<table>
<thead>
<tr>
<th>Title</th>
<th>Android camera car - Sengital Ltd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Lui, Tik On (雷廸安)</td>
</tr>
<tr>
<td>Citation</td>
<td>Lui, T. O. (2012). Android camera car - Sengital Ltd. (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2012</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2031/6740">http://hdl.handle.net/2031/6740</a></td>
</tr>
<tr>
<td>Rights</td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

Project Title

Android Camera Car – Sengital Ltd.

Student Name: Lui Tik On
Student ID: 
Supervisor: Dr. Ricky W.H. Lau
Assessor: Dr. Ray C.C. Cheung

Bachelor of Engineering (Honours) in Computer Engineering
Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

Project Title: Android Camera Car – Sengital Ltd.

Student Name: Lui Tik On

Student ID:

Signature

Date: 12th April, 2012
No part of this report may be reproduced, stored in a retrieval system, or transcribed in any form or by any means – electronic, mechanical, photocopying, recording or otherwise – without the prior written permission of City University of Hong Kong.
# Table of Contents

Student Final Year Project Declaration ................................................................. i  
Table of Contents .................................................................................................. iii  
List of Figures ......................................................................................................... v  
List of Tables ......................................................................................................... ix  
Project Abstract .................................................................................................... x  
Chapter 1.............................................................................................................. 1  
  1.1 Objective of the project .................................................................................. 1  
  1.2 Outlines of the report ................................................................................... 3  
Chapter 2 .............................................................................................................. 4  
  2.1 Hardware Part ............................................................................................... 4  
  2.1.1 REAL6410 .............................................................................................. 4  
  2.1.1.1 Core Board - E6COREV3 ................................................................... 4  
  2.1.1.2 Arm Controlled Board for E6COREV3 .............................................. 6  
  2.1.2 Core Board - E210COREV2 ................................................................... 8  
  2.1.3 LPC1768 ................................................................................................. 11  
  2.1.4 Motor Driver for Car Motors ................................................................... 12  
  2.1.5 Soldering ................................................................................................. 14  
  2.1.6 Fabrication .............................................................................................. 15  
  2.2 Software Part ............................................................................................... 15  
  2.2.1 Camera Driver ......................................................................................... 15  
  2.2.2 UART communication .......................................................................... 16  
  2.2.3 PWM command in LPC1768 for controlling the motor driver ............. 17  
  2.2.4 Android application on Arm Controlled Board and Android mobile device .......................................................... 18  
  2.2.4.1 Android's Background ....................................................................... 18  
  2.2.4.2 Android's Platform ............................................................................. 19  
  2.2.4.3 Android SDK .................................................................................... 20  
Chapter 3 .............................................................................................................. 21  
  3.1 Overview of the system .............................................................................. 21  
  3.2 Connection between Android Camera Car (Arm Controlled Board) and Client's Android Mobile Devices .......................................................... 23  
  3.3 Sending command from Client's Android Mobile Devices to Android Camera Car (Arm Controlled Board) .......................................................... 25  
  3.4 Reading input command from Client's Android Mobile Devices in Android Camera Car (Arm Controlled Board) .................................................... 26  
  3.5 Sending Video packets from Android Camera Car (Arm Controlled Board) to Client's Android Mobile Devices ...................................................... 26  
  3.6 Receiving Video packets from Android Camera Car (Arm Controlled Board) in Client's Android Mobile Devices .............................................. 27  
  3.7 Sending command message from Android Camera Car (Arm Controlled Board) to LPC1768 through UART28 .................................................. 28  
  3.8 Receiving command message from Android Camera Car (Arm Controlled Board) in LPC1768 through UART28 .................................................. 28  
  3.9 Set the PWM to control the Car Motor ...................................................... 29
3.10 Combine all the functions into one application in Android Camera Car (Arm Controlled Board) ................................................. 31
3.11 Combine all of the functions into one application in Client's Android Platform Mobile Devices and video streaming improvement ............................................................ 33

Chapter 4 .................................................................................................................. 36
4.1 Hardware Part ...................................................................................................... 36
4.1.1 Designing a specified functional REAL6410 ..................................................... 36
4.1.2 Routing the PCB layout of core board E21OCOREV2 ......................................... 45
4.1.3 Designing a specified functional of Arm Controlled Board based on E21OCOREV2 ................................................................. 48
4.2 Software Part ....................................................................................................... 58
4.2.1 Android Application of Arm Controlled Board ...................................................... 58
4.2.1.1 Structure of the program ............................................................................. 58
4.2.1.2 Program flow of the system connection and operation .................................... 59
4.2.2 Android Application in Android Mobile Device ................................................. 63
4.2.2.1 Structure of the program ............................................................................. 63
4.2.2.2 Program flow of the system connection and operation .................................... 65
4.2.3 Program in LPC1768 for controlling the movement of motors ......................... 68
4.2.3.1 Structure of the program ............................................................................. 68
4.2.3.2 Program flow of the system connection and operation .................................... 69
4.2.4 Modifying the camera driver ........................................................................... 71

Chapter 5 .................................................................................................................. 73
5.1 Operation procedure ............................................................................................. 73
5.2 Result ...................................................................................................................... 74
5.2.1 Hardware Part .................................................................................................. 74
5.2.2 Software Part .................................................................................................. 78

Chapter 6 .................................................................................................................. 80
6.1 Hardware Part ....................................................................................................... 80
6.1.1 Studying the datasheets of the different ICs for designing the Arm Controlled Board ................................................................. 80
6.1.2 Drawing the schematic and PCB footprint into the library ................................ 80
6.1.3 Soldering on the Arm Controlled Board ............................................................. 82
6.2 Software Part ....................................................................................................... 83
6.2.1 Integrating the Android Application in both Arm Controlled Board and client's Android mobile device ......................................................... 83
6.2.2 Modifying the camera driver ........................................................................... 84

Chapter 7 .................................................................................................................. 85
7.1 Designing Arm Controlled Board for Contex - A8 in version 2 ........................ 85
7.2 Adding GPS Navigation ....................................................................................... 85
7.3 Adding Robot Arm ............................................................................................... 86

Chapter 8 .................................................................................................................. 87
Reference .................................................................................................................... 88
Appendix ..................................................................................................................... 89
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The schematic of core board E6COREV3</td>
<td>P.5</td>
</tr>
<tr>
<td>2.2</td>
<td>The Chip of E6COREV3 in real view</td>
<td>P.5</td>
</tr>
<tr>
<td>2.3</td>
<td>The Arm Controlled Board with Touch Screen in real view</td>
<td>P.7</td>
</tr>
<tr>
<td>2.4</td>
<td>The schematic of core board E210COREV2</td>
<td>P.9</td>
</tr>
<tr>
<td>2.5</td>
<td>The chip of E210COREV2 in real view</td>
<td>P.9</td>
</tr>
<tr>
<td>2.6</td>
<td>LPC1768 in real view</td>
<td>P.12</td>
</tr>
<tr>
<td>2.7</td>
<td>HIP4020 in real view</td>
<td>P.13</td>
</tr>
<tr>
<td>2.8</td>
<td>Pin out of HIP4020</td>
<td>P.13</td>
</tr>
<tr>
<td>2.9</td>
<td>Block Diagram of HIP4020</td>
<td>P.14</td>
</tr>
<tr>
<td>2.10</td>
<td>Process of soldering</td>
<td>P.14</td>
</tr>
<tr>
<td>2.11</td>
<td>Sample of circuit board after processing fabrication</td>
<td>P.15</td>
</tr>
<tr>
<td>2.12</td>
<td>Files related to camera driver</td>
<td>P.16</td>
</tr>
<tr>
<td>2.13</td>
<td>Characters Framing of UART</td>
<td>P.17</td>
</tr>
<tr>
<td>2.14</td>
<td>Icon of Android</td>
<td>P.18</td>
</tr>
<tr>
<td>2.15</td>
<td>System architecture of Android’s Platform</td>
<td>P.20</td>
</tr>
<tr>
<td>3.1</td>
<td>System connections</td>
<td>P.22</td>
</tr>
<tr>
<td>3.2</td>
<td>DatagramSocket Request Connections</td>
<td>P.24</td>
</tr>
<tr>
<td>3.3</td>
<td>DatagramSocket with connection successfully</td>
<td>P.25</td>
</tr>
<tr>
<td>3.4</td>
<td>Typical motor control application circuit showing directional and braking</td>
<td>P.30</td>
</tr>
<tr>
<td>3.5</td>
<td>Truth Table of HIP4020</td>
<td>P.30</td>
</tr>
<tr>
<td>3.6</td>
<td>Pin Description of HIP4020</td>
<td>P.31</td>
</tr>
<tr>
<td>3.8</td>
<td>The function of the combined application in the Arm Controlled Board</td>
<td>P.33</td>
</tr>
</tbody>
</table>
Figure 3.8  The function of the combined application in the client’s Android mobile devices…………………………………………………………………………………………………..P.35

Figure 4.1  Design Explorer DXP……………………………………………………...P.36
Figure 4.2  Welcoming screen of DXP…………………………………………….P.37
Figure 4.3  Schematic of Camera Port……………………………………………P.37
Figure 4.4  Schematic of Wi-Fi port………………………………………………P.38
Figure 4.5  Schematic of 3G port……………………………………………………P.38
Figure 4.6  Schematic of RS232 and UART port…………………………………P.39
Figure 4.7  Schematic of LCD Display port………………………………………P.40
Figure 4.8  Schematic of Keypads………………………………………………..P.40
Figure 4.9  Schematic of SD Card port…………………………………………….P.41
Figure 4.10 Schematic of Mini-USB port…………………………………………P.41
Figure 4.11 Schematic of Power port……………………………………………..P.42
Figure 4.12 Schematic of 4 Adapter headers……………………………………P.43
Figure 4.13 Schematic of LM232 in Schematic Library………………………..P.43
Figure 4.14 Schematic of 4 adapter header in Schematic Library………………P.44
Figure 4.15 Schematic of specified function of REAL6410……………………..P.45
Figure 4.16 Footprint of E210COREV2 in Footprint Library…………………P.46
Figure 4.17 Footprint of 4 adapter headers in Footprint Library………………P.46
Figure 4.18 PCB Layout of E210COREV2 Core Board………………………..P.47
Figure 4.19 Schematic of Camera port with E210COREV2…………………..P.48
Figure 4.20 Wi-Fi modules in Schematic Library………………………………P.49
Figure 4.21 Schematic of Wi-Fi module with E210COREV2…………………..P.49
Figure 4.22 Schematic of 3G port with E210COREV2………………………..P.50
Figure 4.23  Schematic of RS232 port with E210COREV2................................................................................P.51
Figure 4.24  Schematic of UART port with E210COREV2................................................................................P.51
Figure 4.25  Schematic of Touch Screen Header in Schematic Library.................................................................P.52
Figure 4.26  Schematic of LCD Display port with E210COREV2........................................................................P.52
Figure 4.27  Schematic of Keypads with E210COREV2.........................................................................................P.53
Figure 4.28  Schematic of SD Card port with E210COREV2................................................................................P.53
Figure 4.29  Schematic of Mini-USB in Schematic Library.....................................................................................P.54
Figure 4.30  Schematic of MINI-USB with E210COREV2.....................................................................................P.54
Figure 4.31  Schematic of Power port with E210COREV2....................................................................................P.55
Figure 4.32  Schematic of Adapter Header with E210COREV2...........................................................................P.56
Figure 4.33  Schematic of specified function Arm Controlled Board with E210COREV2........................................P.57
Figure 4.34  Structure of the program in Arm Controlled Board...........................................................................P.59
Figure 4.35  Flow Diagram of program in Arm Controlled Board........................................................................P.62
Figure 4.36  Structure of the program in client’s Android Mobile Device..............................................................P.64
Figure 4.37  Flow Diagram of program in client’s Android Mobile Device............................................................P.67
Figure 4.38  Structure of the program for LPC1768............................................................................................P.69
Figure 4.39  Program flow of the program in LPC1768......................................................................................P.70
Figure 4.40  Mirror and flip control of OV3640....................................................................................................P.71
Figure 4.41  System Control Register of OV3640...............................................................................................P.72
Figure 5.1  Front view of Arm Controlled Board with core board E6COREV3.....................................................P.74
Figure 5.2  Back view of Arm Controlled Board with core board E6COREV3.......................................................P.74
Figure 5.3  Front view of Specified function Arm Controlled Board for E210COREV2........................................P.75
Figure 5.4 Back view of Specified function Arm Controlled Board for E210COREV2.................................................................P.75

Figure 5.5 Front view of Adaptor Board of E210COREV2.............................................P.76

Figure 5.6 Back view of Adaptor Board of E210COREV2.............................................P.76

Figure 5.7 Android Camera Car.................................................................P.77

Figure 5.8 The 1st design of the application layout........................................P.78

Figure 5.9 The capture of the application after connection setup in Android Mobile Device.................................................................P.78

Figure 5.10 The capture of the application before connection setup in Arm Controlled Board.................................................................P.79

Figure 6.1 One example of the hardware design details from the datasheet...........P.81

Figure 6.2 Structure of the pins of LPC Display header in real view..................P.82
List of Tables

Table 1: Variables used in Uart_cameraActivity{} ..................................................P.89
Table 2: Variables used in FfmpegdecodeActivity{} ..............................................P.90
Table 3: Variables in pwm{} ..................................................................................P.92
Table 4: Methods used in class Uart_cameraActivity{} ..........................................P.92
Table 5: Methods used in FfmpegdecodeActivity{} ..................................................P.93
Table 6: Methods used in pwm{} .............................................................................P.94
Table 7: Command in Android Application.............................................................P.94
Table 8: Command in LPC1768................................................................................P.95
Project Abstract

In this project, the main part is, how can we implement video streaming on every android platform devices and how can we make a specified “tablet” devices that we want? In this project, we want to design a device which is designed on embed Android platform based on Linux Kernel to transmit real time video signals to any Android devices. To make it more interesting, we try to embed this system into a simple motor car and it provides the user playing the car with the first-person view likes a driving a car in the real world. For the performance, the “tablet” device runs smoothly and also video streaming is well performed in the good quality of network or high speed CPU environment. To deal with this issue, we have tried modified the program and its performance is finally improved. Apart from that, the motor car can be controlled smoothly by the user without any problems. This project may be useful for the people who want to study about video conferencing and tablet hardware design since we will explain the rationale and procedure behind of these two parts.
Chapter 1 – Introduction

1.1 Objectives of the project

1.1.1 Hardware part

a) To study designing hardware techniques such as fabrication and soldering
b) To study drawing Schematic Diagram and routing PCB layout for hardware design by using Altium DXP
c) To design a module including 2 motor drivers in order to drive the car movement
d) To design a module including 6 motor drivers and one multiplexer in order to drive the robot arm
e) To study the reference design for the Arm Controlled Board of REAL6410
f) To redesign and make a specified functional device that we want based on the reference
g) To study the architecture of the core board E210COREV2
h) To design an Arm Controlled Board which is based on the core board E210COREV2 and the reference of Arm Controlled Board of REAL6410

1.1.2 Software part

a) To study Camera and Wi-Fi drivers which are provided by the sellers
b) To study the Linux Kernel, Android Operating System and U-Boot which are the open sources from the Internet and reference design (Real 6410)
c) To modify and transplant Wi-Fi and Camera drivers in the Linux Kernel

d) To port the Android Operating System, Linux Kernel and U-Boot to the own designed Arm Controlled Board by the reference REAL6410 and the own designed Arm Controlled Board based on the core board E210COREV2

e) To write and modify C-program for LPC1768 to control the movement of the car’s motors, servo motor and robot arm

f) To write an Android apps for the Arm Controlled Board which is embedded on the car including the following functions:

1. Can get the capture from the camera and real-time stream the video to client’s android mobile device which connected to the car

2. Can record and stream the video simultaneously and the recorded video is stored in the SD card of the EVB

3. Can receive and distinguish the command sent by client and send the command to LPC1768 through UART port of the EVB
To write an Android apps for the Android mobile device including the following functions:

1. Can receive the streamed video and display the image on the screen of Android mobile device in real-time
2. Can send the controlling command to the EVB which is embedded on the car through Wi-Fi or 3G

1.2 Outlines of the report

First of all, we will talk about the background about all the stuffs that related to our project in Chapter 2. Then, the system architecture about the overview of the system and the connections among different components will be discussed in Chapter 3. Next, the program used will be introduced briefly in Chapter 4. Furthermore, the result will be given in Chapter 5. In Chapter 6, there are some discussions about the project such as some technical problem we had met and the solution of the issues. Furthermore, Further Development will be suggested in Chapter 7 and finally conclusion is given in Chapter 8.
Chapter 2 – Background

2.1 Hardware Part

2.1.1 REAL6410

2.1.1.1 Core Board – E6COREV3 [1]

E6COREV3 is smoothly passed through the EMC testing. It is added JTAG interface on the basis of the E6COREV1, the R&D of E6COREV3 is strongly supported by Samsung China, WOLFSON (Shanghai), SanDisk (Shenzhen) and MPS (China) that support component selection and testing. Apart from that, it is worthy to mention that the Top layer and Bottom layer alignment of E6COREV3 is using 8 layers buried blind design. Therefore, it effectively prevents the electromagnetic radiation and electromagnetic interference. It is fully tested in the industrial environments.

The specification of E6COREV3:

1. ARM-11 Kernel
2. CPU Clock Rate 667MHZ
3. 256M Byte mDDR
4. 1GByte MLC NAND FLASH
5. WM9713 (WOLFSON Official Product), support mobile audio design project
6. iNAND (support 16GB SanDisk)
Figure 2.1    The Schematic of the core board E6COREV3

Figure 2.2    The Chip of E6COREV3 in real view
2.1.1.2 Arm Controlled Board for E6COREV3 [2]

It has different ports in one reference EVB to developers for basic testing or developing. Now I am going to introduce that the ports we may use in this project and the ports are:

1. Mini-USB OTG port
2. USB Host 1.1 port
3. Two RS-232 ports
4. SD Card port
5. Two TTL UART ports
6. Camera port
7. SPI port
8. Embedded MIC
9. SDIO WIFI port
10. 24-bit RGB LCD port

Furthermore, it has provided different modules for developers for different specified testing or developing. Now I am going to introduce the module we may use in this project and the modules are:

1. 3G module
2. WIFI module
3. Camera module
4. 7 inch Touch Screen
Figure 2.3  The Arm Controlled Board with Touch Screen
in real view
2.1.2 Core Board – E210COREV2 [3]

Based on the R&D from the previous version of E6COREV3, E210COREV2 is finally designed and produced, it is much more powerful than E6COREV3 which contains the function that E6COREV3 has and it is:

1. The only one which support audio conversation in Arm Controlled Board
2. The only one which has the best power management support

The Specification of E210COREV2:
1. ARM Cortex-A8 Kernel
2. CPU Clock Rate 1GHZ
3. 4G bits DDR2
4. 2G bits SLC NAND FLASH
5. Support NEON command
6. Support MPEG-4/MPEG2, H.264/H263. VC-1. DivX encoding and decoding in 1080p@30fps
7. Support SD/MMC/SDIO (max. support 32GB)
8. Support JPEG hardware encoding and decoding
9. Contain integrated monolithic power management IC (PMIC)
Figure 2.4  The schematic of core board E210COREV2

Figure 2.5  The chip of E210COREV2 in real view
Why do we need use E210COREV2 rather than E6COREV3 is because E6COREV3 does not support video recording at that stage. We did sourcing again and finally we found this E210COREV2. Its architecture is similar to E6COREV3 and that most important part is it supports video recording. Therefore, we use E210COREV2 to be the core board of our final product.
2.1.3 LPC1768 [4]

The NXP LPC1768 which is founded by Philips is an ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100MHz, 512kB on-chip Flash ROM with enhanced Flash Memory Accelerator, In-System Programming (ISP) and In-Application Programming (IAP), 64kB RAM, Nested Vectored Interrupt Controller, Eight channel General purpose DMA controller, AHB Matrix, APB, Ethernet 10/100 MAC with RMII interface and dedicated DMA, USB 2.0 full-speed Device controller and Host/OTG controller with DMA, CAN 2.0B with two channels, Four UARTs, one with full Modem interface, Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interface, General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC, Four 32-bit Timers with capture/compare, Standard PWM Timer block, Motor control PWM for three-phase Motor control, Quadrature Encoder, Watchdog Timer, Real Time Clock with optional Battery backup, System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit, Power-On Reset, Power Management Unit, Wakeup Interrupt Controller, Crystal oscillator, 4MHz internal RC oscillator, PLL, JTAG and Serial Wire Debug/Trace Port with ETM.
2.1.4 Motor Driver for Car Motors

The motor drivers are provided by Sengital Limited. Since Alan Lam, who is the Chief Executive Officer of Sengital Limited, wants our products can be compatible with their products. Therefore we use this module to develop our own motor driver module.

The model of the provided motor IC is HIP4020. Based on the reference design provide from Sengital Limited, we can redesign the modules in a short period of time. The main duty for me is to write a program to control the movement of the car. The motor driver should be controlled by LPC1768 so that I need to write and modify the program of pwm.c.
Figure 2.7  HIP4020 in real view

Figure 2.8  Pin out of HIP4020
2.1.5 Soldering [5]

Soldering is a method that two or more metal objects are linked together by melting; the filler metal has a lower melting point than the work piece. In brazing, the filler metal melts at a very high temperature, but it does not melt the work piece metal. In the past time, almost all of the solders contained lead. However, due to the highly environmental concerns, more factories increasingly use the lead-free materials for electronics and plumbing purposes.
2.1.6 Fabrication [6]

Fabrication is the procedure used to produce the combined circuits which existing in daily electrical and electronic devices. It is a multi-step arrangement of photolithographic and chemical treating stages during which electronic circuits are made on a wafer made of pure semiconducting material. Silicon is nearly always used; however, several compound semiconductors are used for specialized applications.

Figure 2.11 Sample of circuit board after processing fabrication

2.2 Software Part

2.2.1 Camera Driver

Alan Lam, Chief Executive of Sengital Ltd., found that the camera cannot achieve 30fps when he used the camera apps and requested our team to try to enhance the frame rate as much as we can.

After studying the provided camera module, we found that the camera module is designed by OV3640 image sensor. In the reference of Linux Kernel for REAL6410, we found that there are quite a lot of files related to camera driver.
After we do the research for long time, we found that we just only modify the camera.h file, we will demonstrate how do we modify the file.

2.2.2 UART communication [7]

Universal Asynchronous Receiver/Transmitter is a type of "asynchronous receiver/transmitter", computer hardware that translates data between serial and parallel forms. UARTs are usually used in combination with communication standards. The universal designation specifies that the data format and transmission speeds are configurable and that the actual electric signalling levels and methods classically are handled by a distinct driver circuit external to the UART.

A UART is usually a separate (or part of an) integrated circuit used for serial communications over a computer or device serial port. UARTs are currently comprised in microcontrollers.

Communication may be simplex, full duplex or half duplex.
2.2.3 PWM command in LPC1768 for controlling the motor driver [8]

Pulse-width modulation (PWM) is a frequently used technique for monitoring power to inertial electrical devices.

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off in a fast step. The longer the switch is on compared to the off periods, the higher the power supplied to the load is.

The PWM switching frequency has to be much faster than what would affect the load. Typically, switching has to be done several times a minute in an electric stove.

The term duty cycle defines the proportion of 'on' time to the ordered interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time.

The advantage of PWM is that power loss in the switching devices is extremely low. When a switch is off there is no current, and when it is on, there is nearly no voltage drop across the switch. Power loss is thus in both cases tend to zero. PWM also works in good health with digital controls, which can simply set the needed duty cycle.
2.2.4 Android application on Arm Controlled Board and Android mobile devices

2.2.4.1 Android’s Background [9]

Android is a Linux-based operating system for mobile devices. It is developed by the Open Handset Alliance led by Google. Google bought the developer of the software, Android Inc., in 2005. The opening of the Android distribution in 2007 was announced. Google issues the Android code as open-source. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android.

Android has a large community of developers writing applications that spread the functionality of the devices. Developers write in Java. Apps can be downloaded from third-party sites or through online stores such as Google Play. As of February 2012 there were more than 450,000 apps available for Android, and the estimated number of applications downloaded from the Google Play as of December 2011 over 10 billion.

Figure 2.12 Icon of Android
2.2.4.2 Android’s Platform [10]

Android is a layered environment built upon a basis of the Linux kernel, and it includes rich functions. The UI subsystem includes:

- Windows
- Views
- Widgets for demonstrating mutual elements

Android claims a strong array of connectivity choices, including Wi-Fi, Bluetooth, and wireless data over a cellular connection such as GPRS, EDGE, and 3G. A popular technique in Android apps is to connect to Google Maps to show an address within an app. Support for location-based services, for example, GPS, and accelerometers is also accessible in the Android software.

Generally, two areas where mobile apps have struggled to keep step with their desktop counterparts are graphics/media, and data storage methods. Android addresses the graphics challenge with built-in support for 2-D and 3-D graphics, which includes the OpenGL library. The data-storage load is lessened because the Android platform includes SQLite database.
2.2.4.3 Android SDK [11]

Android Software Development Kit (SDK) provides fully featured application development environment for developing applications for Android Phones and Tablets etc. Until now, it has already announced around 8 key versions of its SDK. Its SDK is being announced in three famous platforms such as Windows, Mac and Linux. There are millions applications available for Android platform and it being most famous operating system its making it a hot platform to code for.
Chapter 3 – System Architecture

3.1 Overview of the system

Figure 3.1 shows a diagram about the system connection. First, Android Camera Car (Arm Controlled Board) is connected to the stable Wi-Fi network by a user and it will show the IP of the EVB on the screen. Afterward, when the user wants to control the car remotely, it is required to connect to a stable Wi-Fi network first, and then turn on our designed Android apps. After, the user enters the IP of the EVB and presses the connect button, the whole system will be connected and comes to a stable state. Simply pressing corresponding buttons, actions such as taking photos, recording video, controlling the servo (camera view) and turning on/off LED lights can be accomplished by the user.
Figure 3.1  System connections
3.2 Connection between Android Camera Car (Arm Controlled Board) and Client’s Android Mobile Devices

Since both applications of the Android Camera Car (Arm Controlled Board) and client’s mobile devices are written in java, the connection between them adopts the class – DatagramSocket and DatagramPacket which are provided in the java.net packet in the Java Platform.

In the program of the applications in both sides of the client and the server, we have already pre-set the port (4567) for command communication and the other one (7777) for video real-time streaming between the Car (Evaluation Development Car) and client’s Android mobile devices. At the beginning, the EVB should be connected by the user to a stable Wi-Fi network, and then its IP will be shown on the screen.

The method using: 
```java
csocket = new DatagramSocket(4567);
cpacket = new DatagramPacket(data,1);
```

At the same time, the program in the server’s side creates a DatagramSocket with the port 4567, and waits to listen and receive the command packet from the client. When the client wants to control the car remotely, they should enter the IP of the Car and then press the connect button. At this stage, the program will create a DatagramSocket with the port 4567 and a DatagramPacket with command in byte. After this stage, the connection command packet is sent to the server’s side.

The method using: 
```java
csocket = new DatagramSocket(4567);
cpacket = new DatagramPacket(data,data.length,serverAddress,4567);
csocket.send(cpacket);
```
Meanwhile, in the client’s side, a DatagramSocket with the port 7777 is also created in order to receive the video packet sent from the server’s side. On the other hand, the server’s side receives the connection command packet from the client’s side; the IP from the sender will be identified automatically and a DatagramSocket with the port 7777 will also be created in order to send the video packet to the specified client’s side. As a result, the whole system is connected.

The method using:

```java
socket = new DatagramSocket(7777);
packet = new DatagramPacket(buffer,num,serverAddress,7777);
socket.send(packet);
socket.receive(packet);
```
3.3 Sending command from the Client’s Android Mobile

Devices to Android Camera Car (Arm Controlled Board)

Since the client wants to control the car remotely, the Android application in the client’s side should send out the command to the Car (Arm Controlled Board). Because the connection can be set up following the way we discussed in the last session, the client’s side can use the socket with the port 4567 to send the command to the server’s side.

The method using: `csocket.send(cpacket);`

The application will keep running until the user wants to exit.
3.4 Reading input command from Client’s Android Mobile Devices in Android Camera Car (Arm Controlled Board)

Since the Car (Arm Controlled Board) is similar to the brain of this system, it needs to determine the input commands from the client’s Android platform.

The method using: `csocket.receive(cpacket);

After receiving the packet which is in byte format, it will be better if we change the byte format into the string one which makes the command determination much easier with the use of if/else case.

The method using: `msg = new String(cpacket.getData());

msg.equals("/*The cmd that we want in short form*/")

After receiving and determining the commands, different actions can be accomplished by the user.

This action can keep running until the user wants to turn off the car.

3.5 Sending Video packets from Android Camera Car (Arm Controlled Board) to Client’s Android Mobile Devices

After the connection setup, the server’s side (Arm Controlled Board) will open the camera and initialise the setting. After that, the video data will be received into file input stream before sending them; the program will sends the H.264 header of the EVB and the video packets to the client’s side consecutively. In fact, the hardware encoding method has already been used in the process.

The method using: `packet=new DatagramPacket(buffer, num,serverAdddress,7777);

socket.send(packet);`
In this transmission process, UDP is used in the video data transmission process.

The main characteristic of UDP as in the following give rise to its high suitability in media sharing:

• Connectionless (out of sequence packets)
• Unreliable (can packet lost)
• No rate control
• No flow control
• No congestion control

In another word, it is not necessary to take the packet loss into consideration, and also not necessary to retransmit the packet is there’s a packet loss due to the lack of flow and congestion control.

3.6 Receiving Video packets from Android Camera Car (Arm Controlled Board) in Client’s Android Mobile Devices

After the connection setup, the client’s side (Android Platform mobile device) will open a receiving socket with the port 7777 which serves to listen and receive the video packets.

The method using: packet=new DatagramPacket(data,2048);
socket.receive(packet);

After receiving the video data, it will be passed (transferred/transmitted into) to the native function in “jni” which decodes the video data into RGB, and then it can be displayed on screen.
The method using: ```public native int decode(byte[] in, int insize, byte[] out);```

### 3.7 Sending command message from Android Camera Car (Arm Controlled Board) to LPC1768 through UART

After determining the command, if the command is related to actions such as the movement of the car, the movement of servo or lighting of LEDs, the Car (Arm Controlled Board) should sends command message to LPC1768 through UART since those actions should be handled by LPC1768. Moreover, it is not necessary for EVB to receive the feedback from LPC1768, therefore, the program will just uses the sending function of UART.

The method using: ```Uart2C.openUart(0); Uart2C.sendMsgUart("/*The cmd that need to pass to LPC1768*/");``` 

In fact, this UART function is not provided by Android Inc. officially. Therefore, in order to use this, after doing research works for a long time we finally found a sample program and modified it so that it can fit for our project.

### 3.8 Receiving command message from Android Camera Car (Arm Controlled Board) in LPC1768 through UART

One of the duties of LPC1768 is to receive the command sent by the Car (Arm Controlled Board), since the commands are sent by EVB through UART and
LPC1768 no need to send back the feedback or commands to Arm Controlled Board. In this program, UART is provided by the NXP officially so that we can use the function directly.

The method using:  

\[
\text{UART0\_Init();}
\]

\[
\text{Input = UART0\_GetChar();}
\]

Before receiving the command, the program should initialize UART port, and the UART0\_GetChar() this function will keep listening the After receiving the command, the received command is in integer form which is according to the ASCII format. Therefore, when the program determines the commands by using if/else case, it will follows the ASCII code. After determining the command, the LPC1768 will do the specific action.

### 3.9 Set the PWM to control the Car Motor

Before receiving the command, the program should initialize the PWM. There are 7 PWMs, first we set them into logic low. After receiving the command, PWM0 should be set to cycle and then PWM1, PWM2, PWM3 and PWM4 should be set to different logic in order to achieve the movement. The movement of the car which is determined by the user lead to the different designs in our setting.
Figure 3.4  Typical motor control application circuit showing directional and braking control

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 A2 ENA</td>
<td>OUTA</td>
<td>B1 B2 ENB</td>
<td>OUTB</td>
</tr>
<tr>
<td>H L H</td>
<td>OH</td>
<td>L L H</td>
<td>OH</td>
</tr>
<tr>
<td>L L H</td>
<td>OL</td>
<td>H L H</td>
<td>OL</td>
</tr>
<tr>
<td>H H H</td>
<td>OL</td>
<td>L H H</td>
<td>OL</td>
</tr>
<tr>
<td>L H H</td>
<td>OL</td>
<td>H H H</td>
<td>OL</td>
</tr>
<tr>
<td>X X L</td>
<td>Z</td>
<td>X X L</td>
<td>Z</td>
</tr>
</tbody>
</table>

L = Low logic level; H = High logic level
Z = High Impedance (off state)
OH = Output High (sourcing current to the output terminal)
OL = Output Low (sinking current from the output terminal)
X = Don’t Care

Figure 3.5 Truth Table of HIP4020
Figure 3.6 Pin Description of HIP4020

Based on the design by Jason, he connects A1, B1 together for one motor and A2, B2 together for the other one. Moreover, when we see the Pin Description of HIP4020, we can find that A1 and B1 are serve to control to directional rotation of a motor, and A2 and B2 serve to activate the dynamic braking of the motor. Therefore, in designing the program, I set PWM1 and PWM3 to logic low to go forward, and logic high to go backward. In addition, I set PWM2 and PWM4 to logic low to run the motor in its maximum speed, and logic high to stop the motor.

### 3.10 Combine all of the functions into one application in Android Camera Car (Arm Controlled Board)

Typically, the functions of the application in the Arm Controlled Board are:

1. Receiving controlling command from the client’s side;
2. Sending the video packet to the client’s side;
3. Determining and sending the command message to LPC1768 in order to control the movement of motors and servo and lighting LEDs;

4. Displaying the current command that the client’s want;

5. Displaying the IP of Arm Controlled Board;

6. Determine the IP of the client’s so that make the connection simpler; and

7. Recording the video in the SD card if the client’s want.

Theoretically, functions 2 and 6 (originally Jason is responsible for them) work when they run independently. However, when I tried to merge them into an application, there are conflicts between them. After my modification, the application now can run for many times as much as user want.

For example, in my program, a thread is created for video streaming. However, when the user wants to record and stream the video simultaneously, the program needs to close the original thread and then open another thread for the simultaneous streaming and recording. In another example, if the user wants to exit the application and wants to notify the Car stop streaming video to the user. The question is how to achieve the closing of the original thread. Chapter 4 will further illustrate the solution of this question.
3.11 Combine all of the functions into one application in Clients’ Android Platform Mobile Devices and video streaming improvement

The functions of the application in the Android Mobile Devices are in the followings:

1. Sending controlling command such as controlling the movement of the car to the server’s side;
2. Receiving the video packet from the server’s side;
3. Decoding and displaying the video data on the screen;
4. Displaying the current command that the client’s want;
5. Sending the command of the car’s movement by using Motion Sensing;
6. Capturing and storing the picture from the client’s side; and
7. Improving the quality of the video streaming in poor CPU performance.

Theoretically, Jason is responsible for part 2, part 3 and part 6 and Chris is responsible for part 5 and it can be worked when they run independently. However, when I tried to merge them into an application, there are conflicts between them. After my modification, the application now can run, and furthermore it can accept users’ various connection requests.

Apart from that, the video streaming is related to the performance of the CPU since it requires software decoding. That means the better performance the CPU has, the better quality of video streaming that the client can be received. If the performance of CPU is poor, some errors such as green or grey areas may display on the screen.

In order to deal with this problem, I tried to create two threads to receive the video packets. However, the two threads cannot receive the video packets simultaneously. If they receive the packets at the same time, the frames will be received in a wrong arrangement. Therefore, “semaphore” is used to solve this problem. Chapter 4 will also focus on this solution in detail.

Apart from that, in order to make the control more user-friendly, for example, controlling the movement of the car, I used onKeyListener but not the onClickListener since the former makes the control of the car more convenient. Furthermore, after pressing any button on the screen, the devices will vibrate in order
to alarm users: “You have pressed the button successfully.” The purpose of those modifications is to make the control panel more user-friendly.

Figure 3.8  The function of the combined application in the client’s Android mobile devices
Chapter 4 – Methodology

4.1 Hardware Part

4.1.1 Designing a specified functional REAL6410

When we got the REAL6410 product, we found that not all components or functions are as we expected in our project. Therefore we have eliminated some components which are in no use and kept important ones for our project.

I use the software called Design Explorer DXP which is recommended by Sengital Limited. Their staff also recommended us to study the tutorial provided by the software by ourselves. After the short tutorial, I start to draw my first schematic in my life.

Figure 4.1 Design Explorer DXP
1. Camera port

   - Since we want to video streaming data to the client, we must first reserve a camera port as following.
2. Wi-Fi and 3G port

- Since we want to video streaming data to the clients through Wi-Fi or 3G

Figure 4.4 Schematic of Wi-Fi port

Figure 4.5 Schematic of 3G port
3. RS232 and UART port

- Since we need to port the Android OS system from PC through RS232 port.
- While UART port is used to transmit command message to LPC1768.

![UART Schematic](image)

Figure 4.6  Schematic of RS232 and UART port
4. LCD display port

- Since we need to let the user set the Wi-Fi setting before use, we need to reserve the LCD display port.

![Schematic of LCD Display port](image)

Figure 4.7 Schematic of LCD Display port

5. Keypads

- After setting the Wi-Fi, the user can click escape button to go back the main layout of the application.

![Schematic of Keypads](image)

Figure 4.8 Schematic of Keypads
6. SD Card port

- Since the user may want to record the video when they are enjoying video streaming, we have a function for the user to store their record video in the Arm Controlled Board. Therefore, SD Card port is reserved.

![SD Card MEMORY](image1)

Figure 4.9  Schematic of SD Card port

7. Mini USB port

- The reason of reserving the mini-USB port is to port the OS and for android development debugging use.

![MINI USB](image2)

Figure 4.10  Schematic of Mini-USB port
8. Power port
   - This is the essential port of Arm Controlled Board. Without it, ACB cannot work.

Figure 4.11  Schematic of Power port

9. Adapter Header
   - This is just the first version we made. Alan suggests that we can first design the Arm Controlled Board, if it works, we can still work on to redesign another Arm Controlled Board that we can still reuse the core board on another design. Therefore, I made 4 headers for the adapter.
However, some of the schematics are not provided by the sellers or Sengital, I need to draw new schematics into the schematic library by myself, such as voltage regulator and 4 adapter headers.
As a result, the full schematic of specified function of REAL6410 is like this.
4.1.2 Routing the PCB layout of core board E210COREV2

Since REAL6410 does not support video recording, we need to do sourcing on the internet and finally we find that E210COREV2 can be used to video recording. This time, I need to route the PCB layout of E210COREV2. Before routing the PCB, we should prepare the entire footprint that we want to route. I had almost footprint that I want except the foot print the core board. Therefore, I need to create and draw the specified footprint for this core board first.

For drawing the footprint, we need to know the distance between each foot. If the distance is not accurate, we cannot solder the IC on the contact point exactly. As a result, some of the functions may be repealed. The distance between the feet of the core board is provided by the seller.
Apart from that, I still need to draw the footprint of 4 adapter headers too.

Other footprints are similar to the REAL6410 and Sengital so that all footprints are well prepared and I can route the PCB layout of core board.
Figure 4.18  PCB Layout of E210COREV2 Core Board
4.1.3 Designing a specified functional of Arm Controlled Board based on E210COREV2

After designing the core board of E210COREV2, we know that the architecture of E210COREV2 is quite similar to E6COREV3. Therefore, I try to design a schematic similar to the schematic I designed before and a circuit which is one of the project requirements.

1. Camera port

- Since the pins of E210COREV2 are more than that of E6COREV3, and some pins can be directly connected to the camera port so that we can use less modules or ICs in the design of the circuit.

![Camera Port Schematic](image)

**Figure 4.19** Schematic of Camera port with E210COREV2

2. Wi-Fi and 3G port

- At this time, the seller recommended us to use the new Wi-Fi module since the size of this new Wi-Fi module is smaller, and
the module can be directly connected to the core board.

Therefore, I need to draw a new schematic of Wi-Fi module in schematic library.

Figure 4.20 Wi-Fi modules in Schematic Library

Figure 4.21 Schematic of Wi-Fi module with E210COREV2

- For 3G port, we still can use the same design with REAL6410.
Figure 4.22  Schematic of 3G port with E210COREV2

3. RS232 and UART port

- At this time, we still need to reserve the RS232 port to port Android OS system from PC through this port.

- Based on previous experience, we just need to reserve one RS232 port so that we can eliminate one RS232 port and the 4 options switch.

- Also, I have changed ADM3202 to MAX3232 since ADM3202 can just operate at data rates up to 460 kbps but MAX3232 can operate at data rates up to 1Mbps. Another reason for me to change the model is that they share the same footprint.

- As a result, the UART port can be directly connected to the core board and also MAX3232 to the core board.
4. LCD Display port

- Since we change the model of screen, the header must be redesigned for display.
5. Keypads

- Similar to the last design, we need to reserve the Keypads for the user to go back the main layout of application after setting the Wi-Fi.
6. SD Card port

- Similar to the last design, the Arm Controlled Board can record and stream the video simultaneously and the recorded video should be stored in the SD Card of the EVB. That is the reason to continue to reserve the SD Card port.
7. **Mini USB port**

- Similar to the previous experience, we need to reserve the mini-USB port to port the Android OS and for the android development of android in debugging use. It is similar to the last design.

![Figure 4.29 Schematic of Mini-USB in Schematic Library](image)

**MINI USB**

![Diagram of Mini-USB connection](image)
8. Power port

- This is the essential port without which the Arm Controlled Board cannot work.

Figure 4.30  Schematic of MINI-USB with E210COREV2

9. Adapter Header

- Similar to the previous experience, we need to make the first version of Arm Controlled Board. In case there are some problems of the board, we still plug the core board out for the reuse purpose.

Figure 4.31  Schematic of Power port with E210COREV2
Finally, the full schematic of Arm Controlled Board’s specified functions is like this.
The considerate part is to make a way for power supply and ground to connect to the core board since the core board requires stable power supply. That’s the very main idea to work with whole device.
4.2 Software Part

In this part, the program written for this project will be introduced briefly. First of all, a structure diagram of the program will be given and the usages of the classes and methods will be briefly talked about. After that, the flow of the system connection and the operation will be shown in a flow chart. Apart from that, there are two applications that one is for Arm Controlled Board and another one is for the client’s Android Mobile Devices. I will divide them into two parts for me to explain.

4.2.1 Android Application of Arm Controlled Board

4.2.1.1 Structure of the program

Figure 4.36 is a diagram showing the structure of the program in tree form. The main class of the program is Uart_CameraActivity{}. It includes the almost main method of the program. Once the user click the start button, a new thread will be started which is called ServerThread{}. The duty of this sub classes is to receive the command from the client’s Android mobile devices.
4.2.1.2 Program flow of the system connection and operation

In Figure 4.38, the flow of the system connection is shown. After starting the application, the program will create a user interface. Once the user click the start button, the program will open a new thread (ServerThread) for listening and receiving the command from the clients. In the thread, socket with port 4567 will be created and wait to receive the command packet. Apart from that, the brightness of the screen will be set dimmer since it will consume too many power if there are no one use the car but the brightness of screen is still bright. Once the socket receives a connection command, the program will open a new thread to do the followings:
1. Display the IP of Arm Controlled Board

2. Determine the IP of the senders

3. Initialize the parameter of the Camera and MediaRecorder

4. Open a new DatagramSocket with 7777 to send out video packet.

   (Since the video streaming is done by Jason, I will just simply explain the flow of the program.)

   After that, the system connection is finished.

   Once the user want to record and stream the video simultaneously, a specified command should be received. The program will set the global variable - goToRecord to be true. The program will first check is there any SD card has already been inserted to the Arm Controlled Board, if no, just show the alarm message on the screen, if yes, release the Camera and MediaRecorder, disconnect and close the socket and end the current thread. Using the function – change_to_record() to re-initialize the setting of camera and MediaRecorder and finally use the function StartVideoRecording1() to start a new thread to stream the video out to the client and record the video into the SD card of Arm Controlled Board simultaneously. Similarly, once the user just wants to stream the video, a specified command should be received. The program will set the global variable - goToStream to be true. The program will first check is there any SD card has already been inserted to the Arm Controlled Board, if no, just show the alarm message on the screen, if yes, release the Camera and MediaRecorder, disconnect and close the socket and end the current thread. Using the function – change_to_record() to re-initialize the setting of camera and MediaRecorder and finally use the function
StartVideoRecording1() to start a new thread to stream the video out to the client and record the video into the SD card of Arm Controlled Board simultaneously.

Once the user want to control the movement of the car, movement of the servo or lighting LEDs, a specified command should be received. The program will send a UART message to LPC1768 by using the function – SendMsgUart().

Once the user want to leave the application, an exit command should be received by the Arm Controlled Board. The program will set the global variable – stop to be true and release the Camera and MediaRecorder, disconnect and close the socket. Disconnect all the connection with the user and wait for another new user.
Figure 4.35  Flow Diagram of program in Arm Controlled Board
4.2.2 Android Application in Android Mobile Device

4.2.2.1 Structure of the program

Figure 4.38 is a diagram showing the structure of the program in tree form. The main class of the program is FfmpegdecodeActivity{}. It is responsible for the command sending. Once the user enter the IP of Arm Controlled Board and click connect button, the function – PlayVideo in subclass Vview will start a new thread. The duty of this function is to receive and decode the video packets to display on the screen of client’s Android mobile devices. Another duty of this class can help the clients so capture the current streamed image when they want. Apart from that, SplashScreen{} is to show the opening and welcoming screen to the user. VerticalSeekBar{} is used to make the seek bar placed vertically since Android do not provide official vertical seek bar, we need to make it by ourselves.
Figure 4.36 Structure of the program in client’s Android Mobile Device
4.2.2.2 Program flow of the system connection and operation

In Figure 4.38, the flow of the system connection is shown. At the beginning of the application, the application will have the splash screen to display the icon of Android Camera Car, CityU and Sengital Limited. Afterward, entering the application, the program will create a user interface and setting different Listeners in order to determine the action to send out different corresponding commands. For example, connect button will be using onClickListener to listen, Up, Down, Right and Left will be using onTouchListener to listen in order to let the user control the car easily. However, I need to ensure that the application should be connected to the Arm Controlled Board before sending the command out; otherwise, error will be occurred. The global variable – connected will be the flag to check the connection. Moreover, after pressing any buttons, the device will be vibrated in order to alarm the user that button is pressed successfully. After that, User should enter the IP of the Arm Controlled Board correctly and then press connect button. The system will be connected to the Arm Controlled Board successfully. The user can control the car freely. However, the video decoding function that finished by Jason is worked in good performance of CPU since software decoding is used. The most important part of this application is how to receive the video packet more efficiently and has better performance. To solve this issue, I start two threads at the same time to receive the video packet in order to make sure to receive all of those packets during one thread is decoding. However, how can I deal with the problem that the thread may receive the data in the same frame and the data will merge in a wrong
sequence? After seeking help from Dr. Ray Cheung, he suggested me to try semaphore. It really inspires me a lot and finally I can use semaphore to achieve the goal that enhance to a better performance.
Figure 4.37  Flow Diagram of program in client’s Android Mobile Device
4.2.3 Program in LPC1768 for controlling the movement of motors

4.2.3.1 Structure of the program

Figure 4.39 is a diagram showing the structure of the program in tree form. The main class of the program is main{} and the sub class of the program is pwm{} and uart{.}

main{} is responsible for several tasks and they are:

i. Calling out the function from the other classes to initialize the UART and PWM of LPC1768

ii. Receiving the command sent from Arm Controlled Board through UART

iii. Determining the command and set the PWM pins for different actions

Apart from that, the duty of pwm{} is:

i. Initializing the PWM pin

ii. Setting the voltage of PWM pin.

And finally, uart{} is responsibility of uart{} is:

i. Providing the UART receive function
4.2.3.2 Program flow of the system connection and operation

The full program flow is shown in the Figure 4.40 completely. Once starting the program, the program will initialize the PWM and UART pin. Afterward, the program can start to listen and receive the command through UART sent by Arm controlled Board. Once the command is received, the program will determine what the received command is. Based on the hardware design, PWM 1 and 3 is used to control the direction of the motor and PWM 2 and 4 is used to control the speed of the motor. Therefore, the program can set the PWM pin according to this rule to achieve the specified action.
Figure 4.39  Program flow of the program in LPC1768
4.2.4 Modifying the camera driver

Because Alan Lam, the Chief Executive of Sengital Limited, discovered that the camera setting is just 15fps when he used the Android official camera apps in REAL6410. He requested us to change the frame rate at least to 30fps.

After studying the provided camera module, we found that the camera module is designed by OV3640 image sensor. In the reference of Linux Kernel for REAL6410, we found that there are quite a lot of files related to camera driver. Then, we did further research and found that I just need to modify the camera.h file.

The main point is which part of the program should I modify in order to achieve the goal? I found the datasheet of OV3640 and use the information found in the datasheet to compare the data in the header file, I found the way to modify the program.

For example, if I want to flip or mirror the camera image on the displayed screen, firstly, I need to find the corresponding function control register address from the datasheet.

![Figure 4.40 Mirror and flip control of OV3640](image-url)
After we found the address, we can add a short specified format program into the specified variable in 2-D array. For example, if I want to Mirror ON and Flip ON to the image, I need to add “{0x30,0x7C, 0x11}” so that the image will be mirrored and flipped. Therefore, in same theory, we need to find out which register address controlling the frame rate and add the short program on that and we can achieve the result. However, there are no register to control the frame rate. We study the datasheet again and finally find the reason.

According to the datasheet, “The OV3640 has an image array capable of operating at up to 15 frames per second (fps) in QXGA” and “XGA (1024x768): 30fps for XGA and any size scaling down from XGA”

That means, if we want to change the frame rate of this chip, we need to change the image format of this chip first. Finally, we find the address of control register that can change the format of image.

![Figure 4.41 System Control Register of OV3640](image)

The program that we need to add is: {0x30,0x12, 0x10} and 30fps is achieved.
Chapter 5 – Operation Procedure and Result

5.1 Operation procedure

In order to get the connection successfully, the following steps should be done.

For Arm Controlled Board:

Step 1. Clicking the Wi-Fi setting button to configure Wi-Fi setting

Step 2. Pressing the start button to start receiving the command

For Android Mobile Device:

Step 1. User should enters the IP of Arm Controlled Board and then click connect button to set the connection

Step 2. User can clicks the command buttons for sending the controlling commands to the Arm Controlled Board

After following those procedures, the connection will be setup.
5.2 Result

The followings are the result.

5.2.1 Hardware part

Figure 5.1 Front view of Arm Controlled Board with core board E6COREV3

Figure 5.2 Back view of Arm Controlled Board with core board E6COREV3
Figure 5.3  Front view of Specified function Arm Controlled Board for E210COREV2

Figure 5.4  Back view of Specified function Arm Controlled Board for E210COREV2
Figure 5.5  Front view of Adaptor Board of E210COREV2

Figure 5.6  Back view of Adaptor Board of E210COREV2
Figure 5.7  Android Camera Car
5.2.2 Software Part

Figure 5.8 The 1st design of the application layout

Figure 5.9 The capture of the application after connection setup in Android Mobile Device
Figure 5.10  The capture of the application after connection setup in Arm Controlled Board

Since the results which need to be show in dynamically, the demonstration is recorded in video and uploaded on YouTube: http://www.youtube.com/watch?v=iQngLYHQSM
Chapter 6 – Discussion

In this part, some technical problems had been faced in this project and the method that can solve the problems will be discussed.

6.1 Hardware Part

6.1.1 Studying the datasheets of the different ICs for designing the Arm Controlled Board

In the design of arm controlled board for ARM11, we have based the references provided by the seller to design our own specified functional of the arm controlled board. However, we known nothing at the very beginning and need to study the datasheet of different ICs which is used in the reference board in order to get more information about the function and the usage of the ICs. Therefore, after studying those datasheet of ICs, we had enough details to design a schematic of the specified functional Arm Controlled Board.

6.1.2 Drawing the schematic and PCB footprint into the library

In the design of arm controlled board for ARM11, we have based the references provided by the seller to design our own specified functional of the arm controlled board. We can use those reference schematic designs to develop our specified functional Arm Controlled Board. However, since the ARM11 chips doesn’t support hardware encoding and video recording, we need to change our design and Cortex – A8 is our final choice. We need to redesign the whole schematic of Arm
Controlled Board for Cortex – A8. Some of them can be reused from the last design and some of them need to draw and design by myself into our own schematic and footprint library. The solution of this problem is studying the datasheet of the core board first. Afterward, we need to find the modules and ICs to fit for the core board and study the datasheets of those suitable modules and ICs again then we can draw the schematic and the footprint of the ICs into own library of the software.

---

**Figure 6.1** One example of the hardware design details from the datasheet
6.1.3 Soldering on the Arm Controlled Board

All the hardware components need to be soldered by ourselves. After the got the basic training provided by Sengital Limited, we learned soldering skill. However, some of the pins in the Arm Controlled Board is very difficult to be soldered with the IC components. For example, the LCD Display header, the gap between two pins is very close together so that, if we solder the components in high temperature or for a long time, it will be damaged. Therefore, there are no other ways to solve this problem, we just can keep solder that header until it succeeds.

Figure 6.2  Structure of the pins of LPC Display header in real view
6.2 Software Part

6.2.1 Integrating the Android Application in both Arm Controlled Board and client’s Android mobile device

Before integrating the applications, I have little knowledge for wiring java language. I kept reading online materials on the website of Android Developer and learn the syntax and function. Afterward, I need to modify and merge the programs which are done by my group mate into one application.

For the application of Arm Controlled Board, it is very hard to make the video encoding and streaming program to run properly again if user has already exit the program and try to reconnect afterward. I need to study the flow of the program very carefully in order to know what this step stands for and is doing. Although I know the procedure of the program flow and the right condition, the application still cannot be reconnected. We tried a lot of methods to solve this problem and finally we found out that we should release the camera and disconnect all the sockets before the next connection. For the application of the Android mobile device, it is easy to make some command buttons and send the command packet to the Arm Controlled Board. However, the critical and difficult part is to solve the performance problem. As the report mentioned before, the program done by group mate cannot receive any video packets when it is doing decoding process. Therefore, I need to think about how to solve it and I tried a lot of method again. I thought a method that using two threads to handle the problem, but how can the program control the thread effectively? Finally, I need to say thank you to Dr. Cheung, he gave me a good
suggestions that using semaphore to solve the problem and it works. It is used around 1.5 months for me to integrating, modifying and enhancing the application for Arm Controlled Board and Android mobile devices.

6.2.2 Modifying the camera driver

It seems that it is very easy to change the camera frame rate to 30fps, however, I had used for a long time to find out that which files should be modified. In fact, there is no solution on the internet. The method I find out is I checked the file one by one by myself. There are several files that can be modified, but only camera.h should be modified since I found out that the main program of running the camera driver is camera.c, it shows that it relies on the header file camera.h. However, next question came up very soon that is, how the file can be modified. There are many unknown data in the camera.h. However after studying the datasheet, I got the answer and the goal is achieved. The main way to solve this problem is keep on investigating the program and studying more about the datasheet.
Chapter 7 – Further Development

In previous chapters, we have discussed the technical problems we confronted with. Bases on the problems and possible solutions, in this chapter, we will propose the directions of further development.

7.1 Designing Arm Controlled Board for Cortex-A8 in version 2

In fact, the Arm Controlled Board can be designed in a smaller size and this may become a practical way in the future. For example, RS232 port is used to import the Android OS into the core board of Cortex - A8 which is useless for the application. We can use the core board which is already imported in Android OS for the Arm Controlled Board without the RS232 port. Another port for example is the mini USB port which is for developing debugging can be eliminated it after completing the development. After redesigning as I started, the size of the next generation of the Arm Controlled board may reduce 80% comparing with the first generation product.

7.2 Adding GPS Navigation

The Arm Controlled Board supports Wi-Fi function and 3G function later on; we can add GPS navigation on in the integrated application of Arm Controlled Board so that the user connecting the device can know the location of the Arm Controlled Board. This may not be useful if the Arm Controlled Board is just embedded on the Car base, but it must be useful if we redesign the Arm Controlled Board in a smaller size and embed it on other remote devices such as remote helicopter or remote devices for Dementia patients’ use. The user can keep track on the Arm Controlled Board if the board is highly probable.
7.3 Adding Robot Arm

On some occasions, we must deal with tasks remotely. However, there is no robot arm on the current generation of our product. For further development, we can add a robot arm on our Android Camera Car providing a device for users to achieve actions such as picking or releasing the objects. In fact, we have tried to find a way to embed robot arm on our Android Camera Car. Although there are technical problems, we have already found out possible solutions. Therefore, it is practical to supplement our current product.
Chapter 8 – Conclusion

In this project, there are many tasks for developing this product. For my duty, I am responsible for,

- The Schematic of Arm Controlled Board with ARM11
- The Schematic of Arm Controlled Board with Cortex-A8
- The PCB layout of the adaptor board with Cortex-A8
- Modifying the Camera Driver
- Integrating and modifying an Android Application of Arm Controlled Board
- Integrating and modifying an Android Application in Android Mobile Device
- Writing a program for controlling the motor driver

These duties are difficult to implement but finally they can be solved by me. The most important point is, Android Camera Car is made very successfully. This really impresses me a lot. In this project, I have learnt a lot. Not only the knowledge and skill of this related topic, but also planning, decision making, time management, information searching and problem solving can be learnt in this period. As a whole, I am very happy that I can work with my group mates: Jason Ng and Chris Lam. They are very helpful and energetic; we always share our ideas in order to make the product to become as ideal as we want. Moreover, I want to say thank you to my supervisor Dr. Ricky Lau and my assessor Dr. Ray Cheung. They gave me a lot of supports such as giving comments on our project and our presentation PowerPoint. I can say that, this project makes me grow up and I am so happy to have this experience.
References


[2] Description of Arm Controlled Board for E6COREV3. Available from:
http://www.realarm.cn/pic/?76_451.html


[5] Soldering - Wikipedia, the free encyclopedia. Available from:
http://en.wikipedia.org/wiki/Soldering

[6] Fabrication - Wikipedia, the free encyclopedia. Available from:
http://en.wikipedia.org/wiki/Fabrication_(metal)

[7] Universal asynchronous receiver/transmitter - Wikipedia, the free encyclopedia. Available from:

[8] Pulse-width modulation - Wikipedia, the free encyclopedia. Available from:
http://en.wikipedia.org/wiki/Pulse-width_modulation

[9] Android (operating system) - Wikipedia, the free encyclopedia. Available from:
http://en.wikipedia.org/wiki/Android_(operating_system)

[10] Introduction to Android development. Available from:

http://www.mobulite.com/android/introduction-to-android-sdk/
# Appendix

<table>
<thead>
<tr>
<th>Variables</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>socket</td>
<td>DatagramSocket</td>
<td>Used to send out the video packet</td>
</tr>
<tr>
<td>packet</td>
<td>DatagramPacket</td>
<td>Used to pack the video data in side this packet</td>
</tr>
<tr>
<td>csocket</td>
<td>DatagramSocket</td>
<td>Used to receive the command</td>
</tr>
<tr>
<td>cpacket</td>
<td>DatagramPacket</td>
<td>Used to store the command</td>
</tr>
<tr>
<td>data</td>
<td>Byte string</td>
<td>Used to store the data of Datagrampacket in byte form</td>
</tr>
<tr>
<td>msg</td>
<td>String</td>
<td>The string of current status</td>
</tr>
<tr>
<td>serverAddress</td>
<td>InetAddress</td>
<td>Used to store the IP in InetAddress form</td>
</tr>
<tr>
<td>stop</td>
<td>Boolean</td>
<td>Used to check whether the client exit or not</td>
</tr>
<tr>
<td>using</td>
<td>Boolean</td>
<td>Used to check whether the car is already used or not</td>
</tr>
<tr>
<td>Ip</td>
<td>String</td>
<td>Used to store the IP of Arm Controlled Board</td>
</tr>
<tr>
<td>turned</td>
<td>Boolean</td>
<td>Used to check whether the start button is already pressed or not</td>
</tr>
<tr>
<td>led</td>
<td>Boolean</td>
<td>Used to check whether the LEDs is turned on or not</td>
</tr>
<tr>
<td>Variable</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>goToRecord</td>
<td>Boolean</td>
<td>Used to check whether the threads has already go to the function go_to_record() or not</td>
</tr>
<tr>
<td>goToStream</td>
<td>Boolean</td>
<td>Used to check whether the threads has already go to the function go_to_stream() or not</td>
</tr>
<tr>
<td>isRecording</td>
<td>Boolean</td>
<td>Used to check whether the application is recording or not</td>
</tr>
<tr>
<td>tv</td>
<td>TextView</td>
<td>Used to display the status</td>
</tr>
<tr>
<td>receive</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for receiving the command</td>
</tr>
<tr>
<td>wifis</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for Wi-Fi settings</td>
</tr>
<tr>
<td>r</td>
<td>Runnable</td>
<td>Used to start to send out the video packet</td>
</tr>
</tbody>
</table>

Table 2: Variables used in FfmpegdecodeActivity{}
<table>
<thead>
<tr>
<th>Variable</th>
<th>Class/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rightward</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for sending rightward command</td>
</tr>
<tr>
<td>stop</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for sending stop command</td>
</tr>
<tr>
<td>connect</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for sending connect command</td>
</tr>
<tr>
<td>capture</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for capturing the current received image</td>
</tr>
<tr>
<td>record</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for sending recording command</td>
</tr>
<tr>
<td>led</td>
<td>ImageButton</td>
<td>Used to create an ImageButton for sending turning ON/OFF LEDs command</td>
</tr>
<tr>
<td>csocket</td>
<td>DatagramSocket</td>
<td>Used to sending out the packet</td>
</tr>
<tr>
<td>cpacket</td>
<td>DatagramPacket</td>
<td>Used to packetize those command data into packet</td>
</tr>
<tr>
<td>show</td>
<td>TextView</td>
<td>Used to show the current car status</td>
</tr>
<tr>
<td>connected</td>
<td>Boolean</td>
<td>Used to check whether the client’s mobile device is connected with the Arm Controlled Board or not</td>
</tr>
<tr>
<td>command</td>
<td>String</td>
<td>Used to store the corresponding command in String format</td>
</tr>
<tr>
<td>recorded</td>
<td>Boolean</td>
<td>Used to check whether the Arm</td>
</tr>
</tbody>
</table>
Controlled Board is recording or not

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>switchLed</td>
<td>Boolean</td>
<td>Used to check whether the LEDs is turned on or off</td>
</tr>
<tr>
<td>data</td>
<td>Byte String</td>
<td>Used to store the command data in to byte string format</td>
</tr>
<tr>
<td>semaphore1</td>
<td>Semaphore</td>
<td>Used to protect the receiving video packet process</td>
</tr>
<tr>
<td>semaphore2</td>
<td>Semaphore</td>
<td>Used to protect the decoding video packet process</td>
</tr>
</tbody>
</table>

Table 3: Variables in pwm{}

<table>
<thead>
<tr>
<th>Variables</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Integer</td>
<td>Used to store the command sent from Arm Controlled Board</td>
</tr>
</tbody>
</table>

Table 4: Methods used in class Uart_cameraActivity{}

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static String getLocalIpAddress()</td>
<td>Used to get the local IP address from the Arm Controlled Board</td>
</tr>
<tr>
<td>private boolean initializeVideo()</td>
<td>Used to initialize the settings of MediaRecorder</td>
</tr>
<tr>
<td>private void releaseMediaRecorder()</td>
<td>Used to release the MediaRecorder</td>
</tr>
<tr>
<td>private void startVideoRecording()</td>
<td>Used to send out the video packet in order</td>
</tr>
<tr>
<td>Method</td>
<td>Usage</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>private void startVideoRecording1()</strong></td>
<td>Used to send out the video packet and record the video simultaneously</td>
</tr>
<tr>
<td><strong>public void SetBright(float value)</strong></td>
<td>Used to set the brightness of the screen</td>
</tr>
<tr>
<td><strong>public void change_to_record()</strong></td>
<td>Used to change the only video streaming mode into video streaming and recording mode</td>
</tr>
<tr>
<td><strong>public void change_to_stream()</strong></td>
<td>Used to change video streaming and recording mode into only video streaming mode</td>
</tr>
</tbody>
</table>

Table 5: Methods used in FfmpegdecodeActivity{}
Table 6: Methods used in pwm

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t PWM_Init(uint32_t ChannelNum, uint32_t cycle)</code></td>
<td>Used to initialize the PWM pin of LPC1768</td>
</tr>
<tr>
<td><code>void PWM_Set(uint32_t ChannelNum, uint32_t cycle, uint32_t offset, uint32_t direction, uint32_t speed)</code></td>
<td>Used to configure PWM pin of LPC1768</td>
</tr>
<tr>
<td><code>void PWM_Start(uint32_t channelNum)</code></td>
<td>Used to enable PWM by setting the PCR, PTCR registers</td>
</tr>
</tbody>
</table>

Table 7: Command in Android Application

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Connection Request</td>
</tr>
<tr>
<td>u</td>
<td>Forward command</td>
</tr>
<tr>
<td>b</td>
<td>Backward command</td>
</tr>
<tr>
<td>l</td>
<td>Leftward command</td>
</tr>
<tr>
<td>r</td>
<td>Rightward command</td>
</tr>
<tr>
<td>s</td>
<td>Stop command</td>
</tr>
<tr>
<td>x</td>
<td>Exit command</td>
</tr>
<tr>
<td>z</td>
<td>Recording command</td>
</tr>
<tr>
<td>0</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>1</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>2</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>4</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>5</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>6</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>7</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>8</td>
<td>Movement of servo command</td>
</tr>
<tr>
<td>0</td>
<td>Turning ON/OFF the LEDs command</td>
</tr>
</tbody>
</table>

Table 8: Command in LPC1768