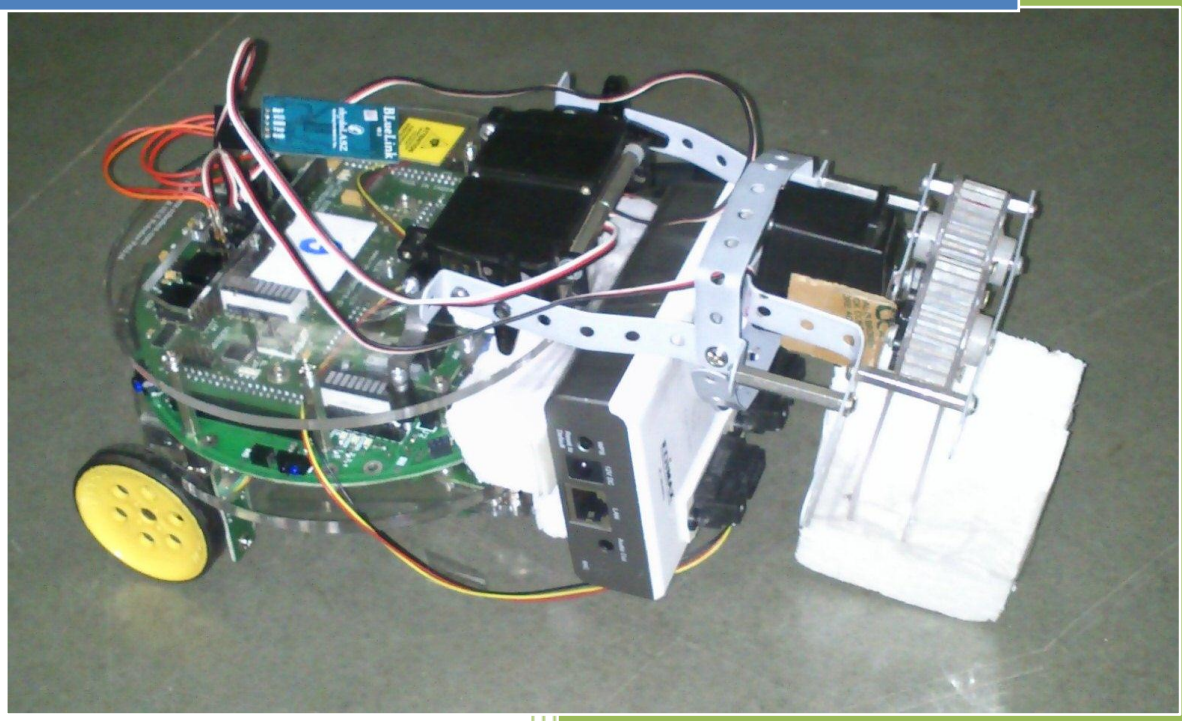
ERTS Lab,
IIT Bombay

Android Based Remote Tennis Ball Collector



Team 14

Hasan 09005065

Vinod 09005071

Bhanu 09005050

Avinash 09005056

Our Firebird functions as a tennis ball collector. It sends the video stream of environment in front of it via an IP Camera fixed in front of it. Android mobile downloads the video from the IP Address of camera and processes it for the presence of tennis ball. Then it sends appropriate signal to the bot via Bluetooth communication module.

1 Introduction

1.1 Overall Description

Our project aims at building a completely remote controlled firebird controller using android mobile. In that direction we implemented a Tennis Ball Collector which utilizes the processing power of Android mobile for image processing.

1.2 Definitions

S.No.	Name	Definition
1.	Mobile	Sony MT11i (Xperai Neo V)
2.	Android	Mobile operating system (sometimes used as 'mobile')
3.	Camera	IP Camera
4.	Bot	Firebird V
5.	PC	Computer, which acts as router in our case
6.	Wi-Fi/ Bluetooth	Communication Protocols
7.	Sensor	Sharp sensors connected to firebird
8.	Gripper Arm	Mechanical arm mounted on Frirebird

1.3 Requirement Specification

- Environment Capture using Camera
- Transmission of media from Camera via Wi-Fi
- Processing the media received
- Sending appropriate signals to the bot
- Analyzing the received signal
- Responding accordingly

1.4 References

- Our project is an extension of
  1. Tennis Ball Collector using ATmega 2560 Robot
  http://www.e-yantra.org/home/projects-wiki/item/131-tennis-ball-collector-using-atmega-2560-robot
  2. Controlling Firebird V using an Android based phone via Bluetooth Controlling FirebirdV ATmega2560
  http://www.e-yantra.org/home/projects-wiki/item/140-controlling-firebird-v-using-an-android-based-phone-via-bluetooth-controlling-firebirdv-atmega2560

- OpenCV + Android setup

  3. Using Android binary package with Eclipse
  http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package.html
  4. OpenCV in Android
  http://www.stanford.edu/~zxwang/android_opencv.html

1.5 Developer Responsibilities

Branch off the existing code or fork it and send us a merge request once every new feature is added.
Adhere strictly to the coding conventions followed.
Give due credit if you have used this code else where.

2 Implementation

2.1 Product Perspective

Now-a-days there have been a lot of dangerous situations where help might not be available and even if available humans cannot afford to risk their lives. In such cases we need an autonomous bot which can analyze its environment and react accordingly. Such requirement has motivated us to build this product, which can autonomously scan its environment and react accordingly.

2.2 Product Functions Overview

Our product captures its environment using an IP camera mounted on it and tranmits it via internet. This can be viewed from any part of the world. They can control its actions or let act by itself (however in our project we are limited by distance of bluetooth).

In autonomous mode, the mobile (video receiver) then scans for a tennis ball and if the ball is detected, adjusts itself to its centre and then sends a signal to move forward. One the bot reaches the ball, the senor predicts the distance between ball and bot and picks it if it is close enough.

In manual mode, user instructions will override default behavior. User can direct it to move forward, backward, turn left or turn right and pick the ball.

2.3 User Flowchart & System Design

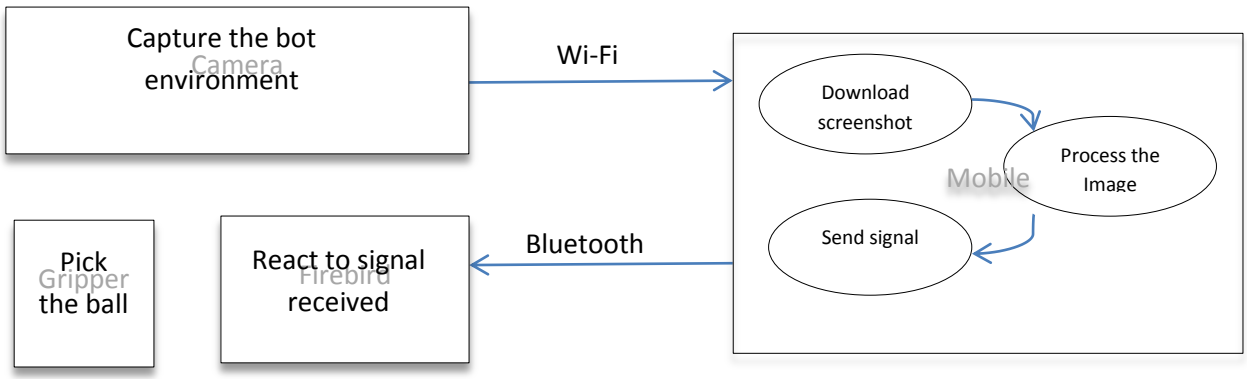


Fig 2.1.Flow Chart Diagram

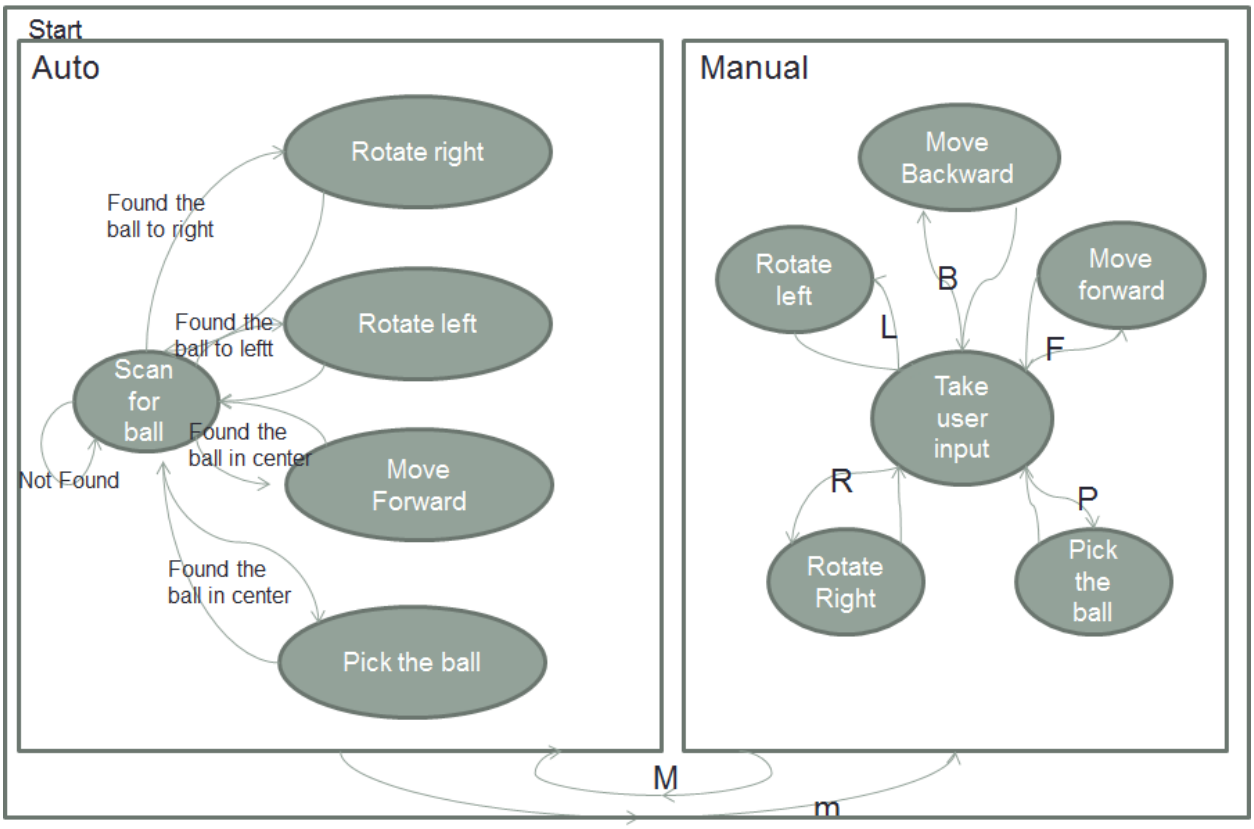


Fig 2.2. State Chart Diagram

2.4 User Characteristics

Here user is the person handling the mobile. He should have an idea of controls on UI (they are very much intuitive). While the application is running in Autonomous mode there is no user involvement as the mobile processes and sends the signals automatically.

2.5 General constraints and assumptions

- The only orange coloured objects present in the arena are TT balls. This is necessary for the algorithm used to work as it is based on colour detection.
- Only one ball will be present in the arena at any particular instant of time. Though the image processing algorithm can distinguish and identify one of the balls as object of interest, this can be tricky sometimes as android as no means of knowing which of the balls is close by.
- After every fresh setup, sensor values are to be re-calibrated as they seem to change with environment and from one-to-another.
- Hardware, Android:
    - armv7 processor is recommended, because of have a floating point preprocessor (Open CV).
    - Touch enabled – by design
    - Bluetooth and Wi-Fi capability – most smart phones have it
- Though initially planned during our SRS, we couldn't implement dropping of ball in basket after picking it up, due to lack of time.
- No packets are lost in transmission

2.6 Additional Hardware

- Bluetooth Module – 5V Serial UART (Rs. 2000)
- IP Camera – Edimax IC3030Wn
- Gripper Arm – mechanical parts
- Android mobile – Sony MT11i

3 Project Setup

- Setup the as shown in *Hardware manual*
- Install android*/bin/RemoteTennisBallCollector.apk* in your mobile
- Burn the *firebird/defult/Tennisball.hex* onto firebird
- Connect both IPCamera and mobile to same Wi-Fi network
- Start the Application
- Enjoy!

4 Implementation of Functional Requirements

4.1 Bluetooth Communciation

4.2 Wi-Fi Download

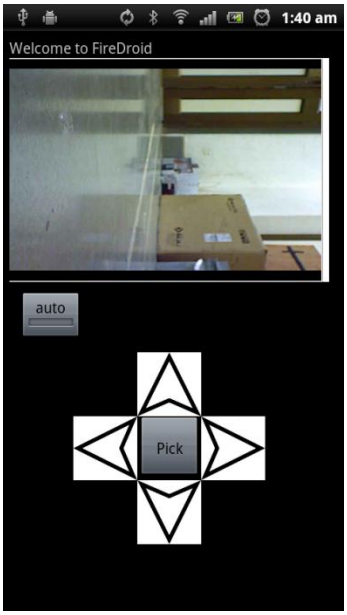
4.3 Image Processing

4.4 Android Application

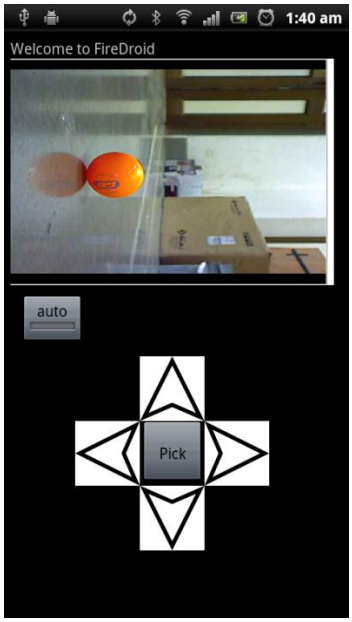
4.5 Firebird Bot Code

5 External Interface Requirements

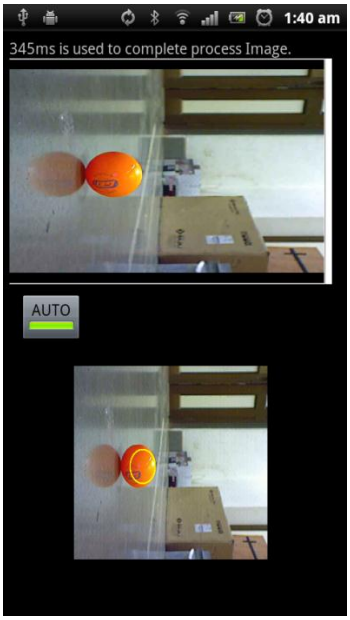
5.1 Android Application



Manual mode



Auto Mode



Auto Mode
(Ball detected)

5.2 Embedded Systems

FireBird V

5.3 Hardware

5.3.1 Bluetooth Module

5.3.2 IP Camera

5.3.3 Gripper Arm

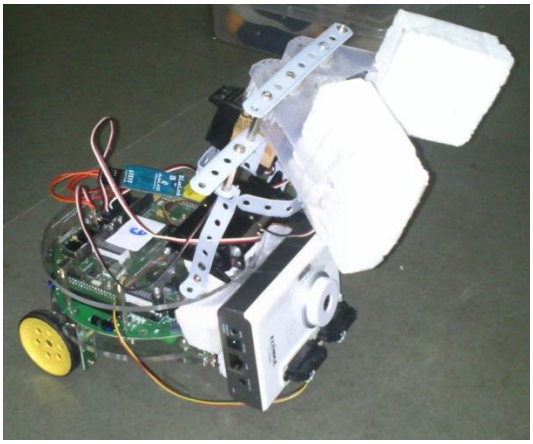


Fig. Gripper Arm

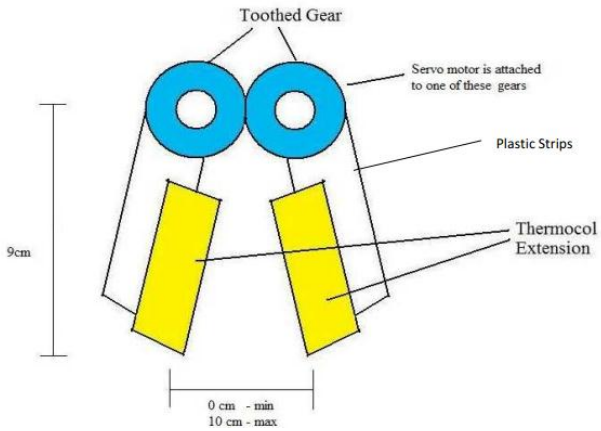


Fig. Gripper Arm

For instructions regarding re-construction of arm refer previous batches documents.

6 Development

Use the Hardware and Software Setup manuals for complete set of instructions.

6.1 Settings & Configuration

say for communication e.g. serial, Bluetooth, Wi-Fi

6.1.1 Environment

1. ***FB5***
2. ***Android***

6.1.2 Project (adjustable)

6.2 Utilities

6.2.1 Hardware

6.2.2 Software

- AVR Studio
- Open CV for Android Environment setup
  ** Refer:
  http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package.html
  ** Download NVPACK
  ** Download OpenCV
  ** Start the Eclipse and choose your workspace location.
  ** Configure your ADT plugin
  ** Import OpenCV and samples into workspace.
- Setup C++ OpenCV Environment setup
  **
  http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package_using_with_NDK.html#android-binary-package-with-ndk
  ** Download Android NDK
  ** Follow all instructions for Eclipse Setup

  By this time all your sample files should be working perfectly fine

- Download the example files
  ** http://www.stanford.edu/~zxwang/android_opencv.html
  ** Download opencv files and put them in jni folder of your android application root directory

  You are ready to go!!

  Troubleshooting

  Google it!

6.3 Code Description

6.3.1 Android

root folder of the project/

- ** jni/
- ** libs/
- ** res/
- ** src/
- ** AndroidManifest.xml
- ** default.properties
- ** ... other files ...

where

- the *src* folder contains Java code of the application,
- the *res* folder contains resources of the application (images, xml files describing UI layout , etc),
- the *libs* folder will contain native libraries after successful build,
- and the *jni* folder contains C/C++ application source code and NDK's build scripts Android.mk and Application.mk. These scripts control the C++ build process (they are written in Makefile language).

Also the root folder should contain the following files

- *AndroidManifest.xml* file presents essential information about application to the Android system (name of the Application, name of main application's package, components of the application, required permissions, etc)
  It can be created using Eclipse wizard or android tool from Android SDK
- *default.properties* is a text file containing information about target Android platform and other build details. This file is generated by Eclipse or can be created with android tool from Android SDK
- *src\com\eyantra\android\tennisball*
  *** BallCollectorActivity.java* - main class which does all control actions
  ** *Bluetooth.java* - Bluetooth communciation module
  ** *DeviceListActivity.java* - List the dveices for bluetooth
  ** *OpenCV.java* - OpenCV Java wrapper for C++ files in jni/ folder
  ** *ReloadImageView.java* - Class to reload images at regular interval (not used)
- res/
  ** layout/ - main layout
  ** drawable/ - images and sub-layouts used
  ** values/strings.xml - Strings used in code
- jni/
  ** cvjni.cpp - contains C++ Open CV functions to be used in other places
  ** cv/ and cvcore/ - opencv library files

6.3.2 Firebird

- Tennisball.c - firebird controller actions. Receives signal from Bluetooth and acts accordingly.

6.3.3 Description of Image Processing Algorithm

Convert the image to binary format

```
for(int i = 0;i < height;i++) {
        for(int j = 0;j < width;j++) {
                if( (data[i*step+j*channels+2] > 50+data[i*step+j*channels])
                                && data[i*step+j*channels+2] >= 225
                                && (data[i*step+j*channels+2] >
50+data[i*step+j*channels+1]) )
                {
                        wdata[i*wstep+j*wchannels] = 0;
                }
                else wdata[i*wstep+j*wchannels] = 255;
        }
}
```

Check the left and right margins of the ball

```
for(int i=0;i<width;i++){
        int count=0;
        for(int j=0;j<height;j++){
                if(wdata[j*wstep+i*wchannels]==0){
                        count++;
                }
        }
        if(count>10){
                if(x1 == -1) x1 = i;
                else x2 = i;
        }
}
```

Check the top and bottom margins of the ball

```
for(int i=0;i<height;i++){
        int count=0;
        for(int j=0;j<width;j++){
                if(wdata[j*wchannels+i*wstep]==0){
                        count++;
                }
        }
        if(count>10){
                if(y1 == -1) y1 = i;
                else y2 = i;
        }
}
```

7 Performance Characteristics

Performance depends on the

- speed of Wi-Fi connection
- whether the network is shared with WLAN used.
- Mobile Processing power
- Distance between bot and mobile (Bluetooth communication)

8 Design Constraints

(These constraints can be overcomed)

- Only one ball in bots view.
- Connects only to one hardcoded Bluetooth module
- Picks only one ball in auto mode and then stops.

9 Testing

9.1 Criteria

We have introduced a delay of 1s between any two communications. This is done is order to give the other end sufficient time to realize the signal sent.

Since the Android mobile has considerably much lesser resources when compared to PC, delays might be larger.

9.2 Results

We have practically observed that the android mobile takes about 500ms to process a single image and transmit the appropriate signal.

We have tested the bot in various environments and background views. It works fine unless there are orange object in background and also slight variations are found in sensor values with environment.

10 Troubleshooting

10.1 Delays

We have introduced a delay of 1 second between any two consecutive communications. This is to make sure that the other end gets sufficient time to receive, analyze and respond to the signal sent.

10.2 Bluetooth not connected

Try restarting both the application and firebird. Wait for about 30 sec between two consecutive starts.

10.3 Image Download very slow

Share your local LAN with the WLAN setup in your PC.
We don't know the reason but this works.

10.4 Bot tries to pick up before reaching the ball

Change the sensor thresholds where pick function is called I auto mode. The sensor values keeps changing with environment.

11 Individual Roles and contribution

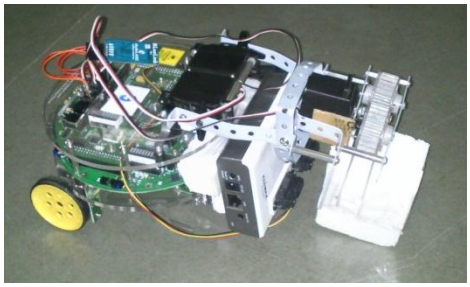
S.No.	Name	Work done
1.	Hasan Kumar Reddy	Android application
2.	Bhanu Prakash	Firebird code
3.	Vinod Reddy	Image Processing OpenCV code
4.	Avinash T	Hardware

12 Roadmap

S.No.	Milestone	Alloted to	Completed On
1.	Project Presentation	Vinod, Hasan	7th March
2.	Identifying modules and allotment	All	16th March
3.	Setting up the Wi-Fi Camera	Hasan	29th March
4.	Received our order for Bluetooth module		3rd April
5.	Android Application	Hasan	7th April
6.	Image Processing Code	Vinod	9th April
7.	Firebird Application Logic	Bhanu	8th April
8.	Fixing Arm on Bot	Avinash	9th April
9.	Bluetooth Transmission	Avinash, Hasan	10th April
10.	Bluetooth Receiving	Bhanu, Vinod	10th April
11.	Project Integration	All	11th April

13 Challenges and Innovation

- Android Mobile (MT11i) doesn't recognize the usual ad-hoc Wi-Fi network setup by PC. But for IP Camera to work even Mobile is to be connected to the internal network (proxy issue).
    - Rooted the mobile and tried various applications which claimed to work, but unfortunately none of them worked.
    - Finally setting a WLAN network bridge (which acted as an Interface mode Wi-Fi router) in windows worked.
      Source: http://www.youtube.com/watch?v=klHzMyYFGeQ
- To display bot's environment in the mobile, we initially tried loading an image at regular intervals (less data transfer). In its implementation, we had to use the native java timers, which made the app non-responsive a few time. (Though a better approach would have been using a handler for a runnable, which calls itself after certain time after its execution.)
  So, we switched to webview, which displayed the default monitoring page of the IP Camera. The problem here is that it always displays four different camera streams, three out of which are defunct for our purpose.
  Soln: We ran a javascript on top of webview to hide all unnecessary fields.
- Open CV :
    - Installing Open CV in Windows is a challenge in itself. We had to set it up for android environment in windows. We had followed various tutorials , but all in vain. Finally got a bundle of open CV C++ files, which we had put them inside our jni folder instead of linking the Open CV library during the build time. This made the whole open CV library a part our project. Though it take a while to build for the first time, it's worth it. Your android application will still be of same size as before as eclipse is clever in linking only those object files which are refered.
    - The usual opencv cvHoughCircles, which is used to detect circles doesn't perform well in non-uniform background and is relatively slow. We came up with our own algorithm (described in 6.3.3) which works fine in most cases and is fast.
- Hardware:
    - Mounting the camera
        - If placed in front, will obstruct the sharp sensor and will be in the path of arm movement
        - If placed on top, will be obstructed by the gripper arm
    - Using a single sharp sensor led to missing the ball in a lot of cases, so we have decided to use two closely placed sharp sensors
    - Solution :
        - Increased the length of gripper arm
        - Placed the sensor on the camera
        - Placed the camera in front of the bot



13.1 Github Repo

We have maintained our github repo in an organized fashion and updated it regularly. We maintain two branches one for development and other for master and merged development to master once after every milestone is reached.

13.2 Bugs & Fixes

- If the Bluetooth module doesn't get connected, restart both bot and application and then try to connect again.
- Both the servo motors were out of sync initially, fixed via adjusting the rotation angles in software.
- One the ball is picked in auto mode, cannot collect any more balls. – by design
- Build your project only for Android version > =2.3 as apparently there is but in Android 2.2 regarding BitmapFactory which makes the image download and processing very slow (around 20 times)

14 Bug Report

- Application seemed to crash when many other applications are running simultaneously in the background.
  Fix: Close all applications before opening this application
- Bluetooth connectivity:
  A few times Bluetooth connection doesn't get established automatically.
  Fix: Restart both the bot and application after waiting for about 30 sec.
- If the first signal sent by the application is to move either left or right, i.e, not forward or backward, the application on bots end gets stuck and will not receive any more signals.

15 Reusability

We have used parts of Bluetooth communication from previous batch's application code. On the similar lines we expect someone else to use our code in forth coming years.

Hence we have our code in a nice modular fashion and documented wherever necessary.

- Image Downloading
- Image processing
- Application code
- Communication
  Each in a different class.

In-code standards:
have adhered to java commenting standards (javadoc), eclipse friends

We have also included a brief description on how to use Open CV for android image processing projects. This could save a lot of time for forthcoming teams.

16 Future Work

- Replace the Bluetooth communication with Wi-Fi shield. This make the application remotely working in real sense.
    - Resources:
        - Application using Wi-Fi sheild http://www.instructables.com/id/How-to-build-an-Arduino-WiFi-4x4-with-Android-Cont/
        - Procurement site http://asynclabs.com/store?page=shop.product_details&product_id=26&vmcchk=1
- Use and existing or write one openCV wrapper for android (Java)
    - http://code.google.com/p/javacv/
- Improve the UI control
  Gives better control over bot and instructions are executed smoothly.
    - e.g., http://www.instructables.com/id/JabberBot-The-Arduino-robot-with-an-ATMega-brain-/
- Implement reliability
  Right now we assume that all the signals sent have reached the other end. But this might not be the case in a real world example. In that case we need to acknowledge in some manner that the packet has not reached the other end.

17 Conclusion

During the course of the project we have learnt a lot including

- Open CV Image processing
- Interfacing Open CV with android
- Bluetooth protocols
- Configuring Bluetooth module
- Various hardware usage techniques

All which could help us even a long way after the completion of the course and are very satisfied with the project.