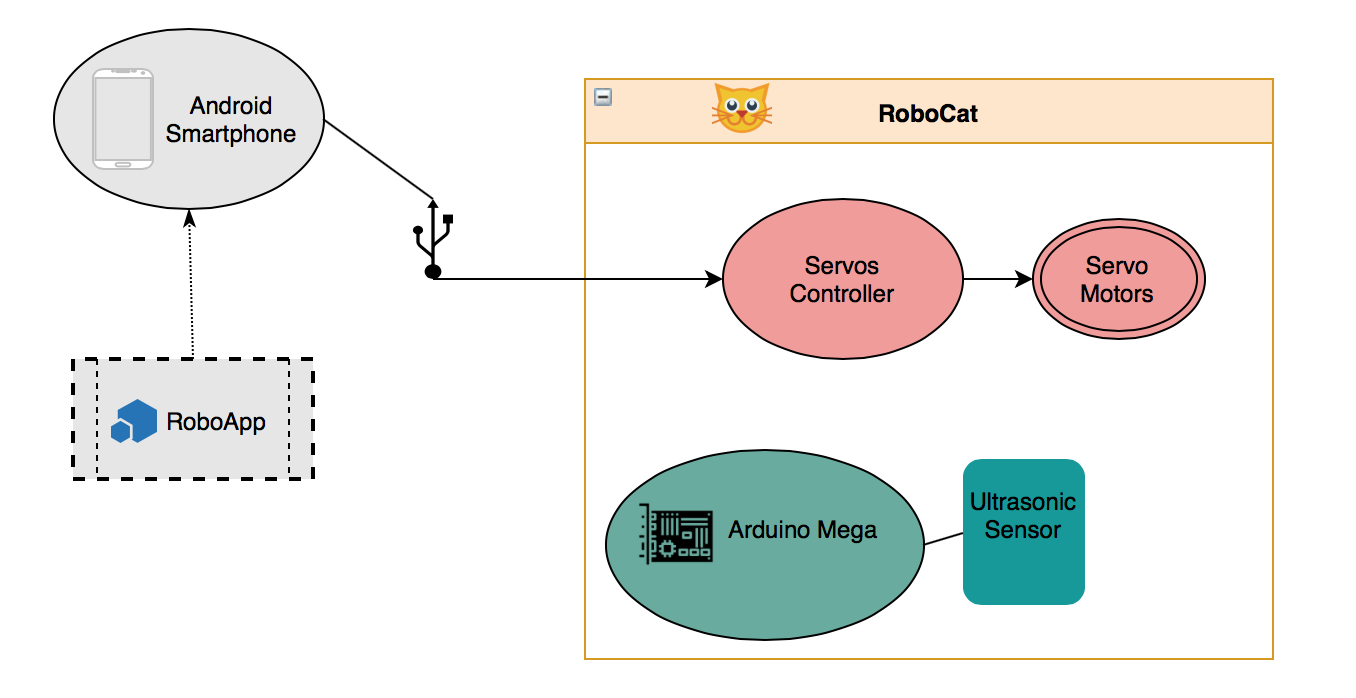
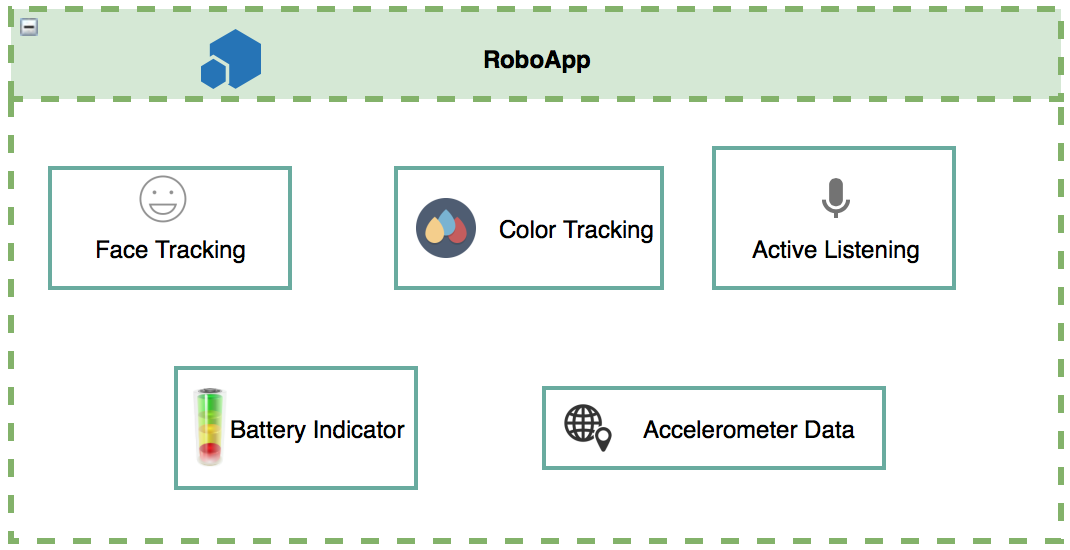
*Analogic Concepts:*

The camera and the microphone act as the eyes and the ears of RoboCat.

*Synthetic Concepts:*

Since RoboCat is unable to move at this point, we cannot implement the concepts that require a moving cat, such as following a user, or navigating through the environment.

**IV. System Architecture**



The Android smartphone that is running RoboApp is attached RoboCat’s head (through USB) and used to recognize commands through PocketSphinx (active listening), and the front camera is used to “look” at and recognize faces of users. The Android sends commands to the Servos controller based on the information received by the app, and then the controller moves the Servo motors in order to mimic the motions of a real cat.

The Arduino mega will be used as a bridge between the Servos controller and the Android so that there will be a line of communication between the three different technologies. The ultrasonic sensor is what we decided to use to collect data from the objects surrounding RoboCat in order to begin mapping the environment.

**V. Implementation Notes**

See GitHub: the code that was added to the RoboApp repository is all fully documented.

The most significant files that were changed this semester were:

* RoboApp/app/src/main/java/com/robodoot/dr/RoboApp/FdActivity.java : main file that controls Facial and Color Tracking, accelerometer data collection, battery indicator implementation.
* RoboApp/app/src/main/java/com/robodoot/roboapp/MainActivity.java : another main file that controls the menu, and the actions generated by the button selected.

Other file that was touched to remove OpenCV:

* RoboApp/openCVLibrary300/openCVLibrary300.iml

In order to change the target device to compile:

* RoboApp/app/src/main/AndroidManifest.xml

Link to Read Me for the project:

* https://github.com/kd345312/RoboApp/blob/master/README.md

**VI. Known Issues**

* The Arduino board is not yet connected physically to the body of RoboCat.
  + Code that is written for the ‘Android to Arduino to Servo’ line of communication is hypothetical and can’t be tested.
* Active listening through PocketSphinx has been merged into the master branch, however it only works with certain Android devices. Known working devices (Galaxy s5 and Google Pixel).
* The front camera on older Android phones only captures images at 15 fps, so RoboCat can’t follow faces if the user is moving too fast.
  + **FIXED: Updated to track faces at 30 fps and it works much better now with faster moving users.**
* Color tracking is still a standalone app and needs to be merged into the master branch, and also needs to receive input from a video feed (currently only captures input from still pictures).
  + **FIXED: Color Tracking is merged into master branch and now tracks color via video feed.**
* Google removed RoboApp from the Android App Store due to a security vulnerability related to a new version of SSL that was released while we were developing.

**VII. Links**

**RoboApp source code:** <https://github.com/kd345312/RoboApp>

**Android Studio:** <https://developer.android.com/studio/index.html>

**Arduino Mega:** <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>

**Arduino Ultrasonic Sensor component:** <https://www.arduino.cc/en/Tutorial/Ping>

**Servo Controller:** <https://www.pololu.com/category/12/rc-servo-controllers>

**Google play services:** <https://developers.google.com/android/guides/overview>

**App on Google App Store**: can’t provide link at this time because it is not released, due to the SSL security vulnerability. Will fix this by next semester.

**VIII. Appendix**

*Initial Informal Client Requirements* (for the CS team):

* Continuous speech recognition.
* Modify face recognition for shape and positioning.
* Add a range sensing mechanism.
* Environment mapping.
* Check battery status.
* Log data from accelerometer.