

Android Reality RC Car

Department of Electrical and Computer Engineering

Fall 2012

Mohammed Samer Abbas

Debra Chen

Jonathon Green

Faculty Advisors: Professor Narasimhamurthi Natarajan, Professor Paul Watta

Abstract

The purpose of this project is to design an Android mobile technology controlled RC car. An Android camera phone and IOIO board are mounted to a RC car to serve as the camera for the behind-the-wheel view as well as the controller for the car's motors (both steering and drive). The IOIO board provides the ability to control input and output ports through Android programming. A hobby-grade RC car is used to take advantage of the easy-to-connect/modify setup up of the internal electronics as well as the existence of servos for proportional steering.

On the user-interface end, an Android Tablet serves as the remote control for the car by communicating wirelessly with the car-mounted device via an adhoc WIFI network. The user is able to control steering and speed by using touch controls. Users may also opt to enable tilt steering for a more immersive experience. Furthermore, a video stream from the point of view of the RC car is viewable on the User Interface.

Table of Contents

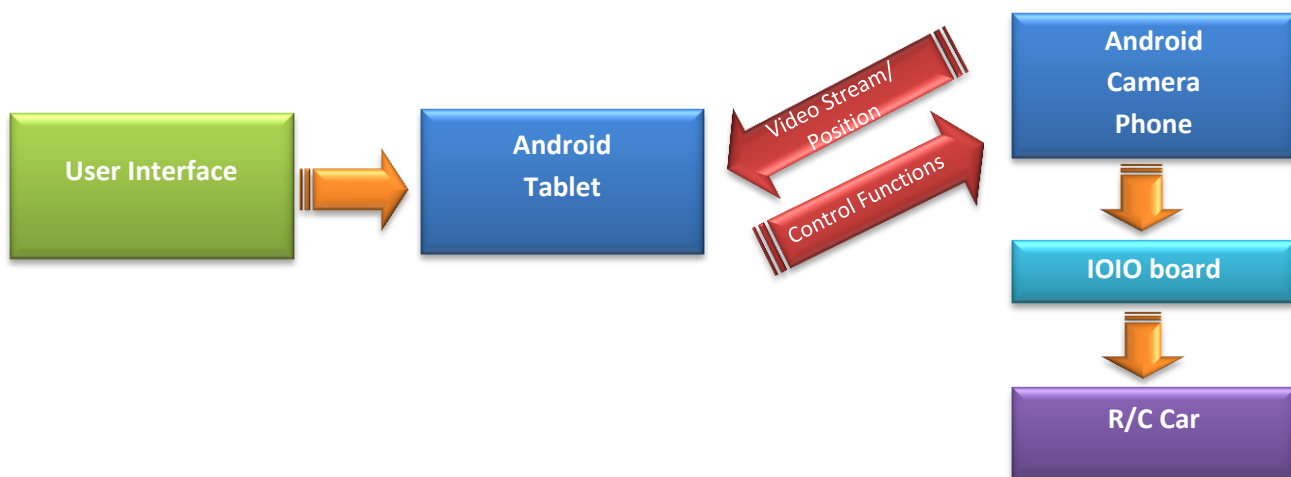
1. Introduction and Design Problem Statement	1
Figure1: Block Diagram of Android Reality Project	1
2. Analysis and Design	2
Wireless Data-Transfer between Android Devices	2
Android Circuit Control	4
Figure 2: IOIO Board Connections Block Diagram	4
Table 1: IOIO Board Pinout	5
User Interface	5
Figure 3: Tablet Control UI	6
Augmented Reality	6
Task Assignment	7
Final Specifications	8
3. Results	9
4. Ethics and Safety	10
5. Cost Analysis	11
Table 2: Product Feasibility	11
6. Conclusion	12
7. References	13
Appendix A: Device Specifications	14
Appendix B: Final Proposal	15
Appendix C: Advisor Meeting Sheets	21
Appendix D: Team Member Resumes	26
Appendix E: Android Application Code	31

Introduction and Design Problem Statement

The purpose of this project is to design a product that fuses the physics and exhilaration of the real world with the feature-rich world of Android mobile technology. Inspired by a love for cars and classical racing games like Mario Kart, this project aims to combine a high-performance radio-controlled (RC) car with Android devices to create an unparalleled gaming experience. The goal is to create an Android application that will allow the user to control an RC car using an Android tablet and its tilting functionalities while receiving a point-of-view video stream from the RC car. This application will implement the methods of Augmented Reality which will provide the user with a virtual racetrack that will overlay the point-of-view video stream coming from the RC car.

The overall operation of the Android Reality Car (AndReality) is illustrated in a block diagram in figure 1. An Android camera phone will be mounted inside of the RC car to serve two functions; point-of-view video streaming and RC car circuit control using an Android IOIO (pronounced “yo-yo”) board. A tablet will contain the main user-interface which receives the video stream and position data from the phone on the RC car and sends control commands to the phone on the RC car.

Figure 1: Block Diagram of AndReality Project



Analysis and Design

In order to achieve the goal of this project, the project was split into four major sub-systems: Wireless data-transfer between Android devices, Android circuit-board control, the User Interface, and Augmented Reality implementation. Each sub-system includes various tasks, challenges and design alternatives that must be considered in order to achieve the project goal.

Wireless Data-Transfer between Android Devices

The RC car will be controlled through wireless communication between the two Android devices. There are a few different ways of approaching this-- each having advantages and disadvantages.

WifiDirect is a feature available on Android Ice Cream Sandwich devices (ICS version 4.03) which allows two Android devices to communicate via an adhoc wifi connection. The benefit to this method is that it provides a range of up to 200 yards and can be set up anywhere. The downside to this approach is that the project would be limited to ICS devices which tend to be more expensive than older models.

Wireless 3g/4g communication is another option. The advantages of using the cellular network are that the range is virtually limitless and any version Android device can be used. The disadvantages are that mobile networks can be slow, hindering performance, and a cellular data contract is required, which can be costly.

The use of a dedicated router is yet another approach. A router could be used with a PC to create a private network through which the Android devices can connect to and communicate. The advantage of using a dedicated router is that it is the most cost effective choice since a data contract is not required and there is no limitation on the version of Android required. The disadvantages of this setup are that the range is the shortest of the other options and the requirement of a router and PC make the product less portable.

The project was approached with the main intent of using WifiDirect for communication between the Android devices. However, a better method was found that allowed compatibility with older versions of Android. Using the reflective capabilities of Java, some undocumented methods in the API were able to be exposed. After building an abstraction layer over the reflective code, a more readable version of the code was compiled as is shown below:

Basic method of extracting and setting up an access point using reflection

```
Method[] wmMethods = wifiManager.getClass().getDeclaredMethods();
for(Method method: wmMethods){
    if(method.getName().equals("setWifiApEnabled")){
        WifiConfiguration netConf = new WifiConfiguration();
        netConf.SSID = "AccessPoint";
        . . .
        boolean apstatus=(Boolean) method.invoke(wifiManager, netConf,true);
        . . .
    }
}
```

It can be seen that reflection is used to get an array of all methods from a *wifiManager*, then the array is searched for *setWifiApEnabled*. The configuration is set and reflection is used again to invoke the method.

Simplified version which is much easier to understand

```
public void createWifiAccessPoint() {
    if(wifiManager.isWifiEnabled())
        wifiManager.setWifiEnabled(false);

    WifiConfiguration netConfig = new WifiConfiguration();
    netConfig.SSID = sSsid;

    netConfig.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN);
    netConfig.hiddenSSID = sIshidden;

    if(setWifiApEnabled(netConfig, true))
        Log.d(TAG, "Access Point created");
    else
        Log.d(TAG, "Access Point creation failed");

    while(!(Boolean)isWifiApEnabled()){
    };
}
```

A global *wifiManager* object is used by this method and other methods in this class. These other methods Extract and use the methods from this object. This hides most of the complexity so the problem at hand can be the focus.

When streaming video, factors such as resolution, frame rate, and bandwidth must be considered. Since wifi is being used to transfer data, a large enough bandwidth exists which allows the streaming of relatively high-resolution, high frame rate video. The minimum specifications of the video stream are a 640x480 resolution at a frame rate of 24fps. This will limit the amount of bitrate required to within 600-700kbps which is well below the data rate of wifi.

Android Circuit Control

In order to control the mechanical movements of the RC car, the Android camera phone that is mounted on the RC car must receive commands from the tablet and control the motor and servo on the car. The simplest, most effective way to achieve this is by using an Android IOIO board. The IOIO board is an input/ output circuit board that connects to any Android 1.5 device or newer via USB to allow developers to control physical circuits through Android Java code. The IOIO board comes with an application programming interface (API) that contains easy to implement input/ output software functions.

For the purpose of this project, an IOIO board is mounted with the Android camera phone on the RC car, replacing the RC car's controller. The IOIO board is powered by the on-board 5V source provided by the Electronic Speed Control unit which was originally used to power the RF transmitter of the car. The IOIO board is used to control the steering servo as well as the driving motor by generating appropriate PWM signals based on user input. The PWM signal mapping is outlined as follows:

- Drive Motor
 - 1.5ms pulse width = stop
 - 1.5-2ms pw = proportional acceleration
 - 1.5-1ms pw = proportional reverse acceleration
- Servo Motor
 - 1.5ms pw = Straight
 - 1.5-1ms pw = Right (proportional)
 - 1.5-2ms pw = Left (proportional)

Figure 2: IOIO Board Connection Block Diagram

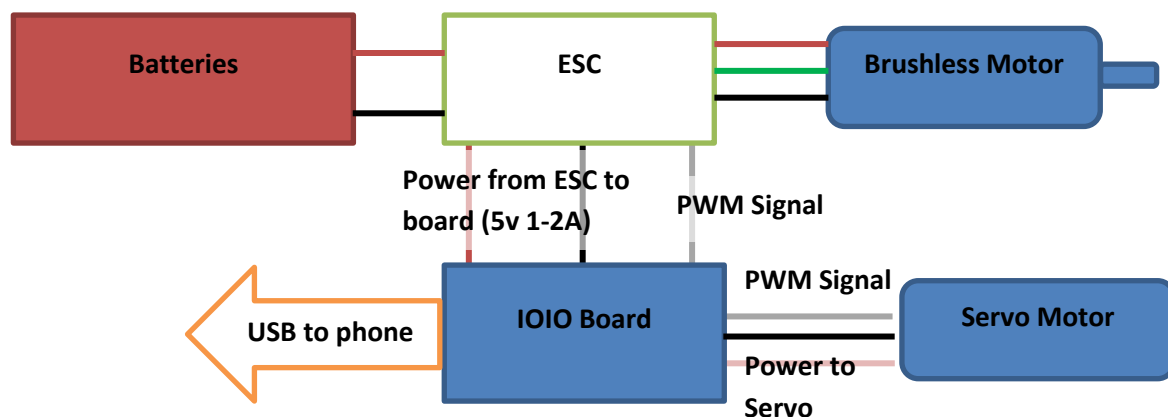


Table 1: IOIO Board Pinout

Pin 1 (Ground)	ESC Ground (black)
Pin 2	ESC signal (white)
Pin 3 (Vin)	ESC 5V (red)
Pin 4 (Ground)	Servo Ground (black)
Pin 5	Servo signal (white)
Pin 6 (Vout)	Servo 5V (red)
USB	To Android Phone

Since a hobby-class RC car is being used, all of the electrical components are easily accessible and are made to connect to different controllers. Thus, the performance and power consumption of the RC car did not differ from the manufacturer values. This holds true for the Android devices as well. The performance specifications can be seen at the end of this section.

User Interface

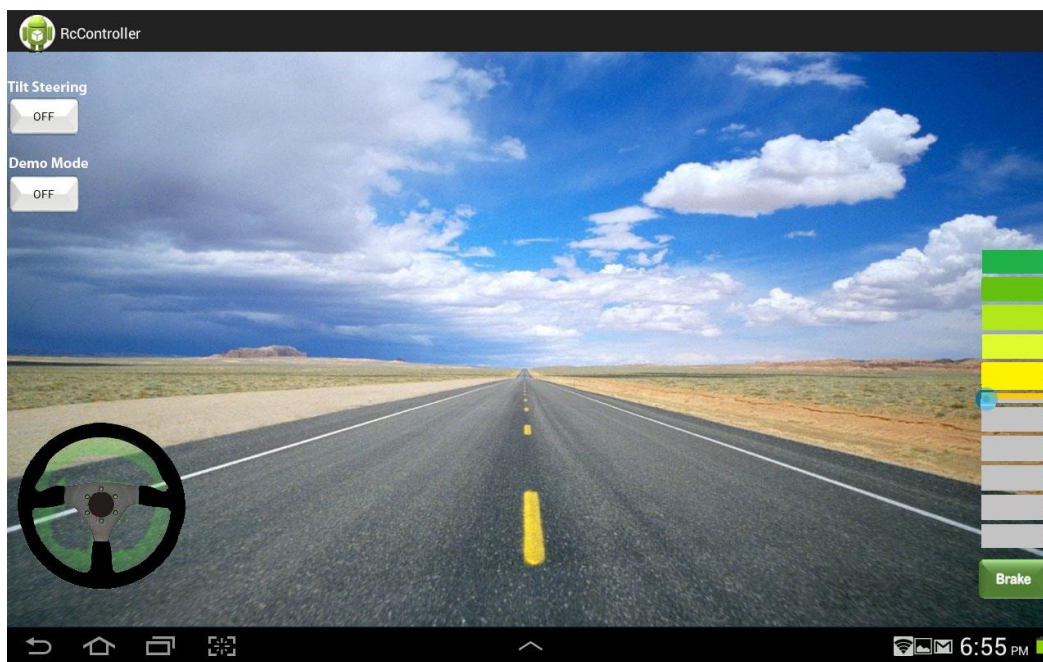
A custom User Interface was created for this project. The steering wheel is a png image that turns to follow the user's touch. Once the device gathers the x and y coordinates from the touched surface, it resumes the thread that redraws the steering wheel. The canvas is always cleared to prevent multiple steering wheel images from appearing.

SurfaceView is used to provide a dedicated drawing surface and control the size and format of the surface. The surface is ordered so that it is behind the window, holding its SurfaceView. The steering wheel image is the SurfaceView. The window is the video streaming in the background. The SurfaceView punches a hole in its window to be displayed. The origin is set to be in the middle of the steering wheel picture. Steering wheel image is scaled to size with the cursor centered to the picture. Then, according to the designated quadrant, the correct angle is calculated at the point where the user pressed down. When the angle is obtained from touch, the progress value (0-100) of the slider is calculated. When the angle is between 0 to 135 degrees, the RC car turns right. When the angle is between 225 to 359 degrees, the RC car turns left. The RC car steers in between 0 to 135 degrees and 225 to 359 degrees because this ensures the RC

car does not suddenly change directions from the user inputs. Thus, the user has steady control over RC car steering.

Moreover, customized user interface elements can make the application look more attractive and game-like. To create the slider, the slider background and the gradient are placed in the drawable folder. The slider background is a nine patch image because a nine patch image enables the image to be stretched, repeated, and scaled properly. To indicate the progress, the same slider image is colored with grey boxes to overlay on top of the slider background. Within the activity_drive.xml, these elements are brought together to create the finished look. Furthermore, the customized buttons are also in the drawable folder. The button look can change according to pressed and not pressed states, which are declared using selectors.

Figure 3: Tablet Control UI



Augmented Reality

Perhaps the most challenging software task in this project is the implementation of Augmented Reality. “Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are *augmented* by computer-generated sensory input such as sound, video, graphics or GPS data”². There are a few AR APIs for Android in existence at the moment that provide a basic framework for what this project aims to accomplish.

Raveneye is one such framework and it is the framework that was planned to be utilized for the purposes of the AndReality project.

The augmented virtual environment intended to be created for this project would initially consist of checkpoints for the RC car to drive through. These checkpoints would have GPS locations that would determine where the checkpoints appear on the user-interface screen. The user would be able to create his or her own track by “walking” the RC car through the track and setting checkpoints at specific GPS locations. Since GPS is only accurate to within 10 meters, the checkpoints have to be wide. Adding in simple image processing, such as detecting the difference between concrete and grass, could help narrow the required checkpoint width. Furthermore, an even more accurate measure of location may be attained using the accelerometer on the RC car-mounted camera phone and complex software algorithms; however, this goes beyond the scope of the project.

In addition to the checkpoints, a heads up display (HUD) consisting of vehicle speed overlays the point-of-view video stream. Further additions of HUD parameters as well as game pieces would also be possible to add to the base developed in this project. The aim of this basic augmented reality environment is to create a base for a vast array of possible game applications. For example, a Mario Kart game may implement this base environment and add “power-ups” at certain locations on the track that give a temporary speed boost (or loss) to the RC car. In addition, “shadow racing” (racing a virtual version of one’s previous best lap time) could be implemented by simply recording the best run and pitting the user against him or herself. Unfortunately, due to the amount of time spent on getting the live video stream to work, the augmented reality portion of this project was not implemented.

Task Assignment

Each group member was assigned a task based on his/her strengths and experience with hardware and software. The tasks were divided into chunks that could be completed simultaneously and then combined in the end. Below is an outline of each member’s responsibilities in completing this project.

Samer Abbas

- Modified the RC car to take commands from the IOIO board

- Used existing Electronic Speed Controller as motor driver
- Drive and Steering servo take in 50Hz PWM Signal with pulse width 1-2ms (5%-10%)
- Created tilt control steering
 - Converted x-axis accelerometer data into servo motor control inputs
 - Tilt control can be turned on/off and used in place of touch steering
- Created UDP motor control methods
 - Send wireless command packets converting slider data into motor control inputs

Debra Chen

- Created app to display the video stream on tablet
- Developed the beta version of the user interface
 - Designed layout of UI for to be simple and easy to use
- Developed RC controller interface
 - Created custom UI elements
 - Programmed movable functionality for custom controls

Jonathon Green

- Worked on communication between devices
 - Transmitting data from tablet to car-mounted phone
 - Adhoc Wifi connection between devices
 - TCP and UDP data transmission
- Video Encoding for easy data transfer
 - Vitamio streaming plugin
 - Low cpu usage to save battery life

Final Specifications

Project Specific Specifications:

- IOIO board used as a motor controller on RC car.
 - Requires 5~15 VDC input (5V supplied by RC car ESC)
- High-performance relatively large RC car

- Motor: 36 x 65 540L Brushless motor 2150KV (PWM Control)
- Battery: 7.4V 3300mAh Li-Po
- Servo: Metal geared steering servo (PWM Control)
- One Android version camera phone
 - 640x480 resolution at a frame rate of 15fps
 - 500kbps bit rate (well below wifi limit)
- One Wifi Hotspot capable Android device
- 200 yards of wifi connection range
- Wifi connection is approximately 22mbit/sec with a max of 54mbit/sec

See Appendix A for device specifications

Results

For the purpose of this project, a custom method was created for establishing a connection between two android devices using wifi. The original plan was to use Wi-Fi Direct, a technology released in android version 4.0, to manage a peer-to-peer (P2P) Wifi connection. This P2P connection would allow the user-operated android phone to wirelessly send commands to a car-mounted android phone, and the car-mounted phone to send video and other data to the user-operated phone. It was challenging to find devices compatible with the new Wi-Fi Direct technology and many hours were spent hacking the devices that were available but to no avail.

Then a method of using undocumented API features with reflective programming was stumbled upon. This allowed one of the devices to be used as a Wi-Fi access point, so the other device can connect directly via wifi. In this way, the tablet (controller) provides the wifi signal and the car mounted device (car) connects to it. During the connection process, the car sends a ping message to the controller, since the IP of the controller will always be 192.168.xyz.1 (where xyz is same subnet as the car). The controller waits for this ping message and uses it to find the car's IP address.

For the purposes of time, the open source project SpyDroid was used as the basis of the streaming code. SpyDroid has a complete implementation of RTSP on Android, so their well written code was used to handle encoding the video output of the car phone into a stream of UDP packets.

On the controller side, these packets are then collected and processed by the Android or Vitamio VideoView which displays the video stream on the controller. The VideoView is given the IP address of the car and after a few seconds it displays the stream. This gives the user a first person view of the RC car.

Vitamio is an Android plugin that extends and improves on the default Android video and media API. It can be used as a drop-in replacement and has a few advanced options that default Android doesn't. Using this plugin greatly enhanced the responsiveness and clarity of the controller's view without requiring major changes to code.

Once the devices are connected, data begins streaming between them. The car sends video data to the controller using Real Time Streaming Protocol (RTSP) and the controller sends the car steering and acceleration commands using a custom minimal protocol. This protocol allows the controller to send datagram packets with the message D<two-digit number 00-99> for drive based on the location of the accelerator slider on the UI or S<two-digit number 00-99> for steering (generated by the touch steering wheel or accelerometer). These messages are then encoded to arrays of bytes and sent to the car. The car decodes these messages by converting them to a number between 0 and 1 to be transmitted as part of the PWM control signal that is sent to the IOIO board.

Ethics and Safety

As with any product design, engineers must always consider safety when developing for a customer base. This is especially important when it comes to products like the RC car used in this project that has a significant mass and can travel at dangerously high speeds. When designing the control commands for the car, safety was the first item addressed. There were two levels of safety implemented in the code; development failsafes and final product failsafes. The development failsafes were implemented to prevent the car from moving out of control during design and testing. This was done by causing all control commands to default back to the neutral position if there was no user interaction. Once the user interface was finalized, the final product safety feature was implemented. The receiver on the RC car constantly checks for communication from the tablet and should communication stop for longer than one second, all of the controls are returned to the neutral position, bringing the car to a stop. This safety feature

eliminates the chance of injury or damage due to controller malfunction or connection loss, making this a marketable product.

Cost Analysis

One of the most impressive outcomes of this project is the potential marketability of a product based on the project. The goal was to develop an add-on system that could be used on any hobby-class RC car without any major modifications necessary. As it stands, one could unplug the RC car from the system developed in this project and plug back into the original system without modifying anything.

In a commercial setting, one could present the product to market in two forms. The first option is to post the Android application to the Android Market (Google Play Store) along with a digital instruction manual on how to set up the system with relative ease to be used on users' own RC cars. This do-it-yourself approach requires no overhead from the company standpoint and minimizes the costs to the customer while requiring the customer to have minimal soldering and hands-on knowledge. The rest of the required hardware (Android phone/ tablet, IOIO board) would have to be purchased separately by the customer. The second approach is to sell an add-on kit that would simply plug into customers' devices and work. This option would include an IOIO board with connections already soldered as well as the mounting hardware for mounting a phone to the customers' RC cars. Both options require that customers have their own hobby-grade RC car and Android devices. The following table illustrates a rough estimate of the cost of the parts that comprise the systems developed along with each system's marketable price, cost to customer and profit margin.

Table 2: Product Feasibility

Do-it-yourself system	Cost to customer	Cost to company	Prepackaged plug-in system	Cost to customer	Cost to company
App w/ instructions	\$9.99	0	Complete package	\$99.99	See below
IOIO board	\$49.99	0	IOIO board + soldering labor	0	\$54.99
Mounting hardware	\$0-\$10	0	Mounting hardware	0	\$10
			Packaging	0	\$5
Total Cost	\$59.98-\$69.98	0	Total Cost	\$99.99	\$69.99
Profit/ product	\$9.99		Profit/ product	\$30	

Conclusion

This project provides an outlook towards a future of high-end toys that can be controlled through Android and introduces a new form of gaming using Android devices to control vehicles. It also demonstrates that Android can be used for external circuit control. Implementing video streaming on android devices was the biggest challenge as the lag of the video stream was hard to reduce because android VideoView has a built in buffer. The video has to first fill the buffer before displaying on screen which results in a lag. Nevertheless, a marketable, stand-alone product was successfully developed and tested which can also serve as a base to build upon for future products.

References

- 1 http://www.pcworld.com/article/254888/3g_4g_performance_map_data_speeds_for_atandt_sprint_t_mobile_and_verizon.html
- 2 http://en.wikipedia.org/wiki/Augmented_reality
- 3 <http://www.streaminglearningcenter.com/articles/streaming-101-the-basics---codecs-bandwidth-data-rate-and-resolution.html>
- 4 <http://www.nitrorcx.com/52c76-maddrift-350tt-orange-brushless-limite.html>
- 5 <http://webtutsdepot.com/2011/08/20/android-sdk-accelerometer-example-tutorial/>
- 6 <http://stackoverflow.com/questions/15141134/android-sensors-to-control-game>
- 7 http://www.anddev.org/udp-networking_-_within_the_emulator-t280.html
- 8 <http://www.helloandroid.com/tutorials/simple-udp-communication-example>
- 9 <https://github.com/ytai/ioio/blob/master/software/applications/IOIOSimpleApp/src/ioio/examples/simple/IOIOSimpleApp.java>

Appendix A: Device Specifications

Samsung Galaxy 10.1 Tablet

- Android version: 4.0.4
- Battery: 7000 mAh
- Weight: 565 g
- Screen size: 10.1in
- RAM: 1 GB
- Storage: 30 GB
- Processor: 1 GHz Nvidia Tegra 2
- Camera: 3.1 Mega-pixel

Samsung Europa

- Android version: 4.0.4
- SD card: 1 GB
- Battery: 1200 mAh
- Weight: 102 g
- Screen size: 2.8in
- RAM: 170 MB
- Processor: 600 Mhz
- Camera: 2 Mega-pixel

MadDrift 1/8 Scale R/C Car

- Length: 620MM
- Width: 310MM
- Height: 175MM
- Wheelbase: 330MM
- Track width: 310MM
- Gear Size: 12T Pinion gear
- Gear ratio: 12.68:1
- Motor: 36 x 65 540L Brushless motor 2150KV
- Battery: 7.4V 3300mAh Li-Po
- Servo: Metal geared steering servo

Appendix B: Final Proposal

ECE 4981 ELECTRICAL/COMPUTER ENGINEERING DESIGN Fall 2012

NAME: Mohammed Samer Abbas, Jonathon Green, Debra Chen

TITLE: Android Reality Car

ADVISORS: Dr. Natarajan and Dr. Watta

AIMS AND GOALS OF THE PROJECT:

The goal of the proposed project is to design an Android mobile technology controlled RC car that employs augmented reality. The basic idea is to mount an android device on an RC car to serve as the camera for a behind-the-wheel view as well as the controller for the car's motors (both steering and drive) via an IOIO (pronounced yo-yo). Using any android phone, the user will be able to control a simple RC car the same way they do in a video game. This means that the user will see a virtual race-track that overlays the real image from the camera in the form of pre-set checkpoints for the user to go through.

Raveneye shall serve as the base. Raveneye is an augmented reality Android application that focuses on Point of Interest discovery and way-point navigation through overlays on top of a camera surface. The new Android Wi-Fi direct technology can enable the RC car mounted device to communicate with the user interface, and Wi-Fi direct has a range of just over 200 yards and/or the mobile cell network which provides a big area for the user to control the RC car. To connect the electronic circuit to the Android device, IOIO allows the user to control the car from an Android application. On the user-interface end, another Android device (most likely a tablet) will serve as the remote control for the car by communicating wirelessly with the car-mounted device. The driver can simply tilt the Android device to steer the RC car in different directions to provide real-life simulated control over the RC car. USB steering wheel and gas pedal set-up may be integrated to the project for more realistic simulation control over the RC car as a supplement depending on time constraints.

Furthermore, the amount of latency, the easiness in maneuverability of RC car, and the user-friendly design of the UI are key factors to measure the design performance of the project. The amount of latency between the driver and the RC car shows the speed of the data transfer

which is crucial to the connection between the android devices. The maneuverability of the RC car to driver demands demonstrates how well the IOIO works. The user-friendliness and the usability of the UI should bring an enjoyable experience for the user when controlling the RC car. The results of design performance can be analyzed through testing and trial and error. If the user can control the car effectively, then the goal of this project is achieved.

RELEVANT PRIOR WORK:

Both Jonathon and Samer have developed an android project. In the summer of 2011, they worked on a campus project together. The project's aim was to develop useful Android applications for college students. Samer and Jonathon's application was called *Lectmarker* and the project was showcased in the engineering magazine. Furthermore, Jonathon developed several example android projects for Paul Watta's mobile computing class, as well as worked over the summer with *Dangos Mobile*, a startup company. He developed a project demo for their theme park navigation app. Moreover, Debra worked at Ford Motors Company as a summer intern in 2011. She assisted Ford engineers to write software specifications for the Manual Hybrid Transmission.

The team members have completed the following classes:

- *Circuits (ECE 210): Samer, Debra, John*
Gain a basic understanding of circuits to wire the IOIO board to the RC car.
- *Electronic Circuits I (ECE 311/ECE 414): Samer, Debra, John*
More in-depth knowledge on circuits and will use transistors as a switch between the battery and the motor.
- *Introduction to Microprocessors (ECE 273/ECE 372): Samer, Debra, John*
Become familiar with functionality of processors and how data transfers occur.
The microprogrammed control unit (MCU) on the IOIO board supervises all SPE elements and to direct and initiate all data transfers between these elements.
- *Systems Design and Microcontroller (ECE 4951): Samer*
Learn to program the IOIO board to interface with the hardware on the car.

- *Embedded System Design (ECE 473): Debra, John*
Programming the IOIO board will be easier once we understand the low level coding.
- *Computer Networks and Data Communications (ECE 471): Debra, John*
Understand the transmission of data over a wifi network and how to set up the wifi network.
- *Computer Methods in ECE (ECE 270): Samer, Debra, John*
Learn to program and understand how to make a logical program to achieve an objective.
- *Advanced Software Techniques in Computer Engineering (ECE 370): Debra, John*
Learn object orientated programming and learn to use different software tools such as qt creator and to how to display things on the screen and how interfacing works which would be crucial when developing the user interface design.

PRELIMINARY IDEAS AND METHODS:

Jonathon Green:

- Will work on communication between devices.
- Will develop a method for viewing the augmented reality on a tablet.
- Will help Samer with coding the java steering objects and car controller.

Samer Abbas:

- Will modify the RC car to take commands from the IOIO board.
- Will create java objects for steering the RC car using the IOIO.
- Will help with the creation of the tilt controller.

Debra Chen:

- Will develop user interface design for Android application.
- Will work on the USB controller and tablet tilt controller.
- Will help develop the augmented reality view.

Specifications:

- Use of Android WIFI direct technology for data transfer and communication.
- IOIO board used as a motor controller on our RC car.
- High-performance relatively large RC car
- One Android version 4.0+ camera phone
- One Android version 4.0+ tablet
- 200 yards of wifi connection range
- Wifi connection is approximately 22mbit/sec with a max of 54mbit/sec

Device Specifications:***Samsung Galaxy 10.1 Tablet***

- Android version: 4.0.4
- Battery: 7000 mAh
- Weight: 565 g
- Screen size: 10.1in
- RAM: 1 GB
- Storage: 30 GB
- Processor: 1 GHz Nvidia Tegra 2
- Camera: 3.1 Mega-pixel

Samsung Europa

- Android version: 4.0.4
- SD card: 1 GB
- Battery: 1200 mAh
- Weight: 102 g
- Screen size: 2.8in
- RAM: 170 MB
- Processor: 600 Mhz
- Camera: 2 Mega-pixel

MadDrift 1/8 Scale R/C Car

- Length: 620MM
- Width: 310MM
- Height: 175MM
- Wheelbase: 330MM
- Track width: 310MM
- Gear Size: 12T Pinion gear
- Gear ratio: 12.68:1
- Motor: 36 x 65 540L Brushless motor 2150KV
- Battery: 7.4V 3300mAh Li-Po
- Servo: Metal geared steering servo

Logitech Driving Force GT USB Steering Wheel

- USB Connectivity: USB 2.0
- AC Adapter: Input 100-240V~50-60Hz
- Output 24V/ 750mA
- USB 900 degree steering wheel and gas pedal setup (optional)

ENGINEERING STANDARDS:

- Android SDK version 15
- Eclipse IDE software
- USB
- Java
- WIFI
- TCP/IP
- IOIO API

PROJECT COST ANALYSIS:

- Android tablet
 - borrowed from university
- Android phone
 - borrowed from university
- Eclipse IDE software and Android SDK version 15
 - free
- IOIO board
 - \$50 (Purchased)
- High-performance relatively large RC car
 - \$260 (Purchased)
- Li-PO Charger For RC Car
 - \$50 (purchased)
- USB 900 degree steering wheel and gas pedal setup
 - \$150 (Purchased)

TIME SCHEDULE:

September	<ul style="list-style-type: none">▪ Created project idea
October	<ul style="list-style-type: none">▪ Research technology<ul style="list-style-type: none">▪ find good documentation▪ code examples▪ Begin development of software.▪ Have code base started and basic application done by the end of the month.▪ Get all members up to speed on Android development.
November	<ul style="list-style-type: none">▪ Work on communication between Android devices.▪ Start developing steering and IOIO to RC car api.
December	<ul style="list-style-type: none">▪ Finish communication and steering aspects of the project.▪ Begin integrating with the <i>raveneye</i> augmented reality.
January	<ul style="list-style-type: none">▪ Create the beta version of the application.<ul style="list-style-type: none">• Should be able to steer and stream video from the RC car.▪ Will begin testing and bug fixing.
February	<ul style="list-style-type: none">▪ Should have project nearly completed.▪ Will add extra features and/or clean up the application.<ul style="list-style-type: none">▪ Make code pretty▪ Make interface more usable▪ General small improvements.
March	<ul style="list-style-type: none">▪ Extra time if project is delayed for whatever reason.
April	<ul style="list-style-type: none">▪ Extra time if project is delayed for whatever reason.

Appendix C: Advisor Meeting Sheets

ECE 498X ELECTRICAL/COMPUTER ENGINEERING DESIGN
INDIVIDUAL PROGRESS MONITOR
Fall Term 2012

DATE: 11/21/12
NAME: Jonathan
TITLE: Android Reality Car
ADVISOR: Wata

DESCRIBE BRIEFLY YOUR PROGRESS TOWARD THE PROJECT:
Just hacked some phones

DESCRIBE PROBLEMS/DIFFICULTIES ENCOUNTERED:
Getting a wifi direct phone

COMMENTS AND SUGGESTIONS FROM ADVISOR:
Please put a budget together for the items you need and obtain funding (see Prof. Miller) Make sure all of the components will work together!

Student signature and date
[Signature]

Advisor signature and date
[Signature]

Date submitted to coordinator
11/21/2012

xii

ECE 498X ELECTRICAL/COMPUTER ENGINEERING DESIGN
INDIVIDUAL PROGRESS MONITOR

Winter Term 2013

DATE: 1/28/13

NAME: Jonathon Green

TITLE: And reality car

ADVISOR: Watta

DESCRIBE BRIEFLY YOUR PROGRESS TOWARD THE PROJECT:

- We've got Video communication working, as well as, the motor controller.
- We need to decrease the latency on the video, and finish the wifi communication

DESCRIBE PROBLEMS/DIFFICULTIES ENCOUNTERED:

- When connecting to the wifi hotspot some devices don't get an IP address.
- There's a lag of between 2 and 8 seconds on streaming video.

COMMENTS AND SUGGESTIONS FROM ADVISOR :

Yes, need to decrease latency!
I would like to see demo next time.
See Prof. Xiang

Jonathon Green
Student signature and date

Watta
Advisor signature and date

1/28/2013
Date submitted to coordinator

**ECE 498X ELECTRICAL/COMPUTER ENGINEERING DESIGN
TEAM PROGRESS MONITOR**

Winter Term 2013

DATE: 1/29/13

NAME: Samar Abbas, Deyan Chen, Jon Green

TITLE: Android Reality Car

ADVISOR: Prof. Natarajan

DESCRIBE BRIEFLY OVERALL PROGRESS TOWARD THE PROJECT:

- Completed IOIO board RC control (steering and drive)
- Established ad-hoc wifi network
- Successfully streamed video across wifi connection

DESCRIBE PROBLEMS/DIFFICULTIES ENCOUNTERED:

Video stream has ~ 8 sec lag

COMMENTS AND SUGGESTIONS FROM ADVISOR :

I would like to see a short demo and source code so
I can give you some pointers on UDP/IP ~ TCP/IP

M Natarajan

Student signatures

Advisor signature and date

Date submitted to coordinator

ECE 498X ELECTRICAL/COMPUTER ENGINEERING DESIGN
TEAM PROGRESS MONITOR
Winter Term 2013

DATE: 2/14/13

NAME: Samer Abbas, Debra Chen, Jon Green

TITLE: Aug Reality Car

ADVISOR: Prof. Natarajan

DESCRIBE BRIEFLY OVERALL PROGRESS TOWARD THE PROJECT:


- Completed UDP connection between phone and tablet
- Tablet wirelessly sends commands to phone and in turn, controls the RC car

DESCRIBE PROBLEMS/DIFFICULTIES ENCOUNTERED:

still trying to solve video lag problem

COMMENTS AND SUGGESTIONS FROM ADVISOR :

Consider space filling subsampling. I can give a short tutorial. It is fairly simple and cool too.


Student signatures


Advisor signature and date

Date submitted to coordinator

ECE 498X ELECTRICAL/COMPUTER ENGINEERING DESIGN

TEAM PROGRESS MONITOR

_____ Term 20__

DATE: 4/12/2013

NAME:

TITLE:

ADVISOR: Watta.

DESCRIBE BRIEFLY OVERALL PROGRESS TOWARD THE PROJECT:

- Finalized user interface
- added tilt steering
- physically mounted hardware onto car
- RC car is fully functional other than video streaming

DESCRIBE PROBLEMS/DIFFICULTIES ENCOUNTERED

- problems ~~with~~ integrating video stream with app.

COMMENTS AND SUGGESTIONS FROM ADVISOR :

Debra Chen

Student signatures

Advisor signature and date

Date submitted to coordinator

Appendix D: Student Resumes

MOHAMMED SAMER ABBAS

7744 Appoline St. Dearborn, Michigan 48126,
(313)492-0347 msabbas@umd.umich.edu

OBJECTIVE: To pursue an Electrical Engineering position that will allow me to push the world's technology forward.

EDUCATION

- ◆ University of Michigan- Dearborn
 - Bachelor of Science in Electrical Engineering (Expected Graduation 4/2013)
 - Cumulative GPA of 3.76, Major GPA of 3.82
 - Relevant courses: Power electronics, Renewable Energy and Power Systems, Electric Machines and Hybrid Drives, Automatic Control Systems
 - Member of Eta Kappa Nu (HKN), Tau Beta Pi, Golden Key Honor Societies, Dean's Scholarship, Dean's list every semester (GPA above 3.5), Michigan Merit Scholarship
- ◆ Massachusetts Institute of Technology (MITx/ Harvard University edX)
 - Earned a credential certificate for completing "Circuits and Electronics" online course
- ◆ Graduate of Fordson High School/ Dearborn Center for Math Science and Tech. (2010, 4.06 GPA)
 - Dually Enrolled in Henry Ford Community College
 - 1st Place Science Fair award in engineering (DCMST)
 - Superintendent Honors, Phi Beta Kappa, National Honor Society, Business department award, honor roll

SKILLS

Computer: CAN Software, Java, C, C++, Android, html, Movie Maker, Microsoft Office 2000-2010, network setup, web design, PSPICE, hardware installation.

Electrical: Automotive electronic wiring including stereo and HID conversion kit installation as well as ignition system wiring. Designing, programming, soldering, constructing, repairing microprocessor circuits.

Mechanical: Rebuilt and modified motorcycle engine for use on a go kart, rebuilt carburetors, repaired motorcycles and jet skis, automotive body work, car engine work.

EMPLOYMENT

- ◆ *Test Engineering Co-op*-Takata Inc. (1/2012-5/2012)
 - Worked in active safety team on lane detection technology
 - Performed design validation tests and submitted reports of results
 - Tested, diagnosed, repaired and modified test equipment when necessary
- ◆ *Android App Developer*-University of Michigan-Dearborn/ Ford (5/2011 – 12/2011)
 - Developed a mobile application on Android Platform for use by University students with intent of porting to Ford Sync system

- Worked in teams using SVN to manage program updates
- ◆ *President/ Business Owner*- Turbo Water Sports LLC (5/2011-present)
 - Founded, Incorporated and manage Jet Ski Rental company
 - Marketed business via the internet and created business website
 - Manage finances, deliveries and customer satisfaction
- ◆ *Cashier/ Cook*- Manhattan Fish and Chicken (12/2009-6/2010)
- ◆ *Community Service* – ICD/ Al-Ihsan Academy (1/2005 – 11/2011)
 - Coached youth basketball for two years/ head of youth basketball program
 - Tutored children ages 5 – 15

Debra Gi Chen

28867 W. King William Dr • Farmington Hills, MI 48331 • 248-994-1905 •

chendj@umich.edu

OBJECTIVE

To gain a full time position in the Computer Engineering or Electrical Engineering field.

EDUCATION:

University of Michigan – Dearborn, Dearborn, MI Expected Graduation May

2013 B.S.E. in Computer Engineering and B.S.E. in Electrical Engineering

G.P.A.: 3.70 / 4.00,

Dean's List - Fall 2009 – Present

Course Projects:

Battleship Game for Advanced Software Techniques

- Created a single player game based on the classic Battleship rules
- The game has built in algorithms to check for collision detection with walls and other ships

Circuit Simulation for Circuits Course

- Built circuit simulation in PSpice
- Built the physical circuit with a photoresistor connected to a buffer amplifier.

TECHNICAL SKILLS

Languages: C, C++, Java, HTML, Matlab, Labview

Applications: Microsoft Word, PowerPoint, Excel, Chinese pinyin characters writing tools, Pspice, Xilinx, Mathematica

Hardware: Digital Multimeter, Oscilloscope, CoolRunner II CPLD Board, HC11 Board, Spartan3A FPGA Board

WORK EXPERIENCE

Ford Motor Company, Dearborn, MI 04/12 – 09/12

Research Engineer Intern

- Worked under the Ford and Toyota engineers on the Manual Hybrid Transmission project which was a joint venture between Ford and Toyota.
- Recorded Meeting Minutes for every meeting.

Worldwide Interpreter Inc, Grosse Pointe Woods, MI 07/09 – Present

Language Interpreter

- Interpret languages either in person or through phone services
- Translate court documents between English and Chinese for the State of Michigan.

HONORS AND ACTIVITIES

- A member of The Golden Key International Honor Society;
- A member of The Electrical and Computer Engineering Honor Society (Eta Kappa Nu)
- A member of The Institute of Electrical and Electronics Engineers (IEEE)
- UM-D Honors Program since fall 2009

Work Experience

Dangos Mobile Inc. (May 2012 – Present)

- Worked on a mobile application targeted at amusement parks.
- Made a GPS data recording app and Demo app for Android.

University Project (June 2011 – December 2011) (7 months)

- Created a lecture bookmarking app for Android.
- It recorded, played lectures and allowed you to save notes at different points in the lecture.

Lab Assistant (February 2012 – April 2012) (3 months)

- Created lab tutorials and example Android applications for Professor Watta.
- Tutorials include: RSS reader, Slider Image rotator, onClick onTouch, xml reader/writer

Professor's Project (May 2012 – July 2012) (3 months)

- Wrote a basic image analyzing Android app for Professor El Kateeb.

Main Street Pizza in Plymouth (May 2012 – Present)

- Delivered pizzas, helped to prep, and answered phones.

Skills & Experience

Linux

- Everyday use for 5 years
- Ubuntu, CentOS, Arch, Gentoo

Android Dev

- 1.5 years experience
- University Projects, Internship

Various Languages & IDE's

- Scheme, JavaScript, CUDA, C, C++, Prolog, Python, PHP, Java, Perl, OpenMP, HTML, MATLAB
- NetBeans, Eclipse, Visual Studio, Dev-C++, Geany

Courses & Projects

Cloud Computing (taking)

- Amazon EC2 project
- Knowledge of the workings of the Cloud
- Making a video game administration system

Mobile Computing

- Several Android and Mobile Web Projects
- Developed course material

Computer programming I & II

- C / C++, Qt framework, Dev-C++
- Concepts of Object Oriented Programming, memory management, lists, hashmaps, the basics

Jonathon Green

OBJ A development position where quick learning and creativity are needed.

SUM I have years of hands on experience with Linux, Android, and many kinds of programming.

EDU U of Michigan Dearborn:
BSE in Computer Engineering
GPA of 3.29

PHP <https://github.com/radikll/jargon-parser>
Mobile site written for the Jargon file.

C <https://github.com/radikll/auto-music>
Generates random music using music theory.

JAVA <https://github.com/radikll/catalog-xml-for-music>
Android app that reads and writes to music catalog xml format.

INFO <http://lnkd.in/CR2RMH>
Check out my professional profile.

<https://github.com/radikll>
Or repositories.



Linkedin



Resume (current)

(734) 589-5632

greenbj@engin.umd.umich.edu

838 Harding St, Plymouth, MI 48170

Appendix E: Android Application Code

[Java Files \(Click to Navigate\)](#)

MainActivity.java (page 34)

ControlActivity.java (page 36)

CarActivity.java (Page 50)

Slider.java (Page 58)

WifiAP.java (Page 60)

[MainActivity.java](#)

```
package com.seniordesign.rccontroller;
```

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.StrictMode;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
```

```
public class MainActivity extends Activity implements OnClickListener{
```

```
    private static final String TAG = "AND Reality Car";
    ImageView title;
    Button carB;
    Button controlB;
    TextView name1;
    TextView name2;
    TextView name3;
    MediaPlayer sound;
```

```
    @SuppressWarnings("NewApi")
    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    setContentView(R.layout.activity_main);

    //play music
    sound = MediaPlayer.create(MainActivity.this,R.raw.mario);
    sound.start();

    //create objects
    title = (ImageView)findViewById(R.id.imageView1);
    controlB = (Button)findViewById(R.id.buttonController);
    carB = (Button)findViewById(R.id.buttonCar);
    name1 = (TextView)findViewById(R.id.TextView1);
    name2 = (TextView)findViewById(R.id.TextView2);
    name3 = (TextView)findViewById(R.id.TextView3);

    controlB.setOnClickListener(this);
    carB.setOnClickListener(this);

    //So Android doesn't go into network lockdown
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();

    StrictMode.setThreadPolicy(policy);
}

@Override
public void onClick(View v) {
    sound.release();
    Log.v(TAG, "outside if" + v.toString());
    if(v.equals(controlB))
    {
        Intent i = new Intent("com.seniordesign.rccontroller.con"/*getBaseContext(),
ControlActivity.class*/);
        startActivity(i);
    }
    if(v.equals(carB)){
        Intent i = new Intent("com.seniordesign.rccontroller.rec");
        startActivity(i);
    }
}
}

```

```
}
```

ControlActivity.java

```
package com.seniordesign.rccontroller;

//import io.vov.vitamio.MediaPlayer;
//import io.vov.vitamio.widget.MediaController;
//import io.vov.vitamio.widget.VideoView;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.ServerSocket;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.res.Configuration;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.PorterDuff;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CompoundButton;
```

```

import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.MediaController;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.ToggleButton;
import android.widget.VideoView;

import com.example.andrealitycarinterface.utils.BitmapScaler;

/**This is the Controller Activity (Tablet) that sends wireless control commands to the
 * device on the RC car
 */
public class ControlActivity extends Activity implements OnTouchListener,
        OnSeekBarChangeListener, SensorEventListener,
        OnCheckedChangeListener, OnClickListener{

    private static final String TAG = "ContrActivity";

    private ViewGroup parent; //portion of the view which shows the steering wheel
    private SteeringSurface mySurfaceView;
    private float x,y;
    public boolean transmitS,demoMode=false;
    public int cnt=0,prevSteer,currSteer;

    private SensorManager sensorManager;
    private SeekBar driveSlider_;
    private ToggleButton toggleTilt;
    private ToggleButton toggleDemo;
    private Button brake;
    private SeekBar steerSlider_;

    public String message=" ";
    private String dprogStr;
    private String sprogStr;
    public boolean sensorsOn=false;

    private boolean isRunControlLoop = true;

    private VideoView mVideoView;

    // communication part
    private InetAddress carConn = null;
    public String SERVERIP = null;
    public static final int SERVERPORT = 4444;

    /**

```

```

    * Called when the activity is first created. Here we normally initialize
    * our GUI.
    */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /*if (!io.vov.vitamio.LibsChecker.checkVitamioLibs(this))
            return;*/

        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        setContentView(R.layout.activity_drive);

        toggleTilt=(ToggleButton)findViewById(R.id.toggleTilt);
        toggleTilt.setOnCheckedChangeListener(this);

        toggleDemo=(ToggleButton)findViewById(R.id.toggleDemo);
        toggleDemo.setOnCheckedChangeListener(this);

        brake=(Button)findViewById(R.id.buttonBrake);
        brake.setOnClickListener(this);

        parent = (ViewGroup)findViewById(R.id.wheel);

        mVideoView = (VideoView) findViewById(R.id.videoview);
        //mVideoView.setVideoQuality(MediaPlayer.VIDEOQUALITY_MEDIUM);
        mVideoView.setMediaController(new MediaController(this));
        //mVideoView.setBufferSize(0);

        mySurfaceView = new SteeringSurface(this);
        mySurfaceView.setOnTouchListener(this); //implement onTouch
        x=0;
        y=0;

        parent.addView(mySurfaceView);
        driveSlider_ = (SeekBar) findViewById(R.id.accSlider);
        driveSlider_.setOnSeekBarChangeListener(this);

        driveSlider_.setMax(100);
        driveSlider_.setProgress(50);
        //driveSlider_.setThumb(getResources().getDrawable(R.drawable.thumb5));

        ProgressDialog progress = new ProgressDialog(this);
        progress.setMessage("Waiting for Connection...");

```

```

new MyTask(progress).execute();

//Loop to snap the controls back to neutral position when not being pressed

Runnable runControlLoop=new Runnable(){
    public void run(){
        while(isRunControlLoop){

            /*if(!(driveSlider_.isPressed())){
                driveSlider_.setProgress(50);
            }
            */

            if(!toggleTilt.isChecked()){

currSteer=mySurfaceView.steerval(mySurfaceView.getCurrRotation());

                if(!(prevSteer==currSteer)){
                    Log.d("transmitS","true");
                    if(currSteer<10)
                        sprogStr="0"+Integer.toString(currSteer);
                    else
                        sprogStr= Integer.toString(currSteer);
                    message= "s"+sprogStr;
                    transmit();
                    /*try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                    transmit();*/

                }
            else{
                /*try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                */

            }

            prevSteer=currSteer;

```

```

        //Log.d(TAG,message);
        //Log.d(TAG,"d"+dprogStr);
    }
    message = "s"+sprogStr;
    transmit();

    message = "d"+dprogStr;
    transmit();

    }
}

};
Thread controlLoop=new Thread(runControlLoop);
controlLoop.start();

//mVideoView.setVideoURI(Uri.parse("rtsp://v4.cache5.c.youtube.com/CjYLENy73wIa
LQkBIQn2OQt0IBMYDSANFEIJbXYtZ29vZ2xlSARSBXdhdGNoYPPshPKHtrCAUQw=/0/0/
0/video.3gp"));

}

public class MyTask extends AsyncTask<Void, Void, Void> {
    private ProgressDialog progress;

    public MyTask(ProgressDialog progress) {
        this.progress = progress;
    }

    public void onPreExecute() {
        progress.show();
    }

    public void doInBackground() {
        //... do your loading here ...
    }

    public void onPostExecute() {
        progress.dismiss();
    }

    @Override
    protected Void doInBackground(Void... params) {
        WifiAP controllerAP = new WifiAP(getApplicationContext());

```



```

        if(!controllerAP.isWifiApActive()) // if no ap then make one
            controllerAP.createWifiAccessPoint();

        /* Gets the ip for the car device by wanting for a packet from it. */
        if(carConn == null){
            try {
                ServerSocket socket = new
ServerSocket(SERVERPORT); // yes it's tcp but that doesn't matter
                //socket.setSoTimeout(100);
                // waits until a client connects to the server port.
                carConn = socket.accept().getInetAddress();

                Log.v(TAG, "Got connection from " +
carConn.getHostAddress());

                socket.close();
                onPostExecute();

            } catch (IOException e) {
                Log.e(TAG, e.toString());
                e.printStackTrace();
            }
        }

        /*mVideoView.setVideoURI(Uri.parse("rtsp://" + carConn.getHostAddress() + ":8086"));
        mVideoView.requestFocus();
        mVideoView.start();
        */
        onPostExecute();
        return null;
    }

    public void turnOnSensors(){
        sensorsOn=true;
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        // add listener. The listener will be HelloAndroid (this) class
        sensorManager.registerListener(this,

        sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_GAME);
    }

```

```

/**
 * UDP Runnable connects to the host(receiver) and sends control commands
 */
public void transmit() {
try {
    // Retrieve the ServerName
    //if(carConn == null) // just in case
    //    throw new NullPointerException("Client not set.");

    Log.d(TAG, "C: Connecting... " + carConn.getHostAddress());

    /* Create new UDP-Socket */
    DatagramSocket socket = new DatagramSocket();

    /* Prepare some data to be sent. */
    byte[] buf = (message).getBytes();

    /* Create UDP-packet with
     * data & destination(url+port) */
    DatagramPacket packet = new DatagramPacket(buf, buf.length, carConn,
SERVERPORT);
    Log.d(TAG, "C: Sending: '" + message + "'");

    /* Send out the packet */
    socket.send(packet);

} catch (Exception e) {
    Log.e(TAG, e.toString());
}
}

/**
 * Check to see if the user is moving sliders. If so, run the UDP Runnable
 */
@Override
public void onProgressChanged(SeekBar seekBar, int progress,
    boolean fromUser) {
    if(toggleDemo.isChecked()){
        Log.d(TAG,"Demo");
        double adjustedProg;
        if (progress>50){
            adjustedProg=50+(float)(progress-50)*0.2;
            dprogStr=Integer.toString((int)adjustedProg);
        }
        else if (progress<50){
            adjustedProg=50-(float)(50-progress)*0.2;

```

```

        dprogStr=Integer.toString((int)adjustedProg);
    }
    else
        dprogStr=Integer.toString(progress);
}
else{

    Log.d(TAG,"Not Demo");
    if(progress<10){
        dprogStr="0"+Integer.toString(progress);
    }
    else if(progress==100)
        dprogStr="99";
    else
        dprogStr=Integer.toString(progress);
}

    message= "d"+dprogStr; //Sends command in the form of (sliderID,value)
i.e s30=steer to 30 pos
    Log.d(TAG,message);
    transmit();

}

@Override
public void onStartTrackingTouch(SeekBar seekBar) {
}

@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    driveSlider_.setProgress(50);
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}

@Override
@SuppressWarnings("unused")
@Override
public void onSensorChanged(SensorEvent event) {

```

```

        if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){

            // assign directions
            float x=event.values[0];
            float y=event.values[1];
            float z=event.values[2];
            int progress;

            if(x>8)
                x=8;
            if(x<-8)
                x=-8;
            Log.d("tilt",Integer.toString((int)x));
            currSteer= (int) (100-((x+8)*6));

            if(currSteer<10)
                sprogStr="0"+Integer.toString(currSteer);
            else if(currSteer>=100)
                sprogStr="99";
            else
                sprogStr= Integer.toString(currSteer);

            message= "s"+sprogStr;
            Log.d("tilt",message);
            transmit();

            /*progress=driveSlider_.getProgress();
            if(progress<10){
                dprogStr="0"+Integer.toString(progress);
            }
            else
                dprogStr=Integer.toString(progress);*/
            message = "d"+dprogStr;
            transmit();

        }
    }

    @Override
    public void onCheckedChanged(CompoundButton arg0, boolean arg1) {

        if(arg0.getId()==R.id.toggleTilt){
            if(arg1==true)
                turnOnSensors();
            else{

```

```

        sensorManager.unregisterListener(this);
        sensorsOn=false;
    }
}
if(arg0.getId()==R.id.toggleDemo){
    if(arg1=true){
        demoMode=true;
        driveSlider_.setProgress(50);
    }

    else{
        demoMode=false;
        driveSlider_.setProgress(50);
        driveSlider_.setMax(100);
        Log.d("demoMode","false");
    }

}

}

@Override
protected void onPause() {
    super.onPause();
    mySurfaceView.pause();
    toggleTilt.setChecked(false);
    sensorsOn=false;
    isRunControlLoop = false;
}

@Override
protected void onResume() {
    super.onResume();
    mySurfaceView.resume();
    isRunControlLoop = true;
}

@Override
//gathers the x and y coordinates when the surface is touched.
public boolean onTouch(View v, MotionEvent event) {

    x=event.getX();
    y=event.getY();

    //When user touch the screen, resume the thread

```

```

        if(event.getAction() == MotionEvent.ACTION_DOWN)
            mySurfaceView.resume();

        //When user stops touching the screen, pause the thread
        else if(event.getAction() == MotionEvent.ACTION_UP)
            mySurfaceView.pause();

    return true; //keep checking variables
}

public class SteeringSurface extends SurfaceView implements Runnable{
    private SurfaceHolder myHolder;
    private Thread myThread = null;
    public boolean isRunning=false;

    private float currRotation = 0;
    private Context mContext;
    private int progress = 0;
    private int cnt;

    //SteeringSurface constructor
    public SteeringSurface(Context context){
        super(context);
        mContext = context;
        myHolder = getHolder();
        cnt=0;
    }

    //calculate angle at the point where user pressed
    public float turnAngle(){
        float xOrigin = parent.getWidth()/2;
        float yOrigin = parent.getHeight()/2;

        float deltaY = y-yOrigin;
        float deltaX = x-xOrigin;

        float degree = (float) -Math.toDegrees(Math.atan(deltaY/deltaX))-90;

        //Log.v("drive angle", "Original degree: " + degree);

        if(x > xOrigin && y < yOrigin) //quadrant 1
            degree = -degree;
        else if(x < xOrigin && y < yOrigin) //quadrant 2
            degree = 180 - degree;
    }
}

```

```

        else if(x < xOrigin && y > yOrigin) //quadrant 3
            degree = 180 - degree;
        else if(x > xOrigin && y > yOrigin) //quadrant 4
            degree = -degree;
        else
            degree = degree;

        //Log.v("drive angle", "degree: " + degree + " x: " + x + " y: " + y);

        return degree;
    }

    //turn steering wheel
    public void imageManipulate(float rotDegree, Canvas canvas){

        Matrix transform = new Matrix();

        transform.postRotate(rotDegree);

        //get bitmap
        BitmapScaler scaler = new
        BitmapScaler(getResources(),R.drawable.steering_wheel6,
            parent.getWidth());

        //scale bitmap to desired size
        Bitmap transformed = Bitmap.createBitmap(scaler.getBitmap(), 0, 0,
            parent.getWidth(), parent.getHeight(), transform, true);

        float centerX= parent.getWidth()/2 - transformed.getWidth()/2;
        float centerY= parent.getHeight()/2 - transformed.getHeight()/2;
        canvas.drawBitmap(transformed,centerX,centerY,null); //center picture to
        cursor

        currRotation = rotDegree;
    }

    //Calculate progress value (0-100) of slider when we have the angle
    public int steerval(float degree){

        if(degree >= 0 && degree <= 133)
            return (int) (50 + (degree/2.7)); //progress value 50-100 turns
        right

        else if (degree >= 225 && degree <= 359)

```

```

        return (int) (50 - ((360-degree)/2.7)); //progress value 0-49 turns
left
    else
        return 50; //default to middle of slider
    }

    //getter function to get progress value
    public int getProgress(){
        return progress;
    }

    public void display(float angle){
        Canvas canvas = myHolder.lockCanvas();

        //clear canvas or else multiple steering wheel images appear
        canvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR);
        imageManipulate(angle, canvas);
        progress = steerval(angle); //calculate progress

        myHolder.unlockCanvasAndPost(canvas);
    }

    public float prevangle=0;
    public boolean steeringchanged=false;

    public void run(){
        float angle = 0;

        while(isRunning){
            if(!myHolder.getSurface().isValid()) //if surface is not valid, loop
                continue;
            angle = turnAngle();

            //let the wheel steer between 0-135 and 225-360
            if(x!=0 && y!=0 && (angle <= 135 || angle >= 225))
                display(angle);
        }
        if(!isRunning){ // moves wheel back to starting position if screen not
touched

            display(0); // reset to the origin
        }

    } //end of run()

```



```

        public float getCurrRotation(){
            if(isRunning)
                return currRotation;
            else
                return 0;
        }

        public void pause(){
            isRunning=false;
            sprogStr="50";
            while(true){
                try{
                    myThread.join(); //wait for threads
                }
                catch(Exception e){
                }
                break;
            }
            myThread=null;
        }
        public void resume(){
            isRunning=true;
            myThread=new Thread(this);
            myThread.start();
        }
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        //if (mVideoView != null)
            //mVideoView.setVideoLayout(VideoView.VIDEO_LAYOUT_SCALE,
0);
        super.onConfigurationChanged(newConfig);
    }

    @Override
    public void onClick(View v) {
        if(v.getId()==R.id.buttonBrake){
            driveSlider_.setProgress(50);
        }
    }
}

```

CarActivity.java

```
package com.seniordesign.rccontroller;

import ioio.lib.api.DigitalOutput;
import ioio.lib.api.PwmOutput;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.BaseIOIOLooper;
import ioio.lib.util.IOIOLooper;
import ioio.lib.util.android.IOIOActivity;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketAddress;
import java.net.UnknownHostException;

import net.majorkernelpanic.networking.RtspServer;
import net.majorkernelpanic.networking.Session;
import net.majorkernelpanic.streaming.video.VideoQuality;

import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.WindowManager;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.ToggleButton;
import com.seniordesign.rccontroller.*;

/**Receiver Activity (This is for the device mounted on the RC car)*/

public class CarActivity extends IOIOActivity implements Runnable{

    private static final String TAG = "CarActivity";

    private ToggleButton button_;
    private SeekBar driveSlider_;
    private SeekBar steerSlider_;
    private TextView drivePwmValTxt;
```

```

private TextView steerPwmValTxt;
private TextView testString;

private int drivePwmVal;
private int steerPwmVal;

public int dSliderVal=50;
public int sSliderVal=50;

public InetAddress controlConn = null;
public static final int SERVERPORT = 4444;

// rtsp stuff
static private RtspServer rtspServer = null;
private final int defaultRtspPort = 8086;
public static int videoEncoder = Session.VIDEO_H263;
public static Exception lastCaughtException;
public static VideoQuality videoQuality = new VideoQuality(640,480,15,500000);

/**
 * Called when the activity is first created. Here we normally initialize
 * our GUI.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    setContentView(R.layout.activity_car);

    Bundle extras = getIntent().getExtras();
    if(extras != null) {
        try {
            controlConn =
InetAddress.getByName(extras.getString("hostIp"));
        } catch (UnknownHostException e) {
            Log.e(TAG, e.toString());
            e.printStackTrace();
        }
    }
    button_ = (ToggleButton) findViewById(R.id.button);

    driveSlider_ = (SeekBar) findViewById(R.id.driveSlider);
    driveSlider_.setMax(100);

```

```

driveSlider_.setProgress(50);

steerSlider_ = (SeekBar) findViewById(R.id.steerSlider);
steerSlider_.setMax(100);
steerSlider_.setProgress(50);

drivePwmValTxt = (TextView) findViewById(R.id.pwmValTxt);
steerPwmValTxt = (TextView) findViewById(R.id.pwmValTxtS);
testString = (TextView) findViewById(R.id.textView123);

if(controlConn == null)
    try {
        controlConn = InetAddress.getByName(getLocalIP());

        } catch (IOException e) {
            Log.e(TAG, e.toString());
            e.printStackTrace();
        } catch (Exception e) {
            Log.e(TAG, e.toString());
            e.printStackTrace();
        }
    }

/* Connects tries to connect to the controller. So the controller can get the car's ip.
*/
    try {
        SocketAddress address = new
InetSocketAddress(InetAddress.getByName(getContrIP()), SERVERPORT);

        Socket controller = new Socket();

        controller.connect(address);

        Log.v(TAG, "Connected to controller at " +
controlConn.getHostAddress());

        controller.close();
    } catch (IOException e) {
        Log.e(TAG,e.toString());
        e.printStackTrace();
    }
}

//while(true)
//run();

/*Session.setHandler(handler);

```

```

        Session.setDefaultVideoEncoder(videoEncoder);
        Session.setDefaultVideoQuality(videoQuality);

        rtspServer = new RtspServer(defaultRtspPort, handler);

        try {
            rtspServer.start();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }*/

    }

    /**
     * Uses the local ip address to find the controller ip. We know that they're on the same
    network,
     * and that the controller is the access point. Adapted from stallion's code.
     * @return A string in the form (192.168.1.1). Subnet will differ.
     * @see http://stackoverflow.com/a/5308046
     */
    private String getContrIP(){
        WifiManager wifiManager = (WifiManager)
getSystemService(WIFI_SERVICE);
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        int ip = wifiInfo.getIpAddress();

        // we need to know the network prefix and subnet are the same and that the host
    number is 1.
        String contrIP = (ip & 0xFF) + "." + ((ip >> 8) & 0xFF) + "." + ((ip >> 16) &
0xFF) + ".1";

        Log.d(TAG,"Controller IP: " + contrIP);

        return contrIP;
    }

    private String getLocalIP(){
        WifiManager wifiManager = (WifiManager)
getSystemService(WIFI_SERVICE);
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        int ip = wifiInfo.getIpAddress();

        // we need to know the network prefix and subnet are the same and that the host
    number is 1.

```

```
String contrIP = (ip & 0xFF) + "." + ((ip >> 8) & 0xFF) + "." + ((ip >> 16) &
0xFF) + "." + ((ip >> 24) & 0xFF);
```

```
Log.d(TAG, "Controller IP: " + contrIP);
```

```
return contrIP;
```

```
}
```

```
/**This is the UDP Runnable: Establishes the Receiver as the host and waits for the
 * Controller to send commands. Contains safeties to prevent the RC car from going
crazy
```

```
 */
```

```
@Override
```

```
public void run() {
```

```
    DatagramSocket socket = null;
```

```
    try {
```

```
        String incoming;
```

```
        String contID;
```

```
        String contVal;
```

```
        Log.d("UDP", "S: Connecting... " + controlConn.getHostAddress());
```

```
        /* Create new UDP-Socket */
```

```
        socket = new DatagramSocket(SERVERPORT, controlConn);
```

```
        /* By magic we know, how much data will be waiting for us */
```

```
        byte[] buf = new byte[17];
```

```
        /* Prepare a UDP-Packet that can
```

```
        * contain the data we want to receive */
```

```
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
```

```
        Log.d("UDP", "S: Receiving...");
```

```
        /* Receive the UDP-Packet */
```

```
        socket.setSoTimeout(1000);
```

```
        socket.receive(packet);
```

```
        incoming = new String(packet.getData());
```

```
        contVal= incoming.substring(1,3);
```

```
        contID=incoming.substring(0,1);
```

```
        if(contID.equals("d")){
```

```
            dSliderVal=Integer.valueOf(contVal);
```

```
            //sSliderVal=50;
```

```
        }
```

```
        else if(contID.equals("s")){
```

```
            sSliderVal=Integer.valueOf(contVal);
```

```
            //dSliderVal=50;
```

```

    }
    else{
        //sSliderVal=dSliderVal=50;
    }

    Log.d("UDP", "S: Received: " + contVal+contID + "");
    Log.d("UDP", "S: Done.");

    socket.close();
} catch (Exception e) {
    Log.e("UDP", "S: Error", e);
    //Stops the RC car in case of error
    sSliderVal=dSliderVal=50;
    socket.close();
}

}

/**
 * This is the thread on which all the IOIO activity happens. It will be run
 * every time the application is resumed and aborted when it is paused. The
 * method setup() will be called right after a connection with the IOIO has
 * been established (which might happen several times!). Then, loop() will
 * be called repetitively until the IOIO gets disconnected.
 */
class Looper extends BaseIOIOLooper {
    // The on-board LED.
    private DigitalOutput led_;
    private PwmOutput drivePwm;
    private PwmOutput steerPwm;

    /**
     * Called every time a connection with IOIO has been established.
     * Typically used to open pins.
     *
     * @throws ConnectionLostException
     *         When IOIO connection is lost.
     * @throws InterruptedException
     *
     * @see ioio.lib.util.AbstractIOIOActivity.IOIOThread#setup()
     */
    @Override
    protected void setup() throws ConnectionLostException, InterruptedException {
        led_ = ioio_.openDigitalOutput(0, true);
    }
}

```

```

        drivePwm = ioio_.openPwmOutput(3, 50);
        drivePwm.setPulseWidth(1500);
        steerPwm = ioio_.openPwmOutput(6, 50);
        steerPwm.setPulseWidth(1500);
        enableUi(true);
        //Thread.sleep(1500);
    }

    /**
     * Called repetitively while the IOIO is connected.
     *
     * @throws ConnectionLostException
     *         When IOIO connection is lost.
     *
     * @see ioio.lib.util.AbstractIOIOActivity.IOIOThread#loop()
     */
    @Override
    public void loop() throws ConnectionLostException {
        led_.write(!button_.isChecked());

        run();//Get the slider values from the controller

        drivePwmVal= 1000+(dSliderVal)*10;
        drivePwm.setPulseWidth(drivePwmVal);

        steerPwmVal= 2000-(sSliderVal)*10;
        steerPwm.setPulseWidth(steerPwmVal);

        /*setSlider(false,sSliderVal);
        setSlider(true,dSliderVal);

        drivePwmVal= 1000+(driveSlider_.getProgress())*10;
        drivePwm.setPulseWidth(drivePwmVal);

        steerPwmVal= 2000-(steerSlider_.getProgress())*10;
        steerPwm.setPulseWidth(steerPwmVal);

        setText(Integer.toString(drivePwmVal),Integer.toString(steerPwmVal));

        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
        }*/
    }
}

```



```

/**
 * A method to create our IOIO thread.
 *
 * @see ioio.lib.util.AbstractIOIOActivity#createIOIOThread()
 */
@Override
protected IOIOLooper createIOIOLooper() {
    return new Looper();
}
/**Can't modify the UI inside IOIO Loop so do it here*/
private void enableUi(final boolean enable) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            driveSlider_.setEnabled(enable);
            button_.setEnabled(enable);
        }
    });
}

private void setText(final String str,final String str2) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            drivePwmValTxt.setText(str);
            steerPwmValTxt.setText(str2);
        }
    });
}

private void setSlider(final boolean drive,final int val) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            if(drive==true){
                driveSlider_.setProgress(val);
            }
            else
                steerSlider_.setProgress(val);
        }
    });
}

private boolean streaming = false;

```

```

// The Handler that gets information back from the RtspServer and Session
private final Handler handler = new Handler() {

    public void handleMessage(Message msg) {
        switch (msg.what) {
            case RtspServer.MESSAGE_ERROR:
                Exception e1 = (Exception)msg.obj;
                lastCaughtException = e1;
                //log(e1.getMessage() != null ? e1.getMessage() : "An error occurred !");
                break;
            case RtspServer.MESSAGE_LOG:
                //log((String)msg.obj);
                break;
            /*case HttpServer.MESSAGE_ERROR:
                Exception e2 = (Exception)msg.obj;
                lastCaughtException = e2;
                break;*/
            case Session.MESSAGE_START:
                streaming = true;
                //streamingState(1);
                break;
            case Session.MESSAGE_STOP:
                streaming = false;
                //displayIpAddress();
                break;
        }
    }

};
}

```

Slider.java

```

package com.seniordesign.rccontroller;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
import android.widget.SeekBar;

```

```

public class Slider extends SeekBar {

    public Slider(Context context) {

        super(context);
    }

    public Slider(Context context, AttributeSet attrs, int defStyle) {

        super(context, attrs, defStyle);
    }

    public Slider(Context context, AttributeSet attrs) {

        super(context, attrs);
    }

    protected void onSizeChanged(int w, int h, int oldw, int oldh) {

        super.onSizeChanged(h, w, oldh, oldw);
    }

    @Override

    protected synchronized void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

        super.onMeasure(heightMeasureSpec, widthMeasureSpec);
        setMeasuredDimension(getMeasuredHeight(), getMeasuredWidth());
    }

    protected void onDraw(Canvas c) {

        c.rotate(-90);
        c.translate(-getHeight(), 0);

        super.onDraw(c);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {

        if (!isEnabled()) {

```

```

        return false;
    }

    switch (event.getAction()) {

        case MotionEvent.ACTION_DOWN:

        case MotionEvent.ACTION_MOVE:

        case MotionEvent.ACTION_UP:

            setProgress(getMax() - (int) (getMax() * event.getY() / getHeight()));
            onSizeChanged(getWidth(), getHeight(), 0, 0);
            break;

        case MotionEvent.ACTION_CANCEL:
            break;

    }

    return true;
}

}

```

WifiAP.java

```

package com.seniordesign.rccontroller;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

import android.content.Context;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiManager;
import android.util.Log;

import com.gmail.radikll.pro.reflect.Extract;

/**
 * This class uses reflection to use undocumented methods from the Android API.
 * These methods allow us to create an access point programmatically.
 *
 * @author Jon Green
 */

```

```

public class WifiAP {

    static final private String TAG = "WifiAP-class";

    // access point states
    public static final int WIFI_AP_STATE_UNKNOWN      = -1;
    public static final int WIFI_AP_STATE_DISABLING   = 0;
    public static final int WIFI_AP_STATE_DISABLED     = 1;
    public static final int WIFI_AP_STATE_ENABLING    = 2;
    public static final int WIFI_AP_STATE_ENABLED     = 3;
    public static final int WIFI_AP_STATE_FAILED      = 4;

    public static final int WIFI_NETWORK_OPEN          = 0;
    public static final int WIFI_NETWORK_WEP           = 1;
    public static final int WIFI_NETWORK_WPA           = 2;

    private WifiManager wifiManager;
    private Context sContext;

    //TODO these objects should be able to be set in a constructor
    private String sSsid = "AccessPoint";
    private String sKey = "clusteroffrogs";
    private int sNetType = WIFI_NETWORK_OPEN;
    private boolean sIshidden = false;

    public WifiAP(Context context){

        this.sContext = context;

        wifiManager = (WifiManager) sContext.getSystemService(Context.WIFI_SERVICE);

    }

    /**
     * This method will create the access point.
     */
    public void createWifiAccessPoint() {

        if(wifiManager.isWifiEnabled())
            wifiManager.setWifiEnabled(false);

        WifiConfiguration netConfig = new WifiConfiguration();
        netConfig.SSID = sSsid;
        netConfig.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN);
        netConfig.hiddenSSID = sIshidden;
    }
}

```

```

        if(setWifiApActive(netConfig, true))
            Log.d(TAG, "Access Point created");
        else
            Log.d(TAG, "Access Point creation failed");

        while(!(Boolean)isWifiApActive()){ };
    }

    // TODO create a destoryWifiAccessPoint method also consider making this a static class

    /* TODO probably not possible to make a connect method
    *
    * helpful link :p
    * https://android.googlesource.com/platform/frameworks/base/+/-/jb-release/wifi/java/android/net/wifi/WifiManager.java
    */

    public void connect(){

    }
    */

    /**
    * Checks if the wifi access point is exists.
    * @return true if access point is active and false if not.
    */
    public boolean isWifiApActive(){

        Method isWifiApEnabled = Extract.getMethod(wifiManager.getClass(),
"isWifiApEnabled");

        if(isWifiApEnabled.equals(null))
            return false;

        boolean isEnabled = false;

        try {
            isEnabled = (Boolean)isWifiApEnabled.invoke(wifiManager);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }

        return isEnabled;
    }

```

```

    }

    /**
     * Allows you to enable or disable (turn on/off) the wifi access point on your device.
     * @param config is the wifi configuration. Check the android documentation for more
info.
     * @param value set true to activate and false to deactivate.
     * @return true if active and false if inactive .
     */
    public boolean setWifiApActive(WifiConfiguration config, boolean value){

        Method setWifiApEnabled = Extract.getMethod(wifiManager.getClass(),
"setWifiApEnabled");

        if(setWifiApEnabled.equals(null))
            return false;

        boolean isSet = false;

        try {
            isSet = (Boolean) setWifiApEnabled.invoke(wifiManager, config, true);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }

        return isSet;
    }

    /**
     * Gets the current state of the access point.
     * @return A integer -1 through 4. See list of access point states at the top of the file.
     */
    public int getWifiApState(){

        Method getWifiApState = Extract.getMethod(wifiManager.getClass(),
"getWifiApState");

        if(getWifiApState.equals(null))
            return WIFI_AP_STATE_UNKNOWN;

        int apstate = WIFI_AP_STATE_UNKNOWN;

```

```

        try {
            apstate = (Integer)getWifiApState.invoke(wifiManager);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }

        return apstate;
    }

    public void setSSID(String ssid){
        sSsid = ssid;
    }

    public void setKey(String key){
        sKey = key;
    }

    public void setNetworkType(int type){
        sNetType = type;
    }

    public void setHidden(boolean hide){
        sIshidden = hide;
    }
}

```