ECE1778: Creative Applications for Mobile Devices

Mover-bot

Android-based Mobile Robotics Development Platform

Paul Bovbel and Fitsum Andargie
Word count: **1639**
4/8/2012

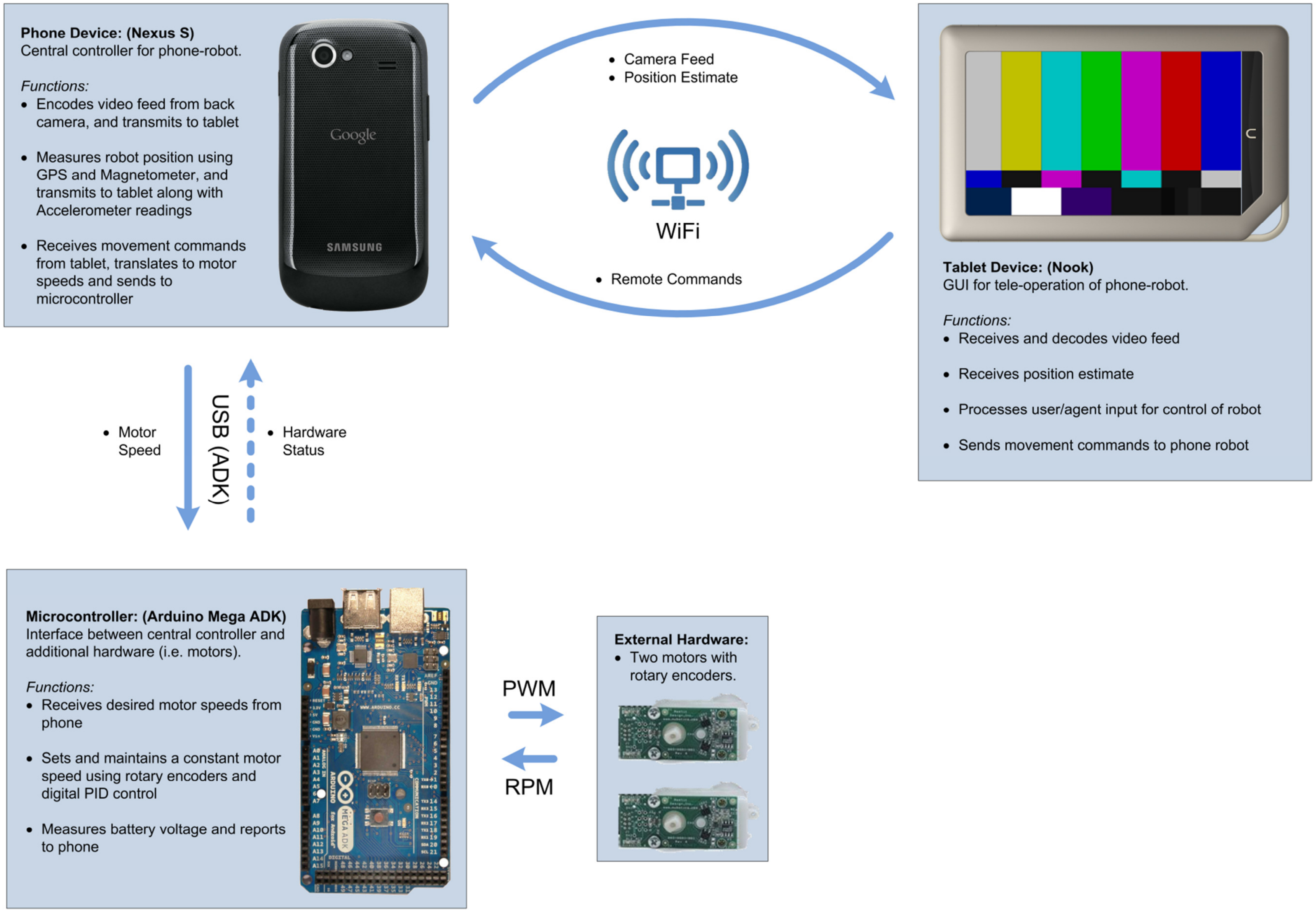
Fitsum Andargie
Paul Bovbel

Introduction:

Traditionally, research in the field of mobile robotics field requires the development or purchase of a robot that integrates multiple sensors and actuators, and subsequently transmits this data to an external agent for processing and control purposes. Purchase of a platform is very cost-prohibitive, and development of a new platform is time-consuming.

Interestingly, the now-ubiquitous smartphone includes the majority of electronic components that are mentioned above. A typical current-generation smartphone contains most of the sensors (camera, positioning, etc.) required for the function of a mobile robot, and has communications systems (IP over Wi-Fi, LTE) that can handle the bandwidth and range required for mobile robotics projects.

For the Mover-bot project, our aim was to develop an application that would use an Android smartphone as the central sensing and communication unit of a mobile robotics platform. Along with this central app, we also wanted to construct a robot chassis to house the phone, as well as a remote controller application to demonstrate the Mover-bot's capabilities. A high-level overview of our proposed system is shown in Figure 1. We envision that the Mover-bot would be a logical choice for a large user-base seeking to experiment with mobile robots, including hobbyists, educators, and researchers on a budget.



Phone Device: (Nexus S)
Central controller for phone-robot.

Functions:
- Encodes video feed from back camera, and transmits to tablet
- Measures robot position using GPS and Magnetometer, and transmits to tablet along with Accelerometer readings
- Receives movement commands from tablet, translates to motor speeds and sends to microcontroller

- Camera Feed
- Position Estimate

- Remote Commands

Tablet Device: (Nook)
GUI for tele-operation of phone-robot.

Functions:
- Receives and decodes video feed
- Receives position estimate
- Processes user/agent input for control of robot
- Sends movement commands to phone robot

- Motor Speed

- Hardware Status

Microcontroller: (Arduino Mega ADK)
Interface between central controller and additional hardware (i.e. motors).

Functions:
- Receives desired motor speeds from phone
- Sets and maintains a constant motor speed using rotary encoders and digital PID control
- Measures battery voltage and reports to phone

External Hardware:
- Two motors with rotary encoders.

Figure 1: System Overview

Software Design:

A lower-level overview of our project's software design is shown in Figure 2, and described in the subsequent sections. Since three separate hardware devices are used in our project, a large part of the work was ensuring smooth and error-free communication flow between each component.

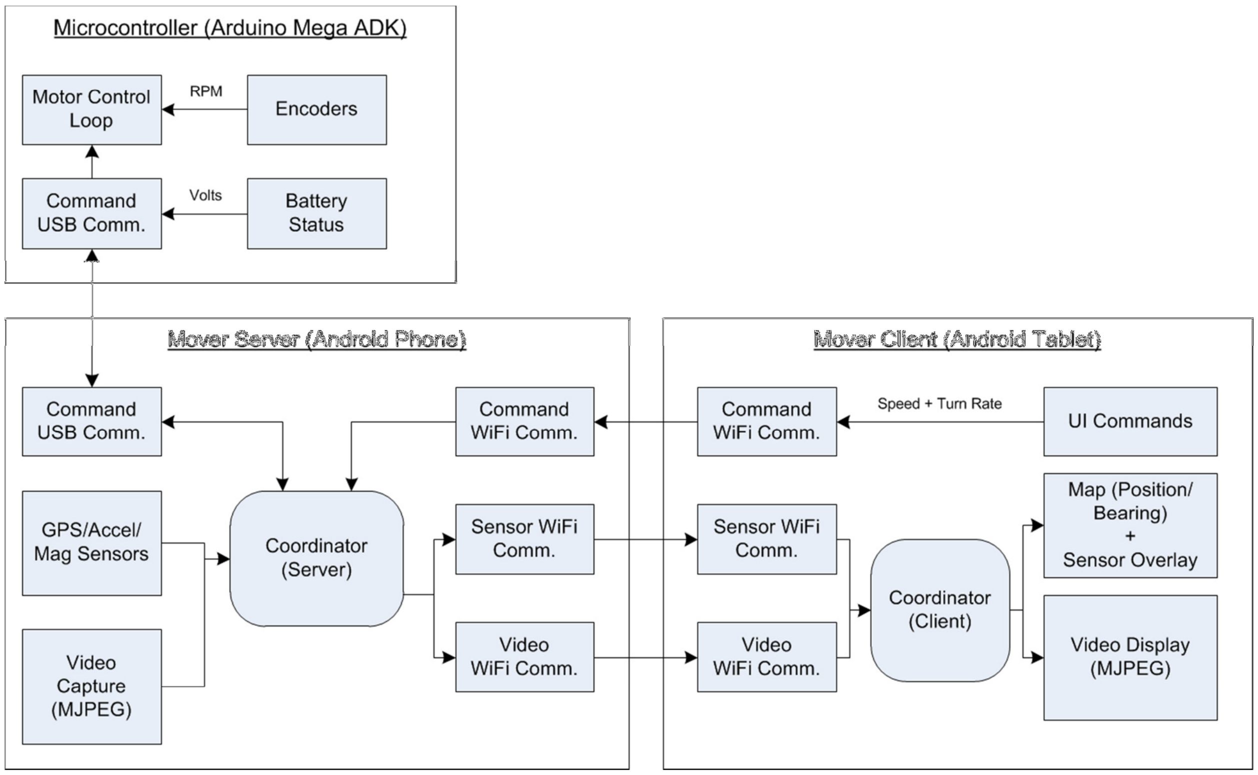


Figure 1: Mover Software Block Diagram

Microcontroller (Arduino Mega ADK)

A microcontroller is used in this project to control the motors and regulate power. Components of the software loaded on the microcontroller are described below.

- *Motor Control Loop*: translates speed and turn rate into a desired speed for each motor. Maintains the speed using PI controllers with feedback from the encoders.
- *Encoders*: these are attached to each motor and measure motor speed between each control loop iteration.
- *Battery Status*: samples the battery voltage at regular intervals, and applies a low-pass filter to the data.

USB Communication:

The following describes the USB communication between the microcontroller and the phone's Mover Server. We use Google's Accessory Development Kit protocol for bi-directional communication between the devices.

- *Battery:* Transmits the filtered battery voltage from the microcontroller to the server.
- *Commands:* Transmits commands from server to microcontroller, including on/off, speed, and turning rate.

Mover Server (Android Phone):

This component is triggered by the Android system upon detecting a USB connection from the microcontroller. Upon launch, it configures an IP server on port 9090, awaiting connection from a Mover client.

- *Coordinator:* Manages USB and WiFi connections, along with the phone's internal state (flashlight, sensors, lifecycle)
- *GPS/Accel/Mag Sensors*: samples the phone's internal location sensors for location (GPS), acceleration (Accelerometer), and bearing (Magnetometer) data.
- *Video Capture*: grabs frames from the camera and encodes the as a JPEG byte stream.

Wi-Fi Communication:

Communication between the Mover Server and Mover Controller is achieved using an IP connection over WiFi. The three layers of communication are described below.

- *Command:* Transmits commands from client to server, including desired speed and turning rate, motors on/off, flashlight on/off.
- *Sensor:* Transmits sensor data from the server to the client, including location, acceleration, bearing, and battery level.
- *Video:* Transmits the camera's JPEG byte-stream from the server to the client.

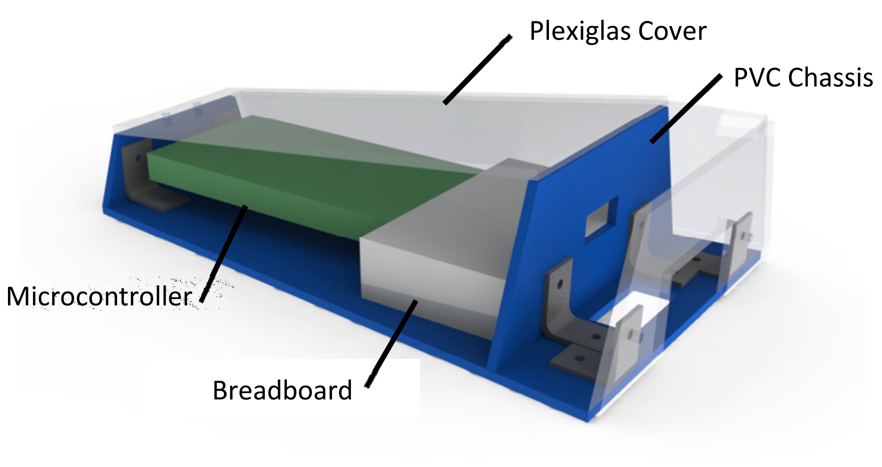
Mover Controller (Android Tablet):

This component will connect to a Mover server at a specified IP address, and then performs the bulk of user input/feedback in our application.

- *UI Commands*: translates user touch input into commands, including desired speed and turning rate, motors on/off, flashlight on/off.
- *Map + Sensor Overlay*: displays location, acceleration, bearing, and battery level data to the user using the Google Maps API and additional overlays.
- *Video Display*: decodes the JPEG byte stream (using error correction to prevent garbage frames), and displays in a container.

Chassis Design:

The project also involved the design and implementation of a Mobile Robot as indicated previously. Figure 3 shows a 3D model of the designed chassis and also its implementation. The chassis has been designed to house the phone device, microcontroller, and other electronic components.



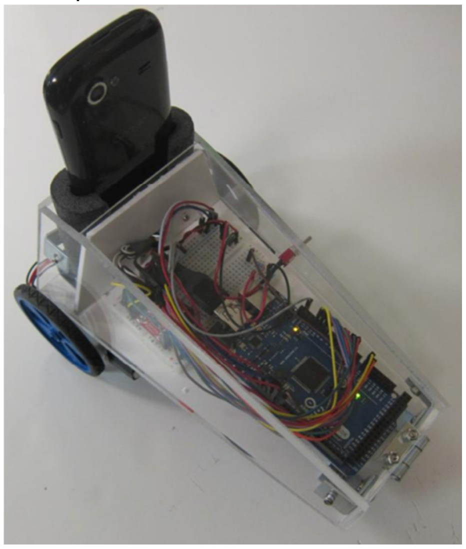


Figure 3: Mover Chassis Design and Implementation

The rear compartment is designed to contain the Nexus-S phone model. The motors, encoders, battery, phone, front caster, and additional wiring and hardware have all been omitted from the model. The materials used in building the chassis are listed below.

Mechanical BOM:

- Plexiglas material
- Sintra PVC material
- Solarbotics GMPW plastic wheels (2)
- Polulu ball caster
- Misc. hardware (i.e. nuts, bolts, hinges, brackets)

Electrical BOM:

- Arduino Mega ADK microcontroller
- Solderless Breadboard
- Dual Motor Driver (TB6612FNG)
- Zippy 3S 11.1V LiPo battery
- Solarbotics GM9 Gear motors (2)
- Solarbotics Wheel Watcher encoder (2)
- Misc. hardware (i.e. switches, wires, connectors, resistors)

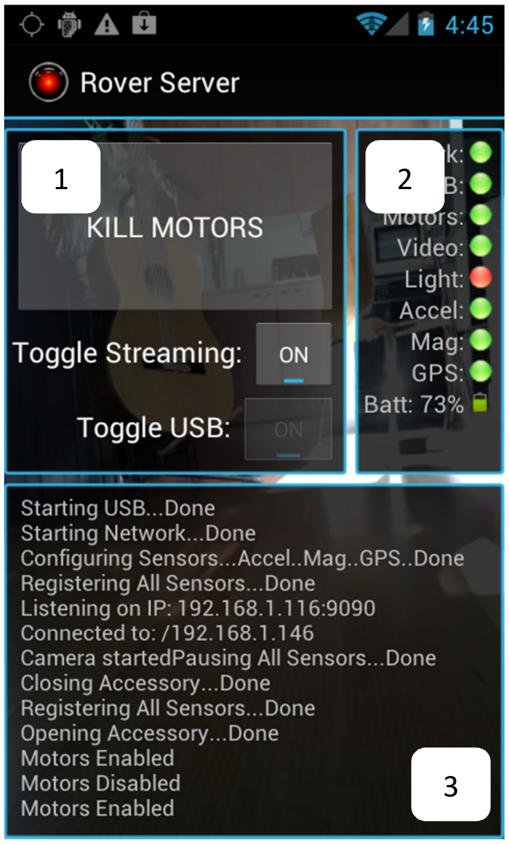
User Interface Design:

Mover Server:

The server application runs on the phone embedded in the Mover-bot chassis, and is therefore not specifically designed for user interaction. It is used primarily to display any relevant status information to a user.

Description of each part of the user interface numbered 1 to 3 is as follows

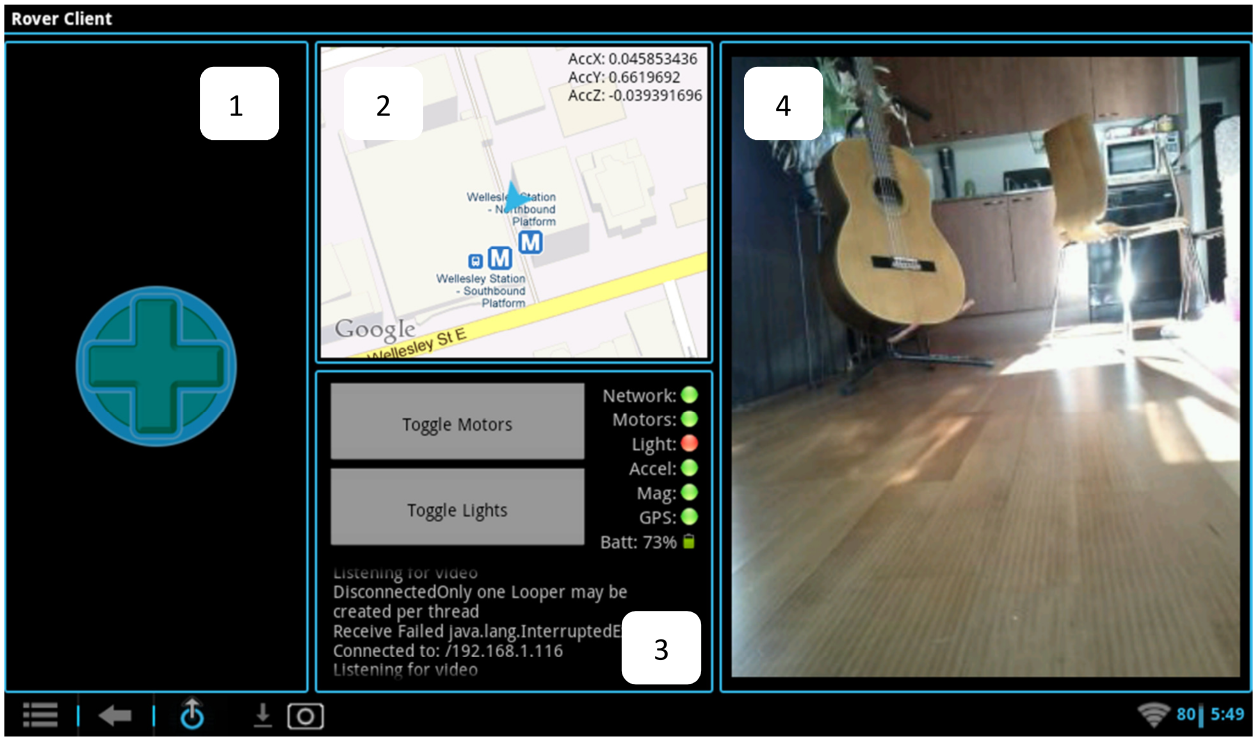
1. This part has three buttons.
   - The KILL MOTORS button is used to stop the motors in case of emergency.
   - The Toggle Streaming button is used to set whether the Mover Server is currently streaming video to a client (to correct out-of-sync issues).
   - The Toggle USB button is used to enable and disable USB communication.
2. This part of the UI visually shows the status of components of the system. When the signs are *green*, it means the component is activated. If it is *red*, it means it is not activated.
3. This part is the console where status reports are displayed in a text form.



```
Starting USB...Done
Starting Network...Done
Configuring Sensors...Accel..Mag..GPS..Done
Registering All Sensors...Done
Listening on IP: 192.168.1.116:9090
Connected to: /192.168.1.146
Camera startedPausing All Sensors...Done
Closing Accessory...Done
Registering All Sensors...Done
Opening Accessory...Done
Motors Enabled
Motors Disabled
Motors Enabled
```

Figure 4: Mover Server UI

Mover Client:


Rover Client
1
2
3
4
Wellesley Station - Northbound Platform
Wellesley Station - Southbound Platform
Google
Wellesley St E
Toggle Motors
Toggle Lights
80
5:49

```
AccX: 0.045853436
AccY: 0.6619692
AccZ: -0.039391696
```

```
Network: ●
Motors: ●
Light: ●
Accel: ●
Mag: ●
GPS: ●
Batt: 73%
```

```
Listening for video
DisconnectedOnly one Looper may be
created per thread
Receive Failed java.lang.InterruptedE
Connected to: /192.168.1.116
Listening for video
```

Figure 5: Mover Client UI

The above figure shows the user interface for the Mover Client, which is the demo controller for our platform and performs the majority of user input and feedback. Each part of this user interface is described below.

1. This part is touch panel where forward/backward and sideways movement commands are given. When a user moves their finger vertically up increase speed command is sent and if the reverse happens the speed is decreased until the rover reverses direction to move backwards. On the other hand, if the user moves their finger laterally, the robot changes direction in tandem with the finger movement.
2. This is where the sensor data displayed. The GPS and magnetometer readings are incorporated in the Google maps to show the location and bearing of the robot. The accelerometer reading is also overlaid on Google maps.
3. There are different components here. The two buttons are used to turn the motors ON/OFF and turn the LED light on the phone ON/OFF, respectively. There is also a visual depiction of the status of different components in the robot. In addition the console window here is used to print status reports in text.
4. Here, the video sent from the robot is displayed. The user can use this to control the movement of the robot.

Conclusion:

Our end product works to the specifications we put in our plan. We set out to build something affordable to be used as mobile robot development platform and we believe we have achieved that.

The functionalities of our product is listed below

1. The Mover-bot application allows real-time control of a mobile robot chassis, using the Android phone as the main communications hub.
2. The application allows the user to receive real-time feedback in the form of positioning sensor, robot status, and video feed.
3. The design of our application allows for easy configurability for use with additional sensors or a different chassis entirely, making the project a functional development platform for mobile robotics.

Contributions:

Fitsum Andargie – Prototype Wifi Communication, JPEG over HTTP, Report Compilation

Paul Bovbel – Chassis and Hardware, User Interface, USB Communication, Final Wifi Communication, JPEG Byte-streaming, System Integration, Report Editing

Learning Experience:

We have learned a lot in programming the android device using java. We have been able to explore USB communication with other hardware, access the sensors on board and utilize that data, set up communication using the Wi-Fi interface and work with the camera to capture video and transmit it. Developing a functional and aesthetic user interface was one of the learning experiences as well. Time management and getting something working with in the deadline was also another insight acquired in the course of the project.

From a technical point of view, something that is never covered in the Commonsware book but would be very useful to know early is the ability to extend the Application class, in order to maintain global information across separate Activities and their instances.

Future Work

There is always a chance to improve. We believe our project would benefit and grow if the following changes are integrated in to it.

- The code written would benefit from restructuring so that it can be used as a library for further use. As such making it more modular and clearly defining the interfaces of each module with the appropriate comments will make it more usable.

- Add additional features like

    - Audio capture and transmission with video for more completeness

    - Integrating other sensor readings like the barometer, temperature …
    - Procedures of automatic recovery in case communication is lost
- Implement a more versatile and reliable video capture with flow control so that it is more reliable and conforms to the common standards
- Optimize for use over lower-bandwidth networks by having more options either selectable by the user or automatically set.
- Test the setup in different environments to see its reliability. These environments can be places where the bandwidth available is very low or very high or simply difficult terrain for navigation or control.