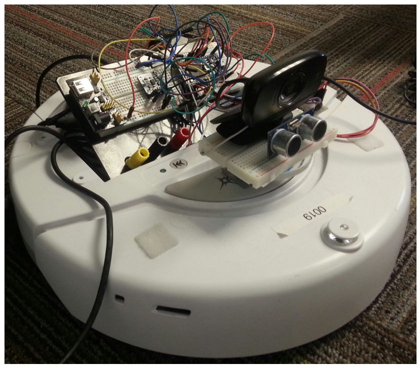
CodeIT | CSE291E: Introduction to Robotics Systems | December 12, 2015



iRover: Exploration Robot

ABHIJIT TRIPATHY| ABHINAV GARG| PULKIT BHATNAGAR

Project Introduction

This project is developed as a part of **CSE 291E**: ***Introduction to Robotics Systems*** course. The course included a series of mini-projects during the quarter starting from the **Mbed microcontroller** and integration with sensors and display units. The next phase was setup of the **BeagleBone** black which would ultimately provide wireless connectivity to our system. Finally, iRobot create was introduced serving as the base of our project. All these accumulated toward the final iRover exploration robots and we have used most of the components provided within the course.

Project Objective

This project is inspired by the Mars' Rover success which showcases the importance of robotics in coming times. NASA has been able to extract invaluable information from the red planet without risking life of anyone, all through robotics. With this project we try to emulate a limited set of those functionalities to get a view of the challenges a realtime application would encounter. Our goal is build an exploration robot that go hazardous places while giving most human like facilities. We implement a fully manual driven rover controlled wirelessly through base station (laptop). We mount a camera and ultrasonic sensor over the head of a rotating motor that gives our robot the ability to see and also stream live events back to the control center. The Magnetometer/accelerometer serve for location detection in events the robot gets lost. Finally, to give it an autonomous touch, we implement a 'search and approach' feature through object recognition. This enables the rover to scan and detect a predetermined object and approach towards it avoiding any obstacles on its way.

Implementation

The following section describes the components, design architecture and various protocols used for implementing the iRover.

iRobot: The main robot/chassis on which the system rests.

Laptop/Mission Control: The host controller will navigate/steer the rover wireless though either commands or real time driving.

BeagleBone: For Wifi communication between the controller(Laptop) and iRobot (Can also think of control over the Internet enabling round-the-world access)

Ultra Sensor: As Collision warning detector. Sits on top of the system and scans like a radar for obstacles detection.

Camera : Realtime video of the iRover's view sent over wifi.

Magnetometer: Reports the Accelerometer/Magnetometer(Heading) and sends to mission control (laptop).

Battery Cape/Portable Supply for wireless operation.

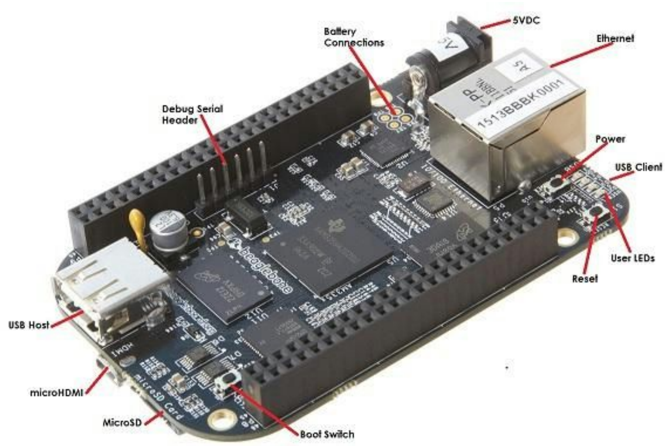
1. Components

a. iRobot Create:

iRobot is an open platform for robotics an provides an easy way to customize the robot. It is used as the main chassis on which the iRover system rests.

b. BeagleBone Black with wifi

This was our server. We used BBB to read sensor data from serial to which our MBED was writing. Also the BBB was used to stream webcam video feed to our client where we did all our processing.

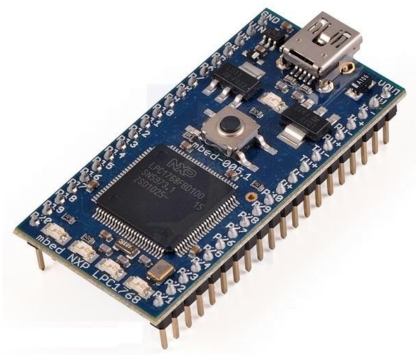


c. Logitech Camera

This was our picture/video capturing device.

d. Mbed

The MBED was used to read all sensors as well as to drive the motor. It would write data to serial , which the BBB would read .



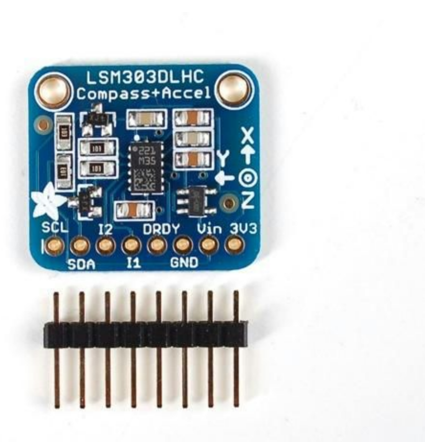
e. Stepper Motor

The motor was used to rotate our panel on which the camera and ultrasonic sensors are mounted.

f. Ultrasonic Sensor

This is critical for our obstacle detection unit. As in Assignment 2 , we use it to search for obstacles by receiving the echo.

g. Magnetometer/Accelerometer



This gives us the heading of the location to which we have reached.

h. Battery Pack:

We used a battery pack to provide power to the Beaglebone black, USB webcam, WIFI dongle & MBED. The Wifi dongle and webcam draw a good amount of power. Without external battery pack ; video wifi and MBED couldn't be operated simultaneously,

i. Gstreamer :

Gstreamer is used for streaming video captured via the Logitech webcam. BBB is used as a wifi router. We used udp protocol to stream the video over wifi. Gstreamer 0.10 was used as well as some gst-bad plugins were used to stream the content. We captured raw yuv data in 4-2-2 format from the webcam. The resolution was 320x240. We used mjpeg encoder at the BBB side. At the client side, we used gstreamer to dump the decoded data into a fifo pipe. OpenCV was then used to read data from the video. We were able to get 10 fps by this method.

Video Capturing & streaming gst pipeline at BBB:

```
gst-launch v4l2src ! image/jpeg,framerate=30/1,width=320,height=240 !
udpsink host=192.168.43.198 port=2000
```

Video receiving gst pipeline at the client side:

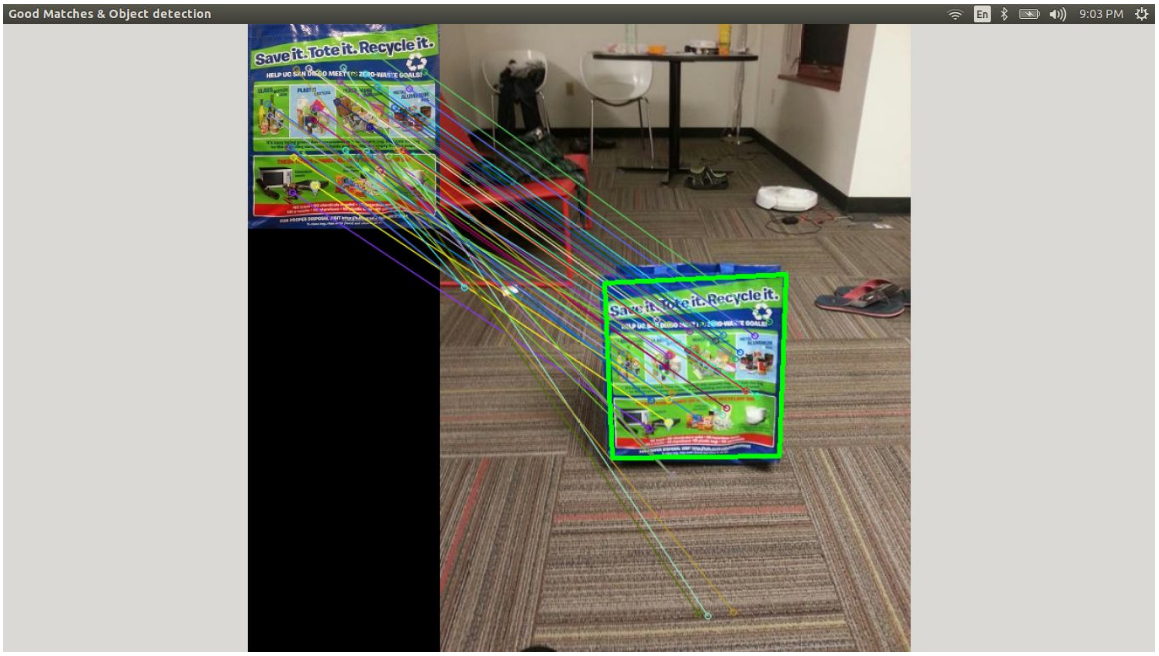
```
gst-launch -v udpsrc port=9991 !  jpegdec ! ffmpegcolorspace !
'video/x-raw-yuv,format=(fourcc)I420' ! ffenc_mpeg4 ! filesink
location=myfifo
```

j. OpenCV :

We have used OpenCV for processing the video data streamed by the webcam.

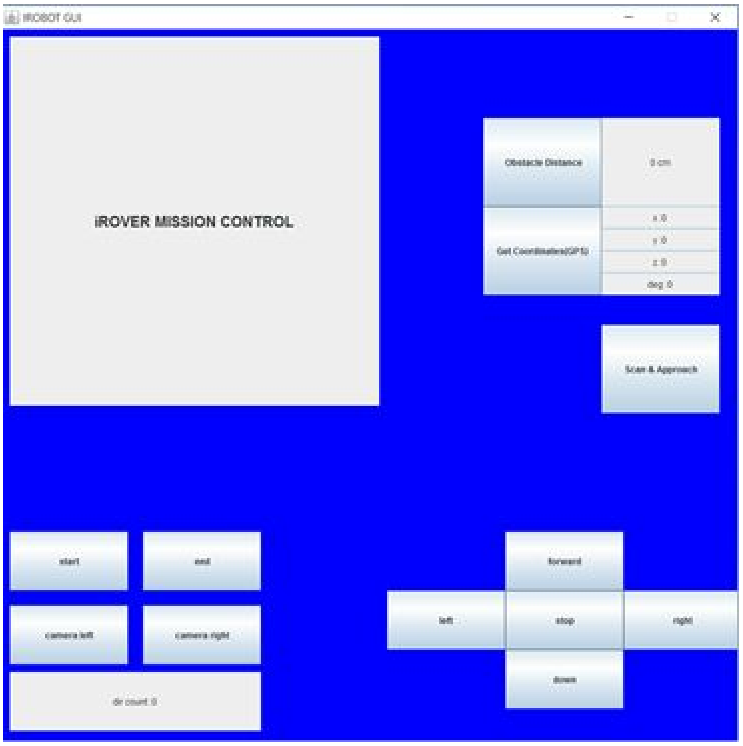
OpenCV used VideoCapture API to read individual frames. Then we used SURF instead of SIFT , to find a object match. SIFT wasn't real time , so SURF is a good approximation to it.When an object is detected a green bounding box is drawn around it. We use the position of object in the scene frame to determine the heading , towards with the iRover is expected to move so as to reach the object. However we noted that the detection match is not completely tolerant to lighting and colour.

The results in case of a match are shown in the pic below:



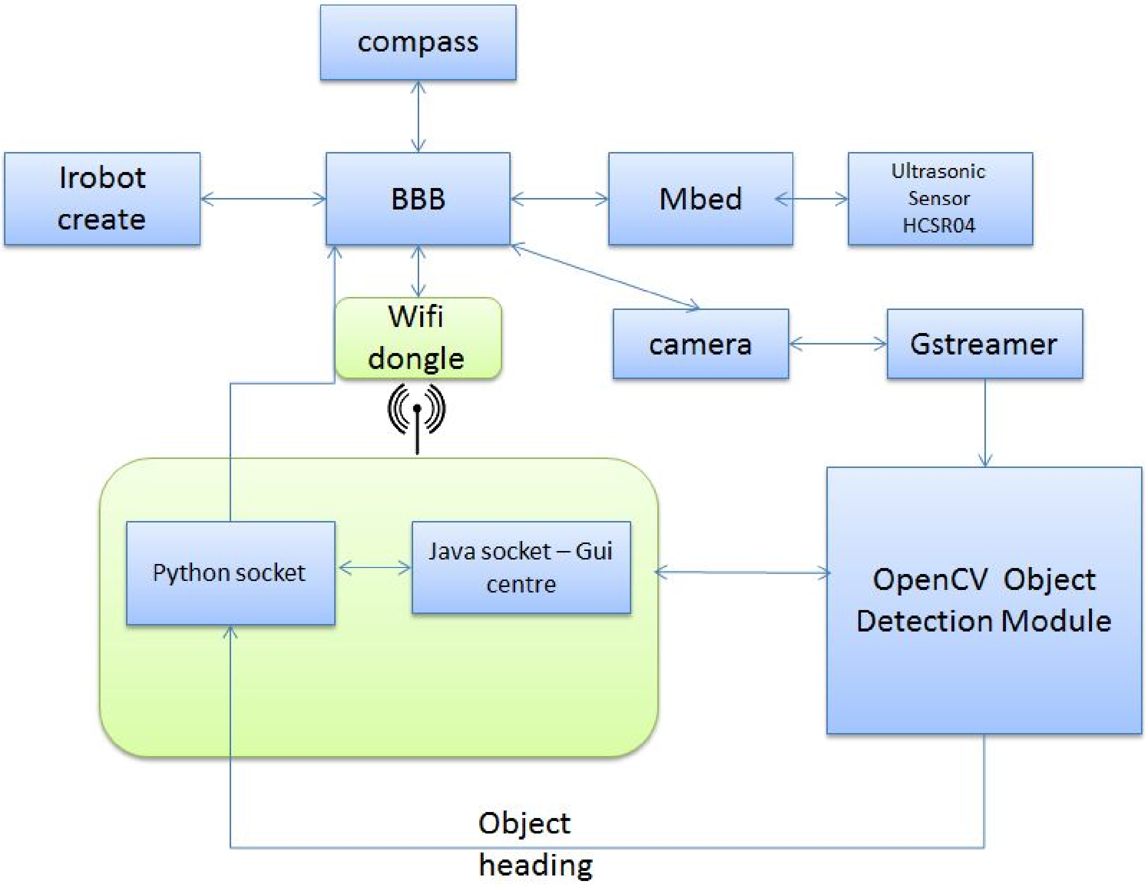
k. JAVA GUI :

The GUI could be used to control the iRover through keyboard, instead of dumping data at seria through code.It looks like :



2. System Architecture.

The system architecture is shown as follows:



3. Protocols and Communication

a. UART

UART was used for serial communication . The sensor data acquired by MBED is sent to BBB via UART @9600 Baud Rate. The serial communication between BBB and Roomba is also done via UART @57600 Baud Rate

b. OpenCV/gstreamer/udp .

Video is streamed using UDP protocol through gstreamer.

c. Python & Java Client-Server model

We have two clients. Our basic socket for data transfer between MBED, BBB and our laptops is through a python client socket code. However , in case of our GUI, we have made a JAVA based client which uses JAVA GUI for manual control.

Challenges

Some of the challenges that we faced while implementation were:

1. Loose connections of components mounted on irobot.

   Debugging these was really painful , as we would always wonder as to why this sensor is not working and the list just goes on.

   Workaround: Double check all connections before carrying out your experiments.

2. FPS of video processing acquired through Gstreamer.

   Workaround: Since video frames are highly correlated, processing at small fps solved the problem. It sometimes proved to be a bottleneck , when the scene changed fast. i.e Moved from a scene having object to one where there was none.

   A stronger wifi for streaming and a GPU to aid processing would also help.

3. Robustness of Object Detection algorithm

   False detection might lead our robot to detect the incorrect object or move in a totally wrong direction.

   Workaround: use images that have a good number of identifiable features. Detect object in exactly the same environment where the training was carried out. Also , take video in good lighting conditions.

   Also if GPU is available use SIFT features instead of SURF for object detection.

4. Slow Wifi leading to slow response

   This is a big problem, as delayed sensor/video data transmission may cause our system to run haywire.

   Workaround: Use Connectify , to share high speed SGA for all devices to connect.

5. Reliability of sensor data.

Sensor data aren’t reliable always. Faulty magnetometer heading may lead to incorrect position calculation. Incorrect ultrasonic data may lead to collision with obstacles. So its a critical problem.

Workaround: Use filtering techniques , to reduce the impact of outliers.

6. Size of irobot.

   The size of irobot , may be a bottleneck while scan and approach as obstacles may be very near to the robot , and might not get detected properly leading to collision.

   Workaround : Always rotate the camera at a good frequency , so as to keep track of all obstacles around the bot.

7. Camera powering off whenever battery pack went below 2 levels.

   This is a big problem as sometimes we would have to wait till battery powers up until we can check our object detection. Also spent a huge time in debugging this issue.

Workaround : Keeping spare powerbanks .

Project Review

Our initial project proposal was limited to the manual maneuver of the iRover through the mission control over Wi-Fi with live video streaming. But as we worked along and feedback from the professor and TA we were able to extent our work to include an autonomous object search and approach feature which gave the project a completeness and widened the scope of the application.

Some of the use cases where our iRover is a perfect fit are:-

1. A warehouse bot : In a warehouse , a fully automated bot could be programmed to search a particular object , say a Laptop and take it near the packaging unit.
2. In a library , a particular book could be searched and then taken out .
3. For spying a dangerous unknown terrain and knowing more about it.
4. For Mars Rovers.kind of application.

The Team: CodeIT

We believe that the team composition was ideal. All the group members had an area of expertize during each lab in which they contributed appropriately. One member was into coding and creating algorithm for the given tasks, another more involved in the hardware aspects and debugging and finally one played a role of a system administrator with contribution in linux/Ubuntu and computer vision (openCV). At the end of each project the ideas of the videos and presentation all worked together well and we can confidently say that it was 10weeks of great and effective team work.

References:

http://homesupport.irobot.com/app/answers/detail/a_id/843/~/irobot-owners-manuals-and-quick-start-guides

http://docs.opencv.org/2.4/modules/features2d/doc/features2d.html

https://bitbucket.org/CodeIT291/codeit/src