Table of Content

1. Background
2. Goals
3. Special Note
4. Current System Configuration
5. Proposed System
6. Details about the setup and installation
   ● Apt-Get
   ● MJPEG Server
   ● iOS Development
   ● Firebase
7. Experiments
8. Conclusion
9. Future Idea
10. References
Background

The CS Robotics team have team of Corobots that look like this.

Corobot is a self-navigating robot that can interact with people both locally (through a mounted netbook or tablet) and remotely over the Web. You can read more about the project here. [https://www.cs.rit.edu/~robotlab/corobots/](https://www.cs.rit.edu/~robotlab/corobots/)

They have a laptop mounted on top that runs the ROS (Robot Operating System). The laptop is connected to two cameras on sides and a Kinect mounted below it. The two side cameras help the robots with localization in the Map. There are QR codes all around the CS floor that are being used by these Corobots.
The above is the third floor of the Golisano building CS Floor Map. All the development are done on this map as these Corobots are not made for outdoor environment and do not like stairs. The Microsoft Kinect has infrared inbuilt that helps in detecting obstacles and their distances.

The Corobots are pretty powerful Robots and can be used for development and deployment by the developers. There is a guide for launch instructions (Check Slack Guide) that lets you launch the Corobots. Once the system is active it lets the web server vhost1.cs.rit.edu know that it is good to go. You can view the status of these Robots at vhost1.cs.rit.edu/status.php

To non-developers, there are limited means of communication with the Corobots. Once the robots are live, it would be more interactive for end users to watch real time the movements and habits of these robots. Also different apps can make the same robot could behave differently.
Goals

In this project we are interested in creating an iOS app that lets user watch the stream of the Kinect of the Corobots, View status of Corobots and localize himself/herself as using QR code.

The scope of this project is to develop a native iOS app that can stream video over the network, localize itself in the map and display feeds from the network server about the robots around them. Google firebase provides authentication services for the project.

Special Note

All special notes like username/password, etc. are attached to Trello and Slack updates. Make sure you check them for any updates about the project. All files of this project are uploaded on Slack.

Current System Configuration

Working on Corobot #6

Currently running ROS Groovy on Ubuntu 12.10

IOS Development version: 10

Language: Swift 3

Firebase version: 3
Getting started

Read ROS tutorial to understand some of the basics of the ROS system. You can follow this guide. [http://wiki.ros.org/ROS/StartGuide](http://wiki.ros.org/ROS/StartGuide)

For best practices, use Slack and Trello for updates on the project. There is a file corobot_readme.pdf file with instructions on how to use these tools.

Proposed System

Robot Operating System (ROS) has lot of packages which are essentially building blocks of ROS system. The different packages deal with different task. Nodes are runtime processes. Different packages create different nodes for different tasks. There is a Master node which communicated with other nodes and manages them all.

The Architecture diagram below is used for this project. We are interested in the video feed from the Kinect, communicating with the webserver and watching movements of the Robots real-time in the Map. The vhost1.cs.rit.edu web server keeps tabs on all the Corobots. The iPhone interface will communicate with this web server to show overview of the activities of these robots.

The ROS node mjpeg_server will stream image topics from the Robots. This stream is captured in the app and live camera feed from the Kinect can then be viewed on the iPhone. The app is able to use the iPhone camera to decode RIT CS Floor QR Codes and localize itself in the CS Floor map. Firebase services provide authentication and real time location services of the user.
Details about the setup and installation

Apt-get

Our solution involves first installing a ROS package on Corobots using apt-get. First make sure you have apt-get, you can check it using

dpkg-query -l apt-get

If you do not have apt-get, install apt-get using Ubuntu Software Center.
**Mjpeg Server**

This is a package found on ROS [http://wiki.ros.org/mjpeg_server](http://wiki.ros.org/mjpeg_server). For the video streaming from the Kinect we are going to use this package. Description is displayed below.

Check if your machine has this node installed, if it doesn’t follow the steps below. Install `mjpeg_server` by using instructions below in terminal. This will help us setup stream Image Topics of Kinect feed over the network.

```bash
sudo apt-get install ros-groovy-mjpeg-server
```

For development purposes, use ports between 7000 and 8000. I randomly chose 7555. You can then launch it by

```bash
rosrun mjpeg_server mjpeg_server _port:=7555
```

You can now view the kinect feed in browser by opening the following URL. This is the same link used by the app to view the stream from the Kinect. For this project we are using the raw image_topic. You can replace the IMAGE_TOPIC below by `?topic=/camera/rgb/image_raw`

```bash
http://localhost:7555/stream?topic=/IMAGE_TOPIC
```
iOS Development

All apps in iOS are build app using MVC design pattern. The following explanations is an overview of the code. Let’s check the view controllers first.

**SplashScreenViewController**

First screen of the app. Provides options for Login or Register.
SignUpViewController

Linked to Google Firebase Authentication. User can Register with their RIT or any other email. User information is then stored into Firebase database. User can login later using these credentials.
**DashboardViewController**

It provides 3 options: View Status, Scan QR code, View Map

ViewStatus connects to the STatusViewController

ScanQRCode connects to the QRCodescannerViewController

ViewMap connects to the FloorMapViewController

User can also logout.
**StatusViewController**

This controller connects to the vhost1.cs.rit.edu server in order to login and dispatch Corobots. User can login and then dispatch the Corobots. It also provides the option “Show Video” when the robot has been dispatched. This “Show Video” button connects to the KinectVideoViewController.
**KinectVideoViewController**

This controller shows the live video stream of the robot. It connects to the localhost server of the robot where the video is streaming and shows live video.
QRcodeScannerViewController

This controller accesses camera on your phone to scan QR Codes. Once User scans the QR code, it connects to the FloorMapViewController and user can see his/her location on the Golisano building 3rd floor map.
**FloorMapViewController**

This controller shows Golisano third floor map and also shows user’s and crobots(if any live) locations on the map. User location is shown in red. Robot location is shown in green. User can see the robot moving on the map. It also shows the coordinates and destination and status of robot when user taps on the robot’s location. It also shows the coordinates and user’s location name when user taps on his/her location.
Models

**ThirdFloorLocation** - Model class to store properties xCoordinate, yCoordinate, locationName for third floor of Golisano building.

**RobotStatus** - Model class to store status of a robot. It has properties like xCoordinate, yCoordinate, destination, status. We get the status of the robot as json object from the link https://vhost1.cs.rit.edu/cgi-bin/jsonOutput.php

Firebase

Firebase is a backend cloud service provider for mobile apps.

Development of Firebase is free but if more users sign up then you can pay for premium service. It provides up to 100 free live connections at any moment, which is enough in my opinion.

All of the Corobot development on Firebase is done using the username “cool.corobot@gmail.com”. Sign in using the credentials to follow instructions below.

*If it is a not a new development and is a similar project but just a different platform, skip the “Creating a new Project” page below.*
Creating a new Project

In case you need to create a new project you can click “Create new project”. This will display the following.

The iOS bundle ID can be found in Xcode (Click on project name>General)

Follow the rest of the steps. They are pretty self-explanatory.
**Existing Project**

One Firebase can handle many projects and is also cross-platform. Therefore, each individual app is a project which would contain the “iOS” as well as “android” version of the code. This project is done with use of username “cool.corobot@gmail.com” to login to Firebase. Check the Trello page for password.

On the Console of the firebase, there should be project named **Corobot**

![Firebase Console](image)

Welcome back to Firebase
Continue building your apps with Firebase using some of the resources below.

- [Documentation](https://console.firebase.google.com)
- [Sample code](https://console.firebase.google.com)
- [API reference](https://console.firebase.google.com)
- [Support](https://console.firebase.google.com)

Your projects using Firebase

- **Corobot**
  - corobot-c5d50.firebaseio.com
  - iOS 1 app

When you click on the project Corobot, you can see **overview** of the app. This project is iOS development but let’s say you are developing android app for the same project, can use this same project in firebase. Because the logic for backend remains the same. Below you can see the Corobot overview page shows one mobile app associated. That is this project. You can add your app here (upper right side) and get started.
If you wish to download the config file for the project. You can click the settings icon and then Project settings.

The settings page below contains the info-plist file critical for development with Firebase and iOS. It also has settings for Database and other features. This is needed for our project.
We are using real time database, Geo-fire, authentication and analytics for this project from the collection of services they offer.

The Auth Table below shows all the users registered with the app.

The database below currently contains the users and their locations.
Other components are not connected but could be used in the future for other developments.

Experiments

The first option “view status” is linked to the vhost1.cs.rit.edu web server. Once you log in to the app, you have to log in to the vhost1.cs.rit.edu from inside the app. This uses the RIT username/password of the user. Once logged in, user can operate the This network system is pretty robust. I ran the system for about two days and the robots still shows up on the map. The connection with the web server is pretty robust.

The second component to the app is the QR Code detection. The QR Code is able to scan all locations in the map and user gets localization on the map. This map also shows the current location of the robots.

The third component is the Map. The map is updated every 1 second for some update. The map contains the green sign for the Robots and shows the update for location (X,Y coordinates) in the bottom of the screen. The user location is in red and also shows the location of the user in the label on bottom of the screen.
Conclusion

The app is able to complete all the functionalities mentioned above. You are able to scan the Robot at any location on the CS floor and is able to detect it. The user is successfully able to communicate with the Web Server and the Corobots connected to it. The app is able to handle any change in location of user or the robot themselves.

Future Idea

The app could be used for various purposes. There are many Robots that need a native IOS app for the ROS system. This would work well for any indoor location with a map.

This app could be used to be the front end for the new users so they get familiar with functionalities and cools things it can do. You can create something as basic as controller for kids to move around the robot (there is already code for it on GitHub!) to something like attaching the ilauncher (has a nerf gun attached) to let the user control where he wants to shoot. Also maybe with the advent of all the new image object detection api’s, you could create some processing in the front end for the user. Or there could be some fun game with all of this.

The Vhost1 server could also be deployed to Firebase server.

References

http://wiki.ros.org/mjpeg_server
https://firebase.googleblog.com
https://vhost1.cs.rit.edu/status.php
https://vhost1.cs.rit.edu/location.php