EE3080 Design and Innovation Project

Group Report

LED-based Projection on Glass

Student Name:

Cao Tongtong

Chong Chen Hwa

Chuah Xiao Fen

Pay Shi Yun Carin

Vu Viet Linh

Wu Jitong

School of Electrical and Electronic Engineering

Academic Year 2015/16

Semester 2

Table of Contents

ABSTRACT

Heads-up display (HUD) is a revolutionising technology to help improve safety and reduce the dangers of driving by keeping driver's focus on the road. This report introduces our Design and Innovation Project, LED-based Projection on Glass, which is to innovate and prototype a design of HUD system based around projector, microcontroller unit (MCU) and On-board Diagnostics (OBD) system. The aim of the project is to achieve retrieval of information from the vehicle OBD, processing of data and outputting the information through a projector at a short focal distance. The implementation of the project proved to be successful as the software side was able to obtain, process and arrange the data into a visualisation that can be easily understood by all drivers. Optical modifications to the projector were also successful in shortening throw distance while achieving a clear, viewable image. Future improvement opportunities is recommended in the last chapter as well as the reference for the coming DIP groups to carry on the project.

ACKNOWLEDGEMENT

We would like to thank the School of Electric and Electronic Engineering for providing us with this great opportunity to practise our hands-on and project management skills in completing Design and Innovative Project (DIP). We also wish to express our sincere appreciation to the following people who have assisted us throughout the project:

- Our project supervisor, Professor Chen Tupei for his guidance and supervision to our group, and vision on the project direction
- Our project mentors, Mr Paul Zhen and Mr Liu Weizhong for their suggestions, advices and encouragements in carrying out this project
- Our lab assistant , Mr Kang Meng Fai from Integrated Systems Research Lab for his generous assistance
- The labs staffs at Autonomous Robotics Research Lab for their kind assistance in usage of machinery and work tools
- Professor Yoo Seong Woo for his patience and invaluable advice on optics

CHAPTER ONE – INTRODUCTION

1.1 Objectives

- To build a type of Heads-up Display (HUD) based on image projection by a projector, that has a short throw distance of between 10cm to 30cm
- To achieve a projected image that is visible under the common driving environment such as during the day under direct sunlight
- To obtain data from the OBD system and process it to be understandable
- To study the working principle of commercially used projector and how to modify the projector to fit the project usage
- To learn how different types of lenses form images on the screen
- To understand the working principle of OBD system including the OBD simulator and OBD cable
- To learn how to visualise the required data and design the graphic user interface (GUI) through python programming

1.2 Project Overview

Windshield projection technologies could one day replace the current dashboard display due to its transparency and flexibility. We have seen a lot of scenes in science fiction movies implementing this idea for information display and sharing, such as during conducting a meeting or driving.

In recent years, some of the research and development institutes have put funds and manpower to bring this idea into reality. For example, the latest version of Buick Lacrosse 2016 has integrated a glass display system in the car drive system. This head-up display (HUD) system could update speed information instantaneously in several layout designs which allows drivers to customize. Besides that, by communicating with radar system, it helps to alert the drivers when pedestrians are walking around the car and assists to back the car in the parking place. Another independent electronic devices company, Navdy, has developed a small isolated HUD system. Apart from the basic car on-time information display and navigation, it also brings notifications from the apps on driver's phone into their front view, helping them to keep their eyes forward all the time during driving.

The big advantage of the transparent HUD system is that it projects information within the driver's vision so that they can keep their eyes forward to make it a safer drive. In the current driving environment, drivers often have to look away from the road to monitor their speed and fuel on their vehicle dashboard. This has proven to be very dangerous, and have caused numerous incidents over the years. The revolutionising HUD technology help drivers keep their focus on the road and hence improve driving safety. Also, making good use of the front glass space to project the critical information, which is currently displayed on the instrument panel, may lead the revolution and innovation on car interior design, especially for the driver's position.

In our project, a basic information projection system was designed. It aims to provide three critical data for driving – vehicle speed in km/h, engine speed in rpm and coolant temperature in degree Celsius (°C). These three data are displayed on the screen at the same time and are updated with a preset time interval. To simulate the real car environment, a piece of glass with a special film acted as the car windshield and a on-board diagnostics (OBD) simulator helped to generate the series of data representing the car digital control system.

CHAPTER TWO – EQUIPMENTS AND COMPONENTS

2.1 OBD simulator (12VDC, 2A)

The available protocols include:

ISO 15765 CAN (250kbit/s 11bit ID),

ISO 15765 CAN (250kbit/s 29bit ID),

ISO 15765 CAN (500kbit/s 11bit ID),

ISO 15765 CAN (500kbit/s 29bit ID),

ISO 9141 (5 baud init),

ISO 14230 KWP (fast init),

ISO 14230 KWP((5 baud init).

It supports real-time data stream (Mode 1) and the freeze frame data (Mode2). The knobs at top of the simulator can adjust the main parameters of engine and these data displays on the LCD screen dynamically. Users can choose DTC mode and no-DTC mode by pressing the DTC button (Mode3, Mode4). It can also produce the vehicle VIN code and other vehicle information (Mode9)

2.2 OBD ELM327 cable

The protocols supported by ELM327 are:

SAE J1850 PWM (41.6 kbit/s)

SAE J1850 VPW (10.4 kbit/s)

ISO 9141-2 (5 baud init, 10.4 kbit/s)

ISO 14230-4 KWP (5 baud init, 10.4 kbit/s)

ISO 14230-4 KWP (fast init, 10.4 kbit/s)

ISO 15765-4 CAN (11 bit ID, 500 kbit/s)

ISO 15765-4 CAN (29 bit ID, 500 kbit/s)

ISO 15765-4 CAN (11 bit ID, 250 kbit/s)

ISO 15765-4 CAN (29 bit ID, 250 kbit/s)

SAE J1939 (250kbps)

SAE J1939 (500kbps)

2.3 Arduino Uno R3

The board has 14 digital input/output pins, 6 analog in pins, a USB connector, a power connector, a 16 MHz quartz crystal an ICSP header and a reset button.

2.4 Raspberry Pi 2 Model B

The board has 4 USB ports, 40 GPIO pins, a full HDMI port (connect to the projector in this project), an Ethernet port and a Micro SD card slot. Raspberry Pi 2 Model B is a very powerful mini computer using a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core

ARM Cortex-A7 processor and1GB RAM. For practical use, the Raspberry Pi was connected to a monitor, a mouse and a keyboard. An 16 GB SD card with the Raspbian Operating System (NOOBS) was prepared for our Pi.

2.5 Seven - segment four - digit LED

2.6 UNIC UC40 Mini portable projector

TFT LCD image system, 800 lumens (lm) brightness, LED lamp, Projection Distance 1.07m - 3.8m, projection size 34-130", Aspect Ratio 4:3/16:9, Keystone Correction ±15°, Power Supply 100-240V 0.9A(Max) 50/60HZ, Power Consumption 55W(Max), Inputs USB/AV/SD/HDMI/IR, Output 5V 500mA, Product Size 20.1×15.3×6.75cm/7.91 x 6.02 x 2.65 inch, Weight: 1002g

2.7 50W COB LED and 50W LED Driver

The 50W LED has 10x5 cells, and is driven by a 10S5P 50W LED driver with Input AC 110-280V 50/60Hz, Output DC 28V-40V, 1.5A ±5%

2.8 Biconvex lens

One 50 x 66mm FL, Grade 2, Double-Convex Lens, one 55 x 87mm FL, Grade 1, Double-Convex Lens, and one 50 x 100mm FL, B270 Glass, Grade 1, Double-Convex Lens from Edmund Optics.

CHAPTER THREE – WORKING PRINCIPLES

3.1 Optical theory

3.1.1 Thin Lens Equation

Optical path can be mapped by drawing a ray diagram that traces the object through the lens to the image.

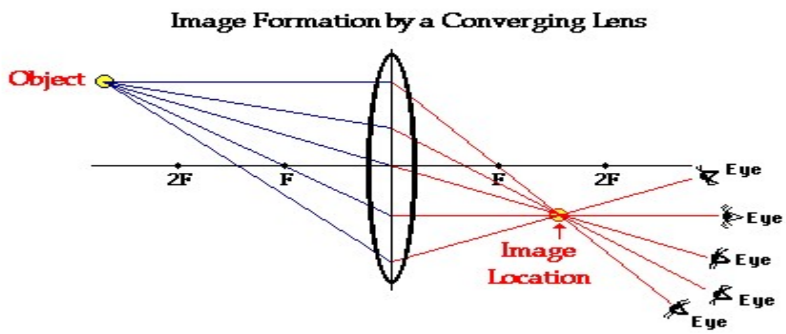


Figure 3.1.1.a Ray diagram for a convex lens

Figure 3.1.1.a shows an example of ray diagram of a convex lens, which will be the lens mainly use for this project. In the figure, one ray travels parallel to the principle axis, and converges through the focal point behind the lens. Another ray passes through the focal point in front of the lens, and travels parallel to the principle axis. A third ray passes through the optical center and have no bending due to it being a thin lens. The point where all 3 fundamental rays and other rays meet is where the image will be located.

The distance of object, u from the lens and its image distance, v through the lens of focal length, f is related by a general equation known as the thin lens equation which can be derived from Figure 3.1.1.a.

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

Equation 3.1.1.a thin lens equation

3.1.2 Image Formation

When the object is placed at different positions on the principle axis, images of different sizes and distances from the lens will be formed.

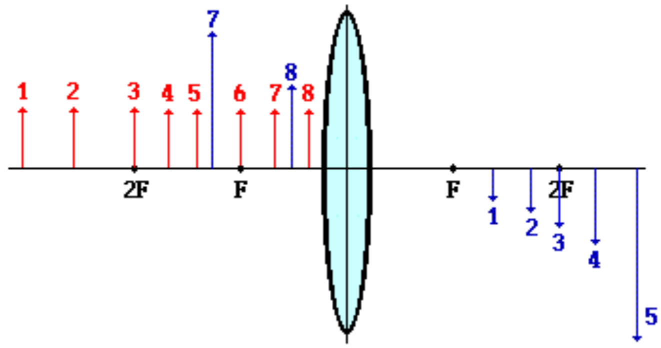


Figure 3.1.2.a Object-Image Relation for converging lens

(Objects are represented in red while images are represented in blue)

	Objects Image Relations		
Object Location	**Image Location**	**Image Characteristics**	**Magnification**
Beyond 2*f*	Between the 2*f* and *f* on the other side of the lens.	Inverted, reduced in size	Less than 1
At 2*f*	Located at the 2*f* point on the other side of the lens.	Inverted, image size are equal to object	Equal 1
Between the 2*f* and the *f*	Located in the specified region	Inverted, larger image	More than 1
At *f*	No image is formed		
Located in front of *f*	Located somewhere on the same side of the lens	Upright, enlarged	More than1

Table 3.1.2.a Explanation of diagram in Figure 3.1.2.a

Unlike converging lenses, diverging lens will only produce images with one kind of characteristics. Images produced are always virtual, upright and reduced in size regardless of location of the object, as shown in Figure 3.1.2.b below. Objects are represented in red while images are represented in blue.

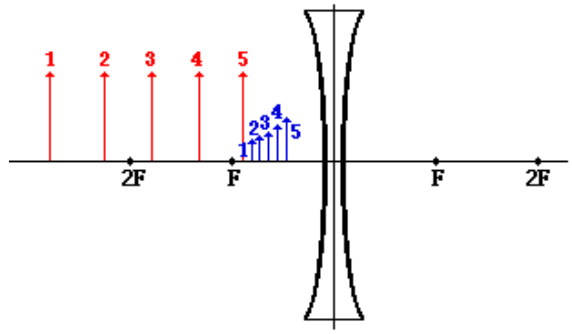


Figure 3.1.2.b Object-Image Relation of Diverging Lens

3.1.3 Magnification Factor

$$M = \frac{h_i}{h_o} = -\frac{v}{u}$$

Equation 3.1.3. a Magnification factor

The magnification equation relates ratio of image distance (v) to object distance (u), to ratio of image height (h_i) and object height (h_o). When magnification is positive, image has the same orientation as the object, and is virtual. When magnification is negative, the image is inverted and real. A magnification larger than '1' shows that the image is magnified, while magnification below '1' implies that the image is reduced. If magnification is '1', object and image are of the same size.

3.1.4 Three-Lens System

From the thin lens equation (Equation 3.1.1.a), the effective focal length of 2 lenses, with focal length f_1 and f_2, separated by a distance d_{12}, can be derived. It is given by

$$f_{eff2} = \frac{f_1 + f_2 - d_{12}}{f_1 f_2}$$

Equation 3.1.4.a Effective focal length of 2 lenses

The effective focal length of 3 lenses can be an extension of the above formula, by taking one of the focal length of the lenses to be the effective focal length of 2 lenses, thus, the formula will be:

$$f_{eff3} = \frac{\frac{f_1 + f_2 - d_{12}}{f_1 f_2} + f_3 - d_{23}}{\frac{f_1 + f_2 - d_{12}}{f_1 f_2} f_3}$$

Equation 3.1.4.b Effective focal length of 3 lenses

Where:

f_1, f_2, f_3 = focal length of lenses 1, 2, 3 respectively

d_{12} = distance between lenses 1 and 2

d_{23} = distance between lenses 2 and 3

3.2 Projector Optical Components

A projector's main function is to convert a variety of input signals and output a focused image onto a screen. The UC40 mini portable projector is a LCD projector that is backlit by a LED. Optical components in the projector includes the LED, reflector, LCD polariser, LCD, fresnel lenses, a mirror and the projector lens. While the projector has been modified to fit the project's application, working principle of the modified projector remains the same.

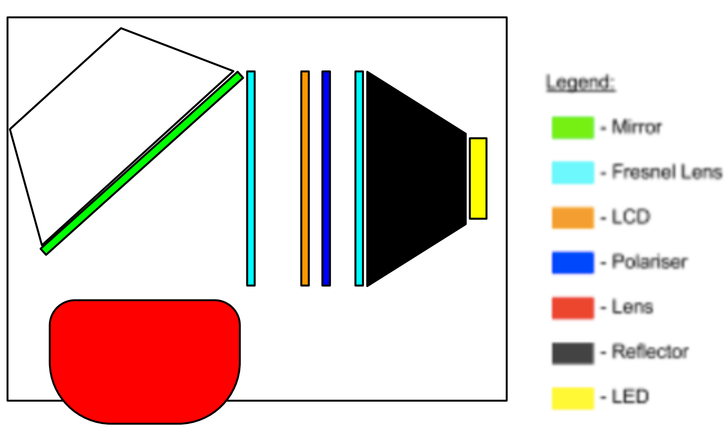


Figure 3.2.a Drawing of optical components in projector

3.2.1 LCD Panel

Figure 3.2.1.a Electronics processing unit and LCD panel

The projector utilises a simple projection system to display its image. It consists of a TFT (Thin-Film-Transistor) LCD which is a transmissive display that is illuminated by an external light source. Signals transmitted to the projector are processed by its electronics processing unit before being outputted by the LCD panel. Each pixel on the LCD consists of three subpixels: red, green and blue. Subpixels appear as a single colour to human eye as they are very small. When voltage is applied to the liquid crystals, they will twist and rotate polarised light. This electronically turns the coloured pixels on or off, and a full-coloured image is produced.

3.2.2 Polariser

The polariser is placed between at the back of the LCD panel, but in front of the LED. The polariser polarises light coming from the LED, so that light reaching the LCD panel is in only one direction.

3.2.3 LED Chip

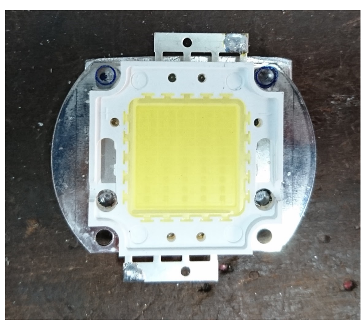


Figure 3.2.3.a 50W (5x10 cells) COB LED

COB (Chip-On-Board) LED consists of multiple LED chips packed together as a lighting module, and looks like a single lighting panel when lit. It is driven by DC (Direct Voltage) applied across its positive and negative terminals, and is able to achieve high intensity light while maintaining its compactness.

3.2.4 LED Reflector

Figure 3.2.4.a Reflector with a hole at the back for the light beam to shine through

A reflector is needed to gather and direct light beam forward, as angle of light beam from the LED is very wide. It is placed in front of the LED, and has a hole the size of the LED to allow light to pass through. The reflector has 4 reflective surfaces, so that light that enters from the back bounces off the surfaces and gets reflected to the front of the reflector. It helps to shape the beam, reduce light leakage and wastage, and increases light output.

3.2.5 Fresnel lens

Fresnel lenses have concentric grooves on one side of the lens that act as individual refracting surfaces. Depending on the orientation, fresnel lenses are able to collect or collimate light. The fresnel lens used in front of the reflector collimates light, as light rays hit the smooth side of the lens first. It also acts as a diffuser to diffuse light evenly and reduce hot spots when illuminating the entire LCD panel.

Figure 3.2.5.a (left) Fresnel lens collimating a point source

Figure 3.2.5.b (right) Fresnel lens collecting collimated rays

The second fresnel nearer to the mirror, which grooved surface is facing the incoming light rays, converges light towards the mirror. Thus, a converging beam is reflected off the mirror into the lens. This helps to make sure the entire image will be reflected into the lens, and the sides of the image will not be cut out, as the lens diameter is shorter than size of the LCD.

3.2.6 Mirror

A mirror is placed in front of and at an angle to the LCD so as to increase object distance between the LCD and the projector lens. This helps to keep the projector compact by removing reliance on physical distance between LCD and lens. The mirror reflects incoming light into the lens.

3.2.7 Projector Lens

Projector lenses are mostly used to have more control over the image size and distance. The lenses help to enlarge the image, so that the LCD size can still be kept small, and thus reducing the projector's size. Projector lenses can also help to throw the image at a certain distance, depending on the application and type of lens. The lens case is mounted on a barrel that can be adjusted to change the image size, which also affects the image distance. The projector or the screen then has to be manually shifted to the new image distance.

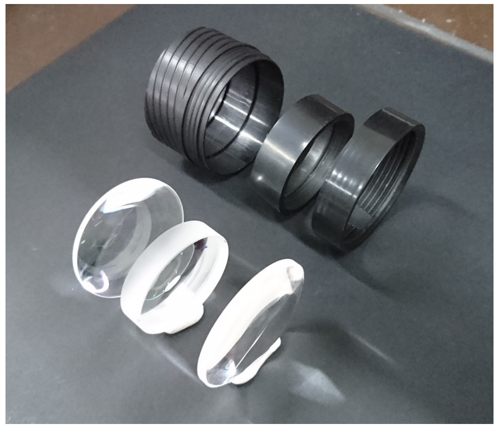


Figure 3.2.7.a Projector lens consists of 3 lenses: two plano convex lenses and a biconcave lens

The modified projector lens consists of three lenses: a biconcave lens in the middle, a plano convex lens as the inner lens while the outer lens is a biconvex lens. The first convex lens converges light ray which focuses behind the concave lens, since the convex lens' focal length is longer than distance between the lenses. The concave lens diverges the virtual object and widens the light beam. This wider light beam then passes through the third lens and converges, forming the final, real image on a screen.

3.3 Python Programming

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

With Raspberry Pi, Python helps integrating the project to the real world.

Figure 3.3.a Python in Raspberry Pi

3.4 OBD II system

On Board Diagnostics-II (OBD-II) is a vehicle network for the diagnosis and checking of a vehicle. Produced cars are equipped many sensors for measurement and regulation which are controlled by the Electronic Control Unit (ECU). As advances in vehicle and computer technologies, all parts of the vehicle are controlled by the ECU including the drive system, braking system, engine system and steering system. Among them, the engine is the most significant part of the vehicle, main features such as ignition timing, idle threshold setting, variable valve timing and fuel injection requires a precise control system. OBD-II provides access to data from the engine control unit (ECU) and offers a valuable source of information when troubleshooting problems inside a vehicle.

All light duty vehicles less than 8,500 pounds sold in North America since 1996 are required to equipped with the OBD-II system and the medium duty vehicles up to 14,000 pounds must also be equipped with OBD II systems beginning in 2004. In addition, since 2010, all heavy-duty vehicles over 14,000 pounds are required to support OBD-II diagnostics as well.

The SAE J1962 specification standardised two hardware interfaces for OBD called type A and type B. Both interfaces are female, 16-pin (2x8), D-shaped connectors. Among the 16 pins, the function of 7 pins are specified in the SAE J1962 including pin4 for chassis ground, pin 5 for signal ground, pin 16 for battery voltage and pin 2, 6, 7, 10, 14, 15 for reading and writing using different signal protocols. The rest of coins are left for manufacturer discretion.

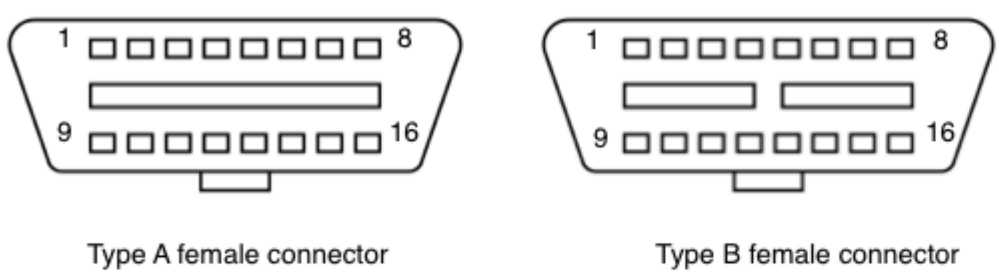


Figure 3.4.a Two types of hardware interfaces for OBD

Currently, there are five OBD-II signal protocols are in use: SAEJ1850 PWM(pulse-width modulation)、SAEJ1850 VPW(variable pulse width)、ISO14230 (KWP-2000) 、ISO9141 、ISO15765-4/SAEJ2480(CAN). The table below shows the properties of each protocols.

	SAE J1850 PWM	SAE J1850 VPW	ISO 9141-2	ISO 14230 KWP2000	ISO 15765 CAN
Data Rate	41.6 kbit/s	10.4 kbit/s	10.4 kbit/s	1.2 to 10.4 kbit/s	250 kbit/s or 500 kbit/s
Utilised Pin	pin 2 : Bus + pin 10: Bus -	pin 2: Bus +	pin 7: K-line pin 15: L-iine	pin 7 : K-line pin 15: L-line	pin 6 : CAN High pin 14 : CAN Low
Voltage Level	Hight voltage : +5V	High voltage : +7V Decision point: +3.5V	K-line idles high	High voltage: +12V	CANH: 3.5V CANL: 1.5V
Message Length	12 bytes, including CRC	12 bytes, including CRC	max 260 bytes (data field max 255)	data field max 255	Base format: 11 bits Extended format: 29 bits

Table 3.4.a Comparison of five signal protocols

A list of On-board Diagnostics Parameter IDs(PIDs) were defined by the SAE J1979 standard. PID codes are used as a diagnostic tool to request data from the vehicle. Technicians can use PID with a scan tool to communicate with the controller area network of a vehicle such as CAN-bus. When a PID code is reported by the scan tool , the controller will recognise the PID report the value for that PID to the bus. Then the scan tool reads the response and displays it to the technician. Table below shows the 10 modes of operation described in the latest OBD-II standard SAE J1979. Each mode contains multiple commands to control specific sensor.

Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

Table 3.4.b PID mode description

3.5 Communication principle of ELM327

The scan tool used in this project is an ELM327 cable. To communicate with ELM327, our software program need to be set properly. A proper setting includes a proper baud rate (9600 bauds if pin 6 = 0V or 38400 bauds if pin 6 = Vcc), 8 data bits, no parity bits, 1 stop bit and the correct port.

There are two command sets supported by ELM327, one is the PID command set which is used to communicate with a vehicle or the OBD-II simulator, (for example, the OBD command '0100' requests the information of availability of PIDs [01 - 20] in Mode 1, in which '01' indicates Mode 01 and '00' means PID 00). Another kind of command is the Hayes AT command set which is considered as the internal commands, (for example, AT command 'atz' resets the ELM327 chip and all setting are returned to their default values.) Because ELM327 is not case-sensitive and will ignore the spacing so the commands 'AT Z', 'atz', and 'AtZ' are the same to an ELM327 chip.

The response to OBD command received from ELM327 are hexadecimal digits in pairs. The first 4 bits will repeat the command and the rest of data is the requested data from OBD. Because of the response echo the command, the mode value in the response would be added with 40 to distinguish with a command. For example, the response to “0100” may be “41 00 BF 9F B9 90’’. The first byte ’41’ represent Mode 01 and “00” represent PID 00. The rest 4 bytes in digital bits are a series of 0(not supported) and 1 (supported) to indicate whether the correspond PID is supported.

3.6 Seven- segment Four - digit LED display

Seven Segment 4 digits LED was used to display the numbers which would be be reflected on the glass. Thus, the digits displayed on the LED look like figure below.

Decimal	7 Segment Display						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	0	1	1	0	1	1
3	1	1	1	1	0	0	1
4	0	1	1	0	1	0	1
5	1	1	0	1	1	0	1
6	1	1	0	1	1	1	1
7	0	1	1	1	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	1	0	1

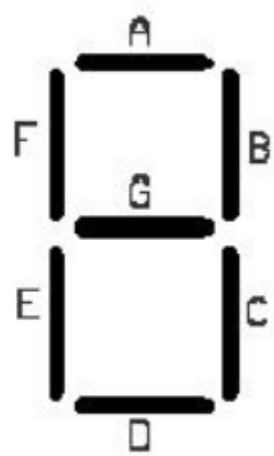


Figure 3.6.a Seven-segment display for reflection

3.7 Terminal Control

A terminal emulator is a program that allows the use of the terminal in a graphical environment. As most people use an OS with a graphical user interface (GUI) for their day-to-day computer needs, the use of a terminal emulator is a necessity for most Linux server users. Each terminal emulator has its own set of features, but all of the listed ones work great and are easy to use.

In a Linux system, the *shell* is a command-line interface that interprets a user's commands and script files, and tells the server's operating system what to do with them. There are several shells that are widely used, such as *Bourne shell* (sh) and *C shell* (csh). Each shell has its own feature set and intricacies, regarding how commands are interpreted, but they all feature input and output redirection, variables, and condition-testing, among other things.

3.8 Microcontrollers

3.8.1 Arduino

Arduino is an open-source prototyping platform on easy to use hardware and software. By connecting the pins on board to specific devices, the Arduino can operate many different tasks due to user’s programming. Arduino language is based on C.

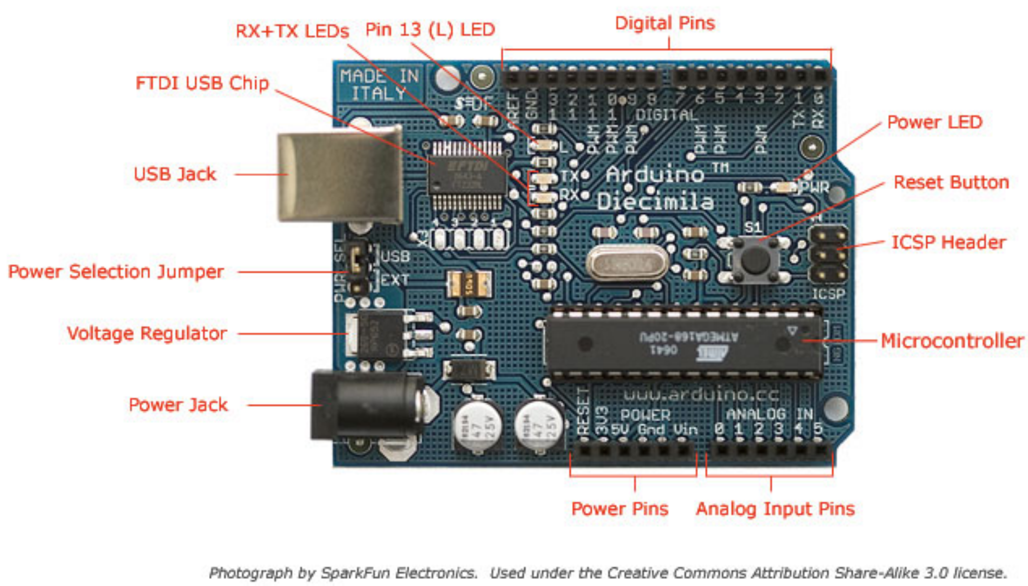


Figure 3.8.1.a Arduino board configuration

In this project we manage to build an Arduino which can control two boards of 4-digits-7-segments LED simultaneously, displaying the data captured from OBD simulator

3.8.2 Raspberry Pi

Raspberry Pi functions as a small-sized CPU, which plugs in a monitor or projector and uses a standard keyboard and mouse. It has the ability to interact with the outside environment through many different channels (USB/HDMI/Ethernet cable/pins), which greatly simplify and enhance the connection between devices in our project.

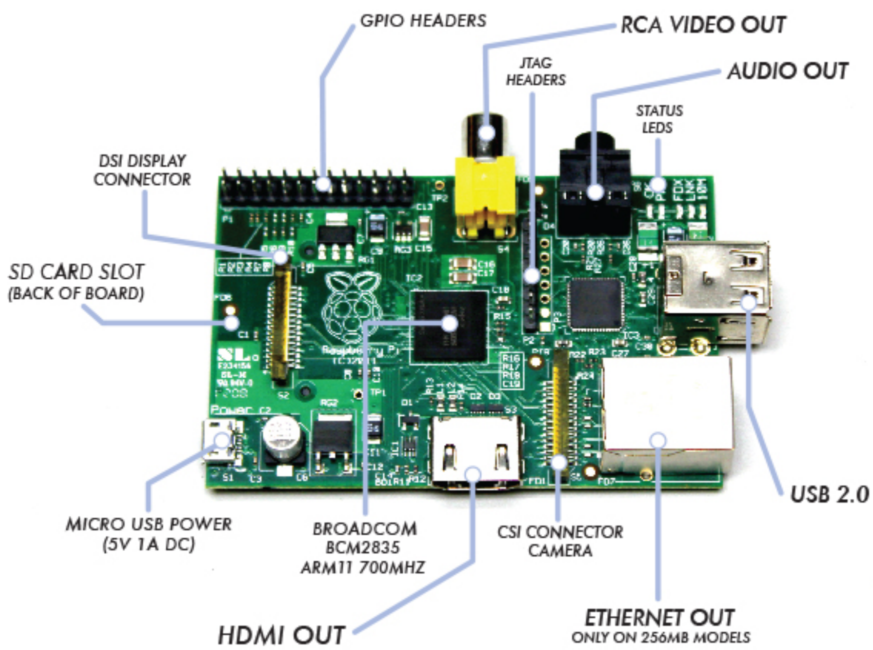


Figure 3.8.2.a Arduino board configuration

In this project we use Python as the language to build our main program, including capture, process and display. The user interface is constructed using wxPython, a GUI toolkit for Python.

CHAPTER FOUR – PROJECTOR MODIFICATION

4.1 Research & Planning

4.1.1 Projector Types

Comparison of the different types of projectors can be seen in the table below:

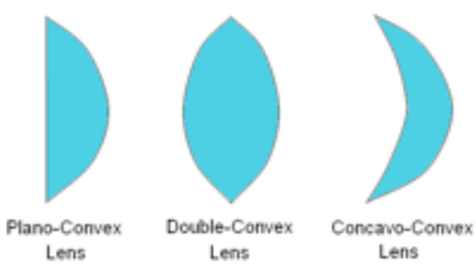
	Advantages	Disadvantages
LCD	1. Higher contrast 2. Sharper image	1. Colour decay: display an amount of yellowish or greenish 2. Limited lifetime 3. Screen door effect: distracting lines between the pixels 4. Dead pixels: when one of the pixel becomes permanently on and off
DLP	1. Higher contrast 2. Less pixilation 3. Filter free design	1. Rainbow effect
LED	1. Longer life for lamp 2. Energy efficient light source	1. More expensive, price per lumen

Table 4.1.1.a Advantages and Disadvantages of LCD, DLP, LED

LED-typed projector is significantly better in terms of lamp efficiency and performance.

4.1.2 Lens Types

There are different types of lenses such as converging (convex) and diverging (concave) lenses. For the project's application, a real image must be formed so that the image can be projected onto a screen and can be viewed. This application requires converging lenses which can produce real images. Converging (convex) lenses have surfaces that curve outward and are thickest at the centre. Spherical, double convex lenses are cheaper to implement than lenses such as as aspherical lens.



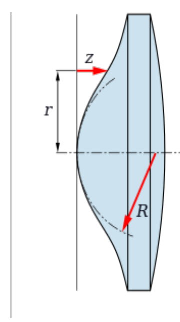


Figure 4.1.2.a (left) Different types of converging lens

Figure 4.1.2.b (right) Aspherical lens

4.1.3 Multiple-lens System

Multiple lens system was used as a method to correct spherical aberration. There are many ways to correct spherical aberration: lens splitting, power splitting, cementing etc. The method used in the project were a system of 3 lenses combination of both positive and negative, i.e. convex and concave, separated by some distance. This multiple lens system is able to minimise spherical aberration significantly. To better understand this, the original lens of the projector was dismantled and studied. The setup is as shown in Figure 4.1.3.a:

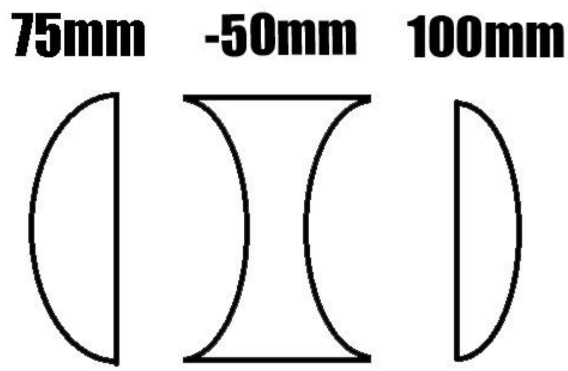
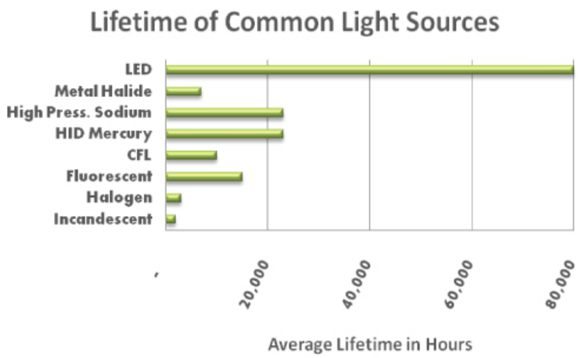


Figure 4.1.3.a Original lens system of projector UNIC UC40

The original lens system consisted of 2 plano-convex lenses on each end, with focal length 7.5cm and 10cm respectively. A biconcave lens of -5cm focal length sits in the middle.

4.1.4 Light Source

Brightness of image can be improved by changing the optical components in the projector. Light loss or absorption can be reduced by improving quality or specifications of components such as the fresnel lens however, the most straightforward method is to change the LED to a light source of higher intensity rating. As compared to other light sources commonly used in projectors, such as metal-halide lamps, LEDs are very energy efficient are able to output high intensity per watt.



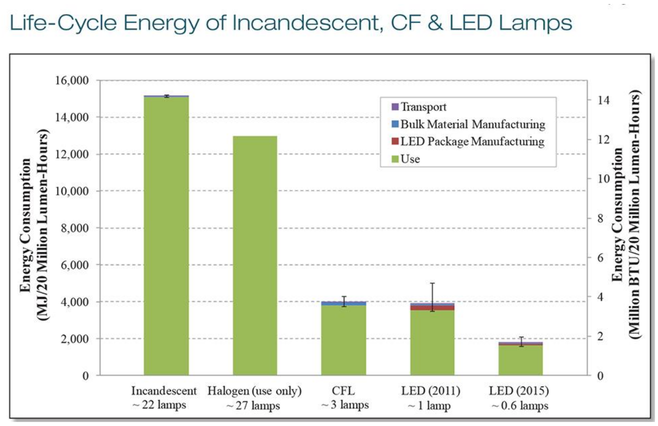


Figure 4.1.4.a (left) Comparison of lifetime of LED and other light sources

Figure 4.1.4.b (right) Comparison of energy consumption of LED and other lamps

LEDs are very directional, compared to lamps which emit multi-directional light, and do not have much light wastages. They require no maintenance and do not need warm up/down time thus do not risk getting damaged easily. As LEDs have generally long lifespan, they do not need to be replaced during the lifetime of the projector. They are also small in size and lightweight, thus are able to be integrated into smaller projectors better. LEDs are also relatively inexpensive. Therefore, LED is chosen for the project's application, however, a higher specification LED will have to be used.

COB LED was also chosen over other kinds of LEDs as the COB LED does not have to be integrated in the PCB (Printed Circuit Board), and can be easily swapped out or reattached to a heat sink. It also has higher uniformity and superior output performance over other LEDs.

Figure 4.1.4.d Comparison of different LEDs

Neutral white (yellow phosphor) was selected over warm white (orange-yellow phosphor) as it will more accurate when reproducing colours.

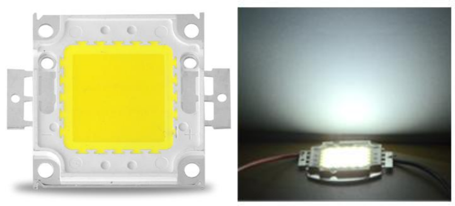
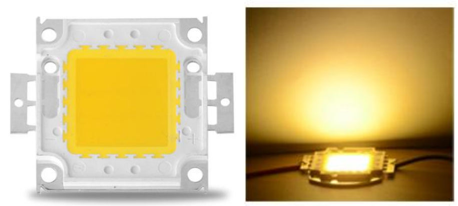



Figure 4.1.4.e (left) Neutral white COB LED when lit

Figure 4.1.4.f (right) Warm white COB LED when lit

The original $24W$ (8x3 cells) COB LED light was to be replaced by a $50W$ (10x5 cells) COB LED. Higher wattage LEDs are commonly manufactured in 50W or 100W however the 100W LED would need a highly efficient cooling system which would take up too much space in our application, and also need to draw a large amount of power. Therefore, a $50W$ LED would be a better compromise for performance and efficiency.

4.2 Modifications in Optics

In order to achieve a image distance between 10cm to 30cm, either focal length of the projector lens has to be shortened, or object distance has to be increased. In practical sense, the lens of the projector can be moved away from the LDC, effectively increasing object distance, *u* or changing the lens of the projector to one of lower focal length.

4.2.1 Increasing object distance *u*

Experiment confirmed the hypothesis, image distance was reduced significantly when lens were moved away from LCD. However, the lens were positioned 6cm away from the projector, thus creating an unnecessarily bulky design, not to mention the lack of space efficiency.

4.2.2 Measurements & calculations to get suitable focal lengths

The simplest approach to reduce focal length *f* of the projector's original lens is to replace the entire lens with a single biconvex lens of suitable *f*.

Firstly, measurements of the projector were made. Object distance from the LCD to the projector lens is 10.8cm, and from the LCD to the front of the projector is 14cm.

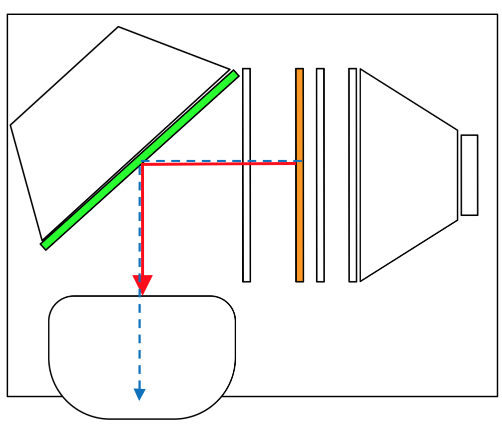


Figure 4.2.1.a Red arrow represents distance to projector lens (10.8cm),

Blue dotted arrow represents distance to front of projector (14cm)

Calculations were made using thin lens equation to obtain a range of f for the new lens. As the single lens would be positioned near the front of the projector, object distance u is taken as 14cm, represented by the blue arrow in Figure 4.2.1.a (above). By varying image distance v, the following table was obtained:

v (cm)	10	15	20	25	30
f (cm)	5.8	7.2	8.2	9	9.6

Table 4.2.1.a Range of single lens f obtained through Thin Lens Equation

The new focal length required is in the range of 5.8 to 9.6. Since custom made lenses were out of budget, a set of 3 biconvex lenses of f = 6.6cm, 8.7cm and 10cm were purchased. All three lenses were of different grades of glass.

4.2.3 Lens Prototyping

The single lenses were tested at different positions to find the shortest image distance, while maintaining a sizeable image (A5 size, 25.7cm) and a feasible u between 10cm to 30cm.

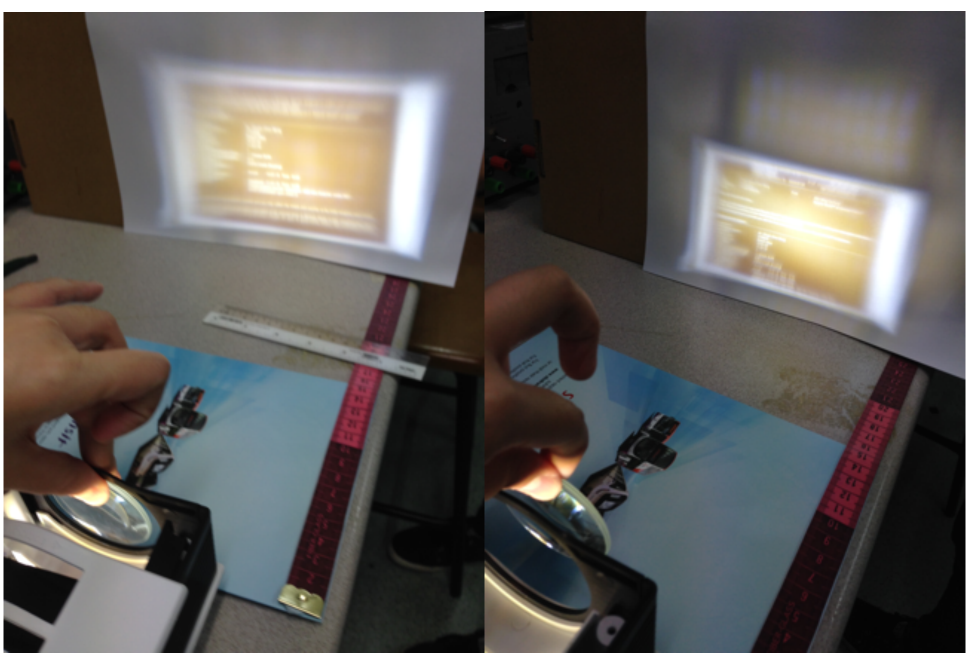


Figure 4.2.3.a A single lens tried at different positions and its resultant image

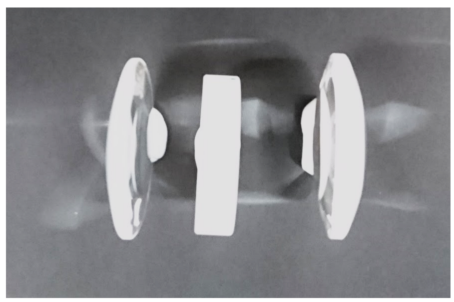
The following observations were made from the tests:

1) If the lens were placed further inside the projector, the image distance *v* increased, and the image size increased. This agreed with the thin lens equation, i.e. v increased when u decrease, and as a result, magnification factor increased. As long as *u* was between *f* and *2f*, the image will be magnified.
2) If the lens were placed further from the projector, image distance *u* is increased, and the image size decreases. Since *u* was still between *f* and *2f*, image will still be magnified.
3) Ideal image size can only be achieved with lenses of focal length *f* 8.7cm and 10cm. Due to the design of the projector, lens with f = 6.6cm cannot be moved into the projector for *u* to be close enough to *f*, to get a image that was large enough.
4) For lens f = 8.7cm, it had to be placed around 1cm inside the projector to get an image of about A5 size, and v = 27cm
5) For lens f = 10cm, it had to be placed around 1cm outside of the projector, v = 28cm
6) Different grades of glass have little visible effect on the quality of image. This may be because the experimental grade lenses had less refined glass no matter which grade of glass, and had no anti reflection coatings.
7) Image formed was reasonably clear at the center, but became increasingly out of focus nearing the edge.

To conclude the test, research was done to theorise why the resultant image is unfocused away from the center. It was found that single lenses are not refined enough to get images with even sharpness throughout. Spherical aberrations occur due to the spherical nature of the lenses. Light rays passing through the lens may not converge at the same point due to the varying thickness of the lens from its centre to its edge, thus, refracting light rays to varying degrees. This aberration may be corrected with an aspheric lens, however aspheric lenses are expensive and would thus make the prototype not cost-effective. The aberration however may be corrected with a simple multiple-lens system which would be cheaper to implement. The multiple lens system used a combination of positive and negative lenses (a.k.a. convex and concave lenses) separated by some distance. This setup was found to correct spherical aberration significantly.

4.2.4 Experiment and measurement to obtain focal length

Analysis of the original projector lens was done to understand how it was able to project an image that has consistent sharpness. The original lens consisted of three lenses: two plano convex lenses on both ends while a biconcave lens sat in the middle. An experiment was conducted to find *f* for each lens, in order to understand more about the multi-lens system used.



Plano convex Lens 1 (left), Biconcave Lens 2 (middle), Plano convex Lens 3 (right)

Figure 4.2.4.a Rough representation of lenses and their positions relative to each other

A collimated ray beam (laser) was shone through each lens at various horizontal positions. Path of the resultant rays were then marked out.

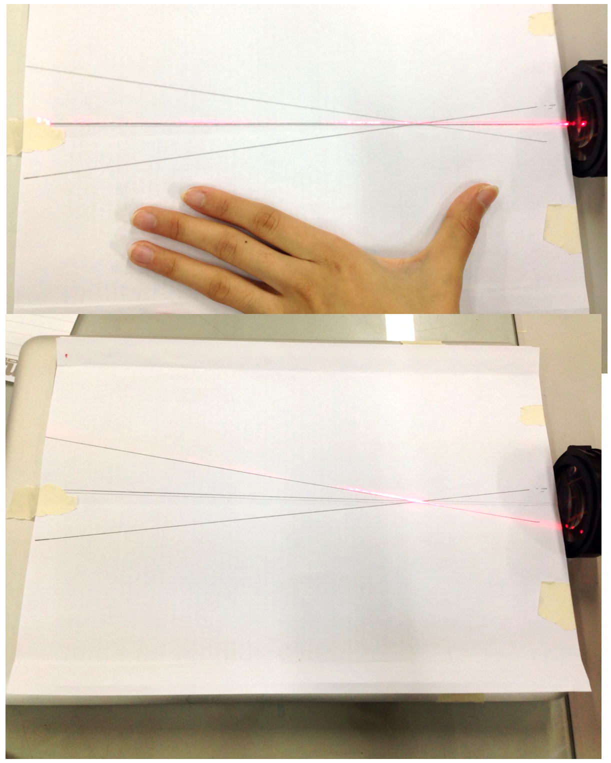


Figure 4.2.4.b (left) Laser beam shone through center of lens goes in a straight path

Figure 4.2.4.c (right) Laser beam shone on the left of the lens bends towards the right

A point where all rays converge can be found, after obtaining at least 3 paths.

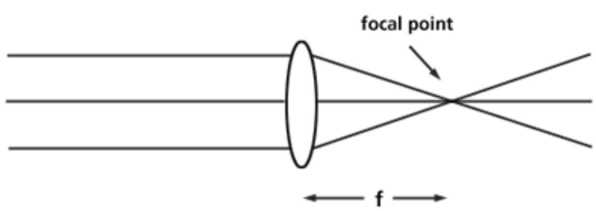
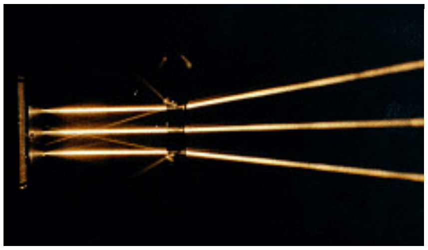


Figure 4.2.4.d shows a schematic drawing of a similar experiment

For concave lenses, light rays behind the lens diverge, and the virtual image is formed in front of the lens instead. By tracing the diverging rays backwards, a point where all rays converge can be found. The focal length of the concave lens is the distance from that point to the lens.



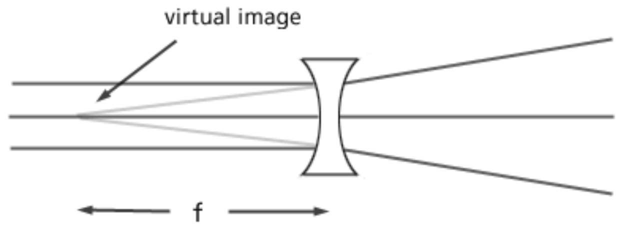


Figure 4.2.4.e (left) shows a similar experiment where collimated light rays are shone through a concave lens, and rays diverge behind the lens

Figure 4.2.4.f (right) shows a schematic drawing of the experiment

Focal length f of each lens were found to be 7.5cm, -5cm and 10cm respectively. Measurements of thickness and distances between each lens were also made with vernier calipers for precision.

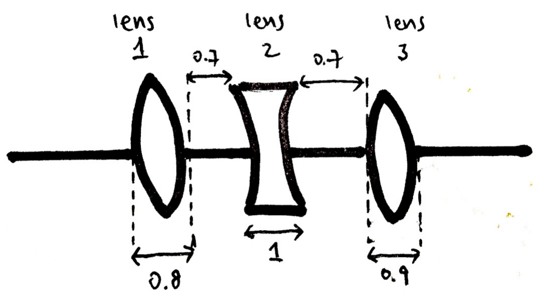


Figure 4.2.4.g Rough drawing of measurements of lenses

As such, the 3-lens system can be understood to do the following. The first lens converges light rays that focuses behind the second lens, since the convex lens' focal length is longer than distance between the lenses. The concave lens diverges the virtual object and widens the light beam. This wider light beam then passes through the third lens and converges, forming the final, real image on a screen.

4.2.5 Calculation for 3-lens

To achieve the desired image distance and size, using the available lenses and the three-lens system equation, different combinations of lenses were found to obtain a reasonable effective focal length of 5.8 - 9.6cm. All lenses were spaced 1.6cm from each other, and the object distance found was 10.8cm.

Lens 1 (cm)	Lens 2 (cm)	D_{12} (cm)	Lens 3 (cm)	D_{23} (cm)	EFL (cm)	v (cm)
6.6	-5	1.6	8.7	1.6	8.7	44.7
6.6	-5	1.6	10	1.6	10	135
7.5	-5	1.6	6.6	1.6	7.5	24.5
7.5	-5	1.6	8.7	1.6	10.5	378
7.5	-5	1.6	10	1.6	12.5	-79.4
8.7	-5	1.6	10	1.6	16.8	-30.2

Table 4.2.5.a Different combination of lenses, their EFL and resultant image distance v

The combination of f = 7.5cm, -5cm and 6.6cm results in an effective focal length of 7.5cm were chosen. The rationale behind this combination is the position of new lens system is at approximately the same position as the original lens, while its throw distance is reduced to a mere 24.5cm.

4.2.6 Second lens prototyping

The position of the outer lens of 6.6cm was readjusted. That lens was shifted outward by 0.4cm till D_{23} = 2cm resulted in an image magnified by 2.1 times of the original LCD panel size, at an image distance of around 23.5cm. The new EFL is 7.4cm. The image projected from the new lens system was found to be quite clear, except at the edges of the image. This may be due to the quality of the lens of 6.6cm focal length at the extreme outer diameter of the lens.

Lens 1 (cm)	Lens 2 (cm)	D_{12} (cm)	Lens 3 (cm)	D_{23} (cm)	EFL (cm)	v (cm)	Magnification M
7.5	-5	1.6	6.6	2	7.4	23.5	-2.1

Table 4.2.6.a Specification of new three-lens system

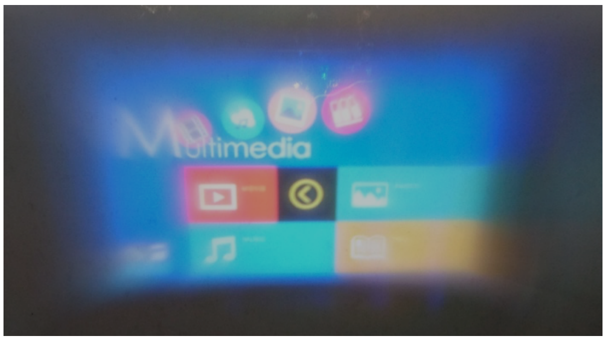


Figure 4.2.6.a (left) Image projected through single biconvex lens
Figure 4.2.6.b (right) Image projected through new modified 3-lens system

Image quality has been greatly improved through the modified new 3-lens system, as compared with image obtained with a single lens, which was covered in Chapter 4.2.3 Lens Prototyping. Initially, only the middle of the image was clear enough to be legible. Through the new lens system, even sharpness was achieved across almost the entire image, and words and images projected are very clear.

4.3 Improvements in illumination

Image illuminated by the original $24W$ light source is not very visible under ambient lighting when projected onto the special film for glass. Therefore, intensity of the light source needs to be improved.

4.3.1 Comparison of LEDs

A $50W$ COB LED was selected as the most suitable light source replacement, after doing research and making comparisons in Chapter 4.1.4. Brightness of both LEDs are firstly compared in a simple test. A multi-meter was used to measure the voltage across the $24W$ LED when it is driven by the projector's internal circuit, and the $50W$ LED when it is driven by its own 1.5A constant current, 40V DC LED driver. Currents were also measured, and using the power law $P = IV$, power drawn can be calculated.

LED	Voltage (V)	Current (A)	Power (W)
$24W$	19.45	1.7	33
$50W$	30.2	1.5	45.3

Table 4.3.1.a Measurements and calculations for 24W and 50W LED

Each LED was then secured in a black box, and a lux-meter was attached directly opposite the LED. When the LED is switched off, the reading on the lux meter is 0. A DC voltage generator was used to supply power for both LEDs while current was maintained at 1.5A for both.

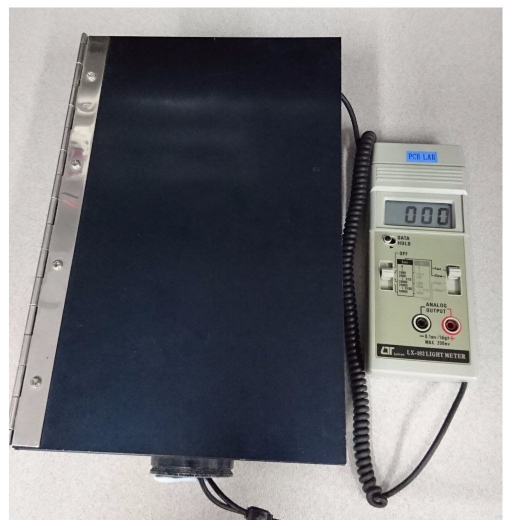


Figure 4.3.1.a (left) Setup of lux measurement, LED turned off, ambient lighting recorded as '103 lx'

Figure 4.3.1.b (right) Black box when closed, no light can enter

LED	Voltage (V)	Power (W)	Lux (lx)
$24W$	19	28.5	21500
$50W$	30.3	45.45	35000

Table 4.3.1.b Power drawn by each LED and light output in lx

Lux measures luminous flux per unit area (lm/m^2). As the area is constant, the $50W$ LED's power of light (lm) is around 1.67 times stronger than the $24W$ LED, and therefore is brighter.

4.3.2 Integrating LED

The new LED was much bigger than the original LED. Therefore, modifications had to be made to the projector's internal casing to make space for the $50W$ LED. Some of the plastic casing had to be sawed away, and an aluminum plate was added between the LED and the heatsink to improve contact for heat dissipation. A few more IC chip heat sinks (small heat sinks) were also added behind the main heat sink as the LED would heat up very rapidly. After fitting the new LED into the projector, a new reflector had to be constructed as the projector's original reflector is too small the $50W$ LED. Space available between the polariser and the LED was first measured out, and a prototype was first designed on paper before being transferred onto thin acrylic sheets. Silver reflective paper was then pasted onto the acrylic, and the four pieces were glued together.

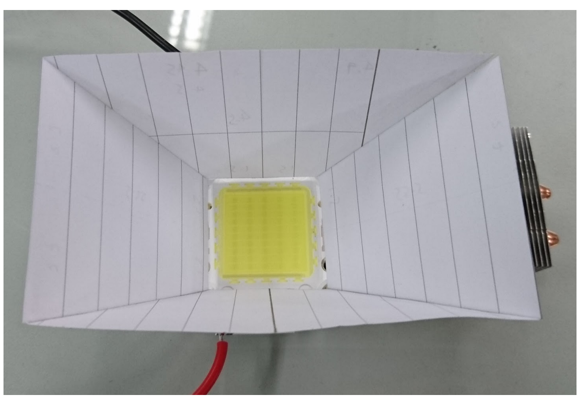
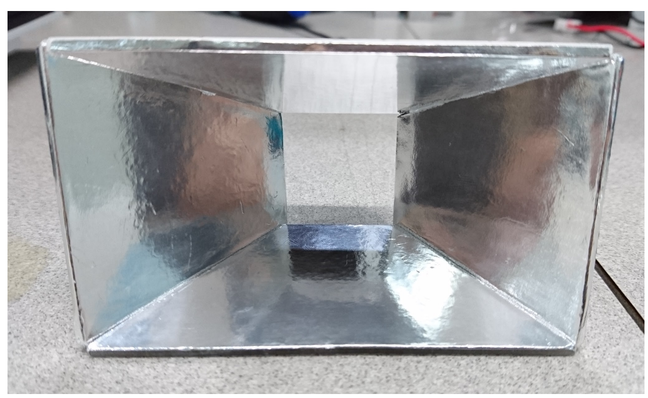



Figure 4.3.2.a (left) Paper prototype tested with 50W LED

Figure 4.3.2.b (right) Final prototype for reflector

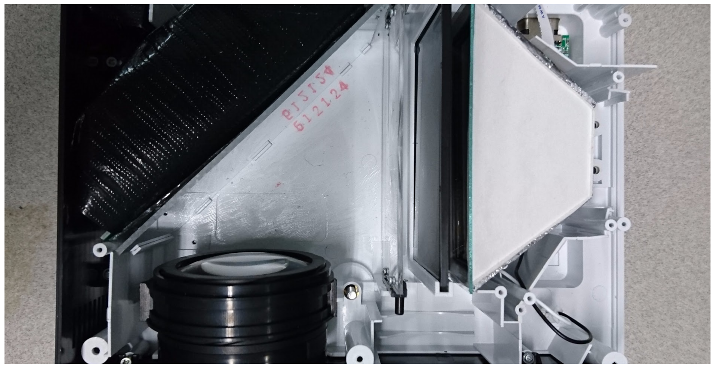
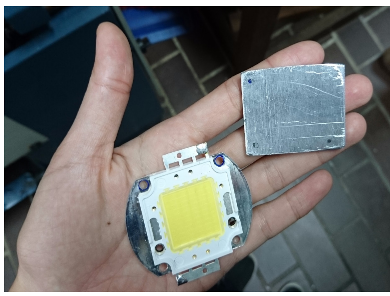


Figure 4.3.2.c Case fitting of reflector final prototype



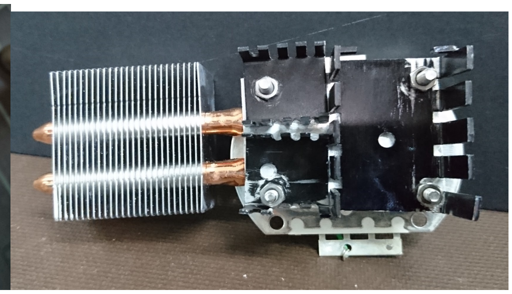
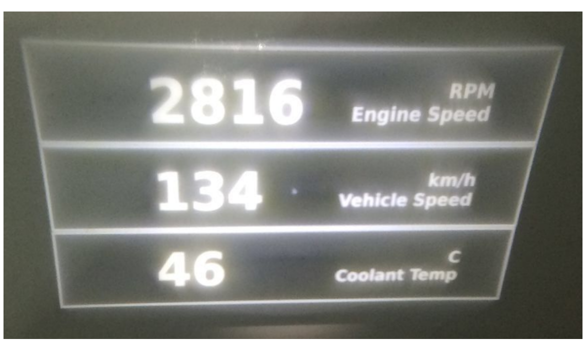


Figure 4.3.2.d (left) LED and aluminium plate

Figure 4.3.2.e (right) Back view of LED mounted onto heatsinks

The new LED and heatsinks fitted well into the projector casing together with the reflector. Both components were placed at relatively similar positions to their original counterparts, however, the depth of the reflector was made to be shallower than the original in order to accommodate the size and of the new LED. The new LED and reflector illuminated the LCD panel well, and image projected onto a screen is very bright. However, the intensity is still reduced when projecting onto glass.



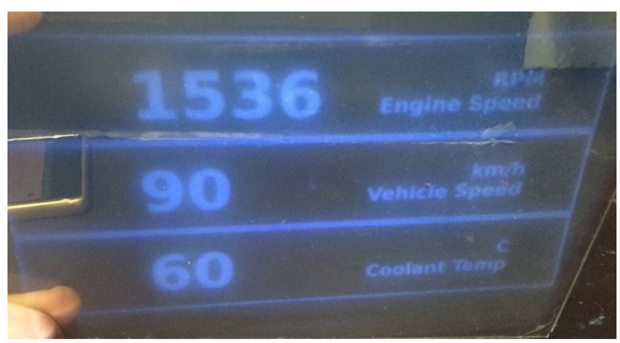


Figure 4.3.2.f (left) Image projected onto screen, illuminated by the new LED
Figure 4.3.2.g (right) The same image projected onto glass

CHAPTER FIVE – OBD DATA PROCESSING

5.1 Two approaches for glass projection

There were basically two approaches to project the information to the glass. First approach was reflecting the image from LED broad to the glass. During the project demonstration, data was transferred to two 7 segment 4 digit LED display broad and controlled by the Arduino Uno. To make it reflect the correct number, the original data must be converted into a special order. The relationship between normal numbers and converted numbers have been discussed in the previous chapter.

The other approach was projecting the image from a projector directly. It provided the flexibility since nowadays the advanced projectors could received the graphic information from laptops, smart phones and etc. through HDMI cable and USB. In this project, Raspberry Pi MCU was selected to be connecting the modified projector as the source of information. In this way, the data could be acquired and processed on the Raspberry Pi platform first, and then transferred to the projector and displayed accordingly.

Figure 5.1.a Projected image on the blackboard

From previous section, it was shown that two different approaches were used to achieve the data projection on the glass (screen). Since the MCUs and display media were different, the construction of the systems was different as well. Below show the structures of each approach.

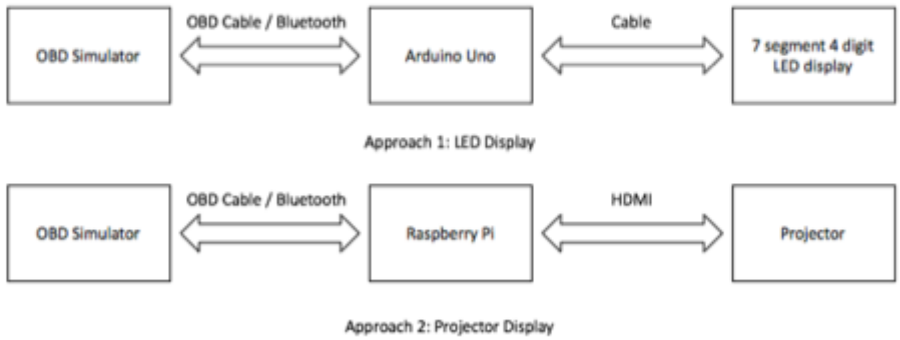


Figure 5.1.b Two approaches of data display

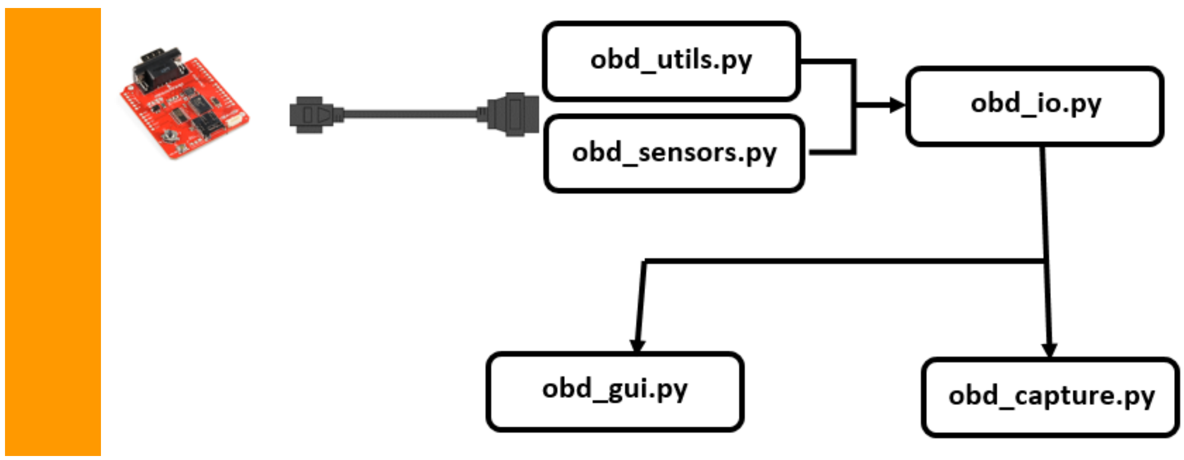


Figure 5.1.c Approach using Python on Raspberry Pi

5.2 Arduino

5.2.1 Single board display

This is an example of how to drive a 7 segment LED display from an UNO R3 with the use of current limiting resistors. Each segment in the display module is multiplexed, meaning it shares the same anode connection points. And each of the four digits in the module have their own common cathode connection point. This allows each digit to be turned on or off independently. Also, this multiplexing technique turns the massive amount of microcontroller pins necessary to control a display into just eleven or twelve (in place of thirty-two).

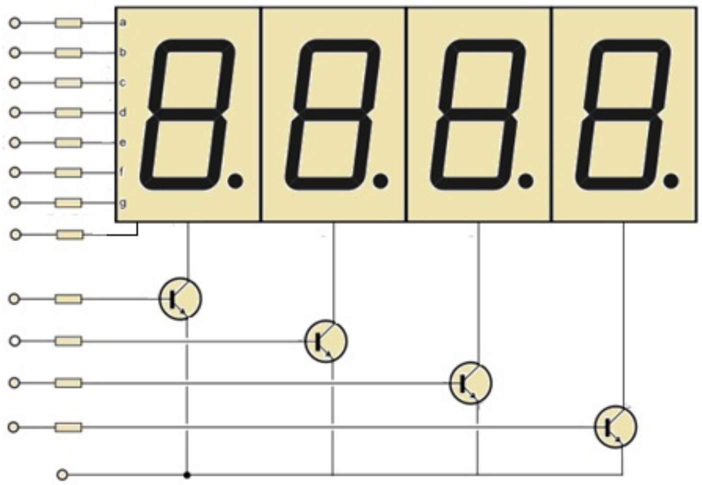


Figure 5.2.1.a Electric circuit of Seven-segment four-digit LED

This 4 -digit 7-segment display section is wired around four common-cathode 7-segment LED displays, and four BC547 npn transistors. The 1K resistors are used for base current limiting, and the 390R resistors limits the operating current of the LED display segments.

Figure 5.2.1.b Seven-segment four-digit LED pins number

In the Arduino board, digital outputs are wired due to this circuit:

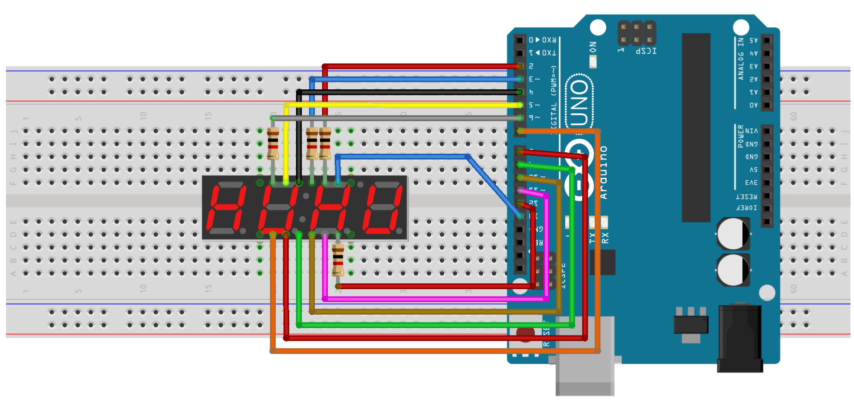


Figure 5.2.1.c The connection between Arduino Uno to 7-segment 4-digit LED

5.2.2 Double boards display

For displaying the data using LED, a system was constructed including of 1 Arduino controlling 2 boards of 4-digits-7-segments. It applies the same logic of 1 board control but extend to 8 digits, using 4 analog pins A0-A3 as extra pins for controlling extra 4 digits.

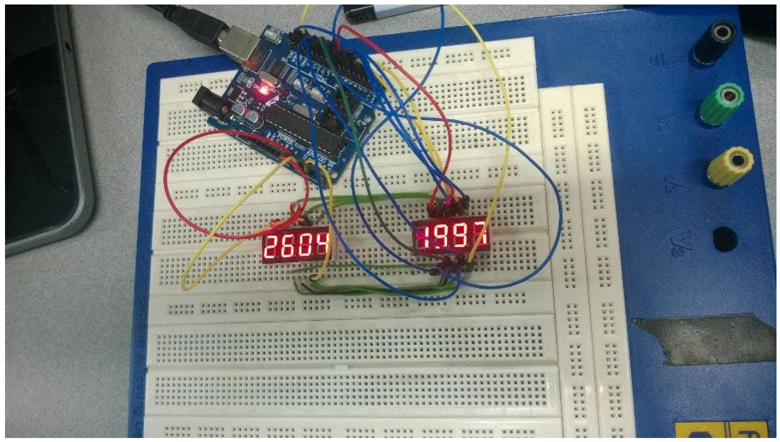


Figure 5.2.2.a Data displaying with 7-segment 4-digit LEDs directly

The displayed numbers can also be modified to mirrored-type, which show the correct version through reflection.

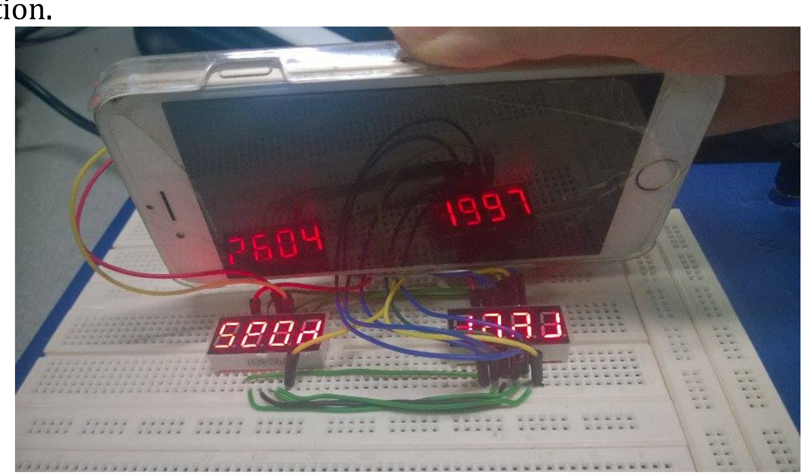


Figure 5.2.2.b Data displaying with the reflection of 7-segment 4-digit LEDs

5.3 Raspberry Pi

5.3.1 Data acquisition

Data acquisition could be divided into three steps. Firstly, the obd_io.py file defined and sent the command which contain the hex numbers from the device to the cable. Secondly, in the obd_sensor.py, hex values were converted back to the decimal values presenting different meaning data. Sensor names and units were also defined in this file. Last but not least, the obd_utils.py files enabled the Bluetooth and USB connections. It allowed the data to be transferred to the external devices.

5.3.1.1 obd_utils.py
This file searches for the available ports and find the serials who pass the data between the microcontrollers and the scan tools.
5.3.1.2 obd_sensor.py
This file defined a series Sensors to assign each PID command to its correspond sensor name, value function and unit for displaying. Moreover, the DTC codes are also defined in 'obd2_codes.py'. (However, we did not display DTC in our project.)

```
class Sensor:
    def __init__(self, shortName, sensorName, sensorcommand, sensorValueFunction, u):
        self.shortname = shortName
        self.name = sensorName
        self.cmd  = sensorcommand #pid code
        self.value= sensorValueFunction# above are the difinitions
        self.unit = u

SENSORS = [
    Sensor("pids"                  , "Supported PIDs"                              , "0100" , hex_to_bitstring ,""       ),
    Sensor("dtc_status"            , "S-S DTC Cleared"                             , "0101" , dtc_decrypt      ,""       ),
```

5.3.2 Obd-io.py

This file defines a series of functions to communicate with ELM327 scan tool.
A 'send_command' function was writing to pass the commands (AT commands or OBD commands) to ELM327 cable

```
 def send_command(self, cmd):
     """Internal use only: not a public interface"""
     if self.port:
         self.port.flushOutput()
         self.port.flushInput()
         for c in cmd:
             self.port.write(c)
         self.port.write("\r\n")
         #debug_display(self._notify_window, 3, "Send command:" + cmd)
```

.

An example of sending AT command is

```
self.send_command("atz")   # initialize
```

This command reset the ELM327 chip and all setting are returned to their default values. Other AT commands are listed in the Appendix.
An example of sending OBD command is as follows:

```
self.send_command("0100")
```

in which '01' indicates Mode 01 and '00' means PID 00. This command requests the information of availability of PIDs [01 - 20] in Mode 1. We can also call 'send_command(sensor.cmd)' for OBD commands that are defined in 'obd_sensors.Sensor' mentioned before. The function 'get_result' shown below is to read data from ELM327.

```
def get_result(self):
    """Internal use only: not a public interface"""
    #time.sleep(0.01)
    repeat_count = 0
    if self.port is not None:
        buffer = ""
        while 1:
            c = self.port.read(1)
            if len(c) == 0:
               if(repeat_count == 5):
                   break
               print "Got nothing\n"
               repeat_count = repeat_count + 1
               continue

            if c == '\r':
               continue
               #'>' inELM 327 indicates the devic is ready to receive characters
               if c == ">":
               break;

            if buffer != "" or c != ">": #if something is in buffer, add everything
               buffer = buffer + c

        #debug_display(self._notify_window, 3, "Get result:" + buffer)
        if(buffer == ""):
           return None
        return buffer
    else:
       debug_display(self._notify_window, 3, "NO self.port!")
    return None
```

The first 4 bits of a response from ELM327 is the echo of the command , therefore, the function 'interepret_result' was added to filter the data returned from 'get_result'.

```
def interpret_result(self,code):
    """Internal use only: not a public interface"""
    # Code will be the string returned from the device.
    # It should look something like this:
    # '41 11 0 0\r\r'

    # 9 seems to be the length of the shortest valid response
    if len(code) < 7:
        #raise Exception("BogusCode")
        print "boguscode?"+code

    # get the first thing returned, echo should be off
    code = string.split(code, "\r")
    code = code[0]

    #remove whitespace
    code = string.split(code)
    code = string.join(code, "")

    #cables can behave differently
    if code[:6] == "NODATA": # there is no such sensor
        return "NODATA"

    # first 4 characters are code from ELM
    code = code[4:]
    return code
```

5.3.3 obd_capture.py

As long as the port is connected to the ELM327 scan tool, the data for each available sensor is requested again and again in this file.

5.4 GUI

5.4.1 wxPython

In order to construct a user interface for the program, wxPython was used as a toolkit for popular cross-platform GUI programming. This is a blend of wxWidgets and Python programming. Original wxWidgets (written in C++) is a huge class library. GUI classes from this library are ported to Python with wxPython module, which tries to mirror the original wxWidgets library as close as possible. So, wx.Frame class in wxPython acts much in the same way as wxFrame class in its C++ version.
wxObject is the base for most of the classes. An object of wxApp (wx.App in wxPython) represents the application itself. After generating the GUI, application enters in an event loop by MainLoop() method. Following diagrams depict the class hierarchy of most commonly used GUI classes included in wxPython.

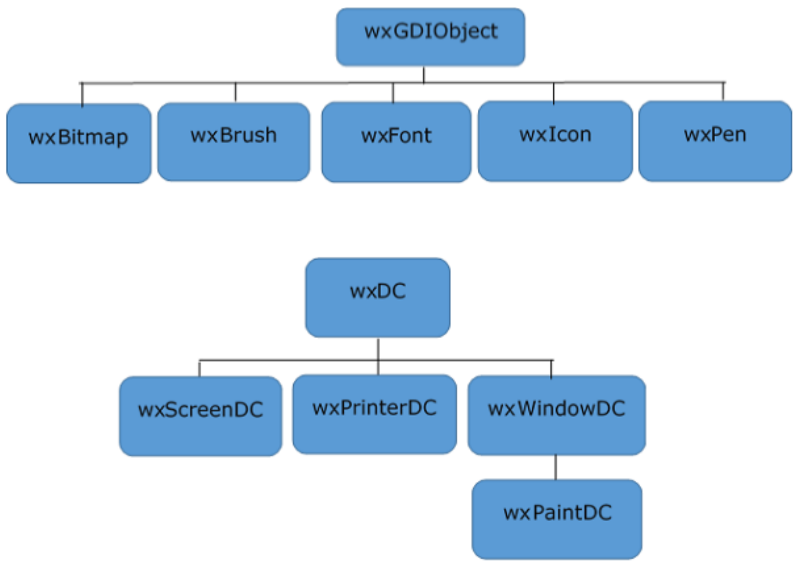


Figure 5.4.1. Hierarchy block diagram of wxPython

5.4.2 obd_gui.py

The GUI program is designed to display information about sensors (name, value, unit) in a grid of 3 rows/1 columns - with all the colour, font, size, style, alignment adjusted as follow:

```
# Text for sensor value
if type(value)==float:
    value = str("%.2f"%round(value, 3))
t1 = wx.StaticText(parent=self, label=str(value), style=wx.ALIGN_CENTER)
t1.SetForegroundColour('WHITE')
font1 = wx.Font(32, wx.ROMAN, wx.NORMAL, wx.NORMAL, faceName="Monaco")
t1.SetFont(font1)
boxSizer.Add(t1, 0, wx.ALIGN_CENTER | wx.ALL, 20)
boxSizer.AddStretchSpacer()
self.texts.append(t1)

# Text for sensor name
t2 = wx.StaticText(parent=self, label=unit+"\n"+name, style=wx.ALIGN_CENTER)
t2.SetForegroundColour('WHITE')
font2 = wx.Font(13, wx.ROMAN, wx.NORMAL, wx.BOLD, faceName="Monaco")
t2.SetFont(font2)
boxSizer.Add(t2, 0, wx.ALIGN_CENTER | wx.ALL, 5)
self.texts.append(t2)
gridSizer.Add(boxSizer, 1, wx.EXPAND | wx.ALL)
```

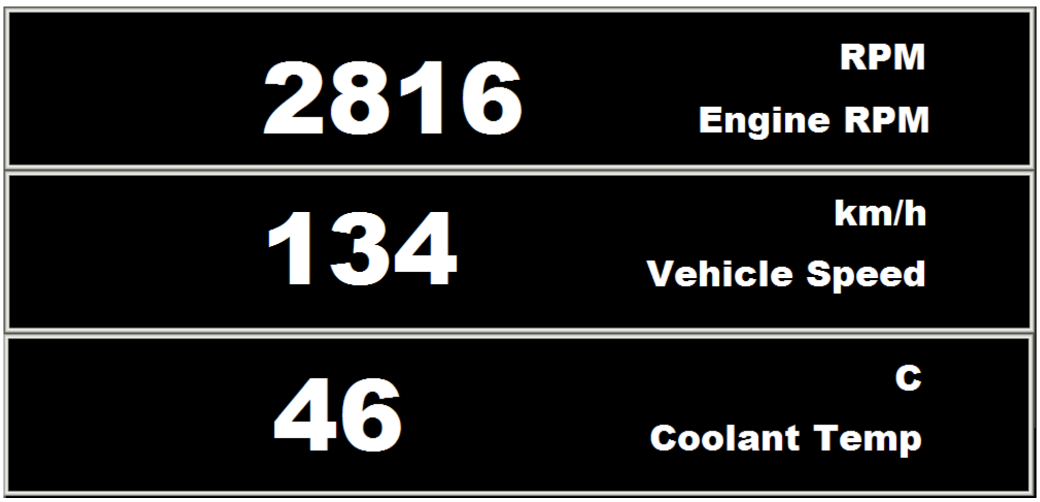


Figure 5.4.2.a Output panel

```
def showLoadingScreen(self):
    """
    Display the loading screen.
    """
    boxSizer = wx.BoxSizer(wx.VERTICAL)
    self.textCtrl = OBDText(self)
    boxSizer.Add(self.textCtrl, 1, wx.EXPAND | wx.ALL, 92)
    self.SetSizer(boxSizer)
    font3 = wx.Font(16, wx.ROMAN, wx.NORMAL, wx.NORMAL, faceName="Monaco")
    self.textCtrl.SetFont(font3)
    self.textCtrl.AddText(" Opening interface (serial port)\n")
    self.textCtrl.AddText(" Trying to connect...\n")

    self.timer0 = wx.Timer(self)
    self.Bind(wx.EVT_TIMER, self.connect, self.timer0)
    self.timer0.Start(1000)
```

Figure 5.4.2.b Loading panel

As the data changing, the information displayed will be updated at the rate /0.3sec (indicated in timer update part). The update rate cannot be too fast since there is a specific amount of time for the processor to acquire all sensors data.

```
    # Timer for update
    self.timer = wx.Timer(self)
    self.Bind(wx.EVT_TIMER, self.refresh, self.timer)
    self.timer.Start(300)


def refresh(self, event):
    sensors = self.getSensorsToDisplay(7)

    itext = 0
    for index, sensor in sensors:

        (name, value, unit) = self.port.sensor(index)
        if type(value)==float:
            value = str("%.2f"%round(value, 3))

        if itext<len(self.texts):
            self.texts[itext*2].SetLabel(str(value))

        itext += 1
```

5.5 Autostart

The GUI design in this project was basically the shell result of a programming file in python. During the modification process, it was fine to adjust the code and use mouse and keyboard to control the file running. However, in order to fit the car working environment, the auto-start of the finalized file was required. It was related to the rebooting setting of the Raspberry Pi. There were several methods to achieve this goal. In this case, a separated folder 'autostart' was created under /home/pi/.config directory and a command file was built within it and with the content shown below. Noted that .config directory was a hidden file. Hence, it must be directed into by typing the full address in the address line.

```
[Desktop Entry]

Name=ScanTest

Comment=My Python Program

Exec=python /home/pi/ScanTest/Scan.py

Icon=/home/pi/ScanTest/Scan.png

Terminal=false

MultipleArgs=false

Type=Application

Categories=Application;Development;

StartupNotify=true
```

The two important lines for this short programming were discussed as following.

Exec = python /home/pi/ScanTest/Scan.py: define which file will be run after system is rebooting

Icon = /home/pi/ScanTest/Scan.png: choose the icon image for this auto-start file

Upon rebooting the system, the system was able to open the destination file and display the desired information.

5.6 Terminal Control

Giving commands in the terminal can simplify the control and be free from the mouse. During the project, we grabbed the opportunity to learn basic and useful commands under LINUX terminal to achieve our purpose. I listed out few commonly used commands as below.

- Sudo reboot: to restart the system
- Man XXX: brings up the online LINUX command manuals to help understand the usage of each command
- Pwd: Shows what directory/folder you are in. In Raspberry Pi LINUX, the home directory is /home/pi
- Cd XXX: Change directory
- Python XXX: run the python code in the terminal shell
- Ls XXX: lists out all the files in the current folder

For example, in the below shown command, the first command lists out all the files in the current folder and then goes to the desktop folder.

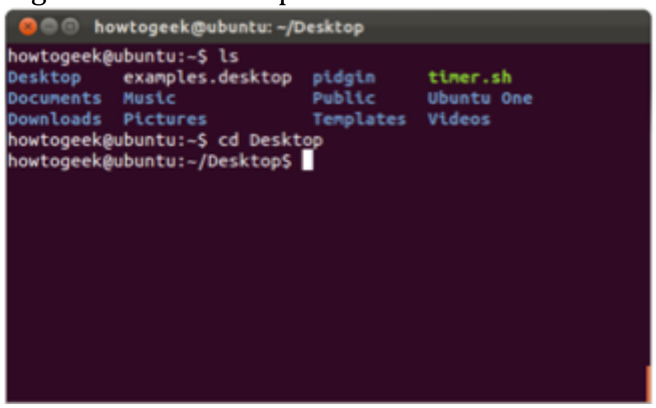


Figure 5.5.a Terminal control example

CHAPTER SIX – RESULT AND DISCUSSION

6.1 System Integration & Demonstration

Two phases of the information were shown after running the GUI file. First phase was the loading window. It was the initializing stage of the program which was in the white background color and displayed the project title and team members' names.

The second phase was displaying the required data – engine speed, vehicle speed and coolant temp respectively in the vertical order. For better display quality, background was in black while the content was in white, and later will be changed to blue series due to the sensitivity of the glass film used in the project. Within each display box, left hand side showed the number in a very big size while right hand side showed the units and data names. The data shown on the LCD screen of the OBD simulator could be used as the reference for data checking. Below two images shows that three data on projection glass and LCD screen were the same.

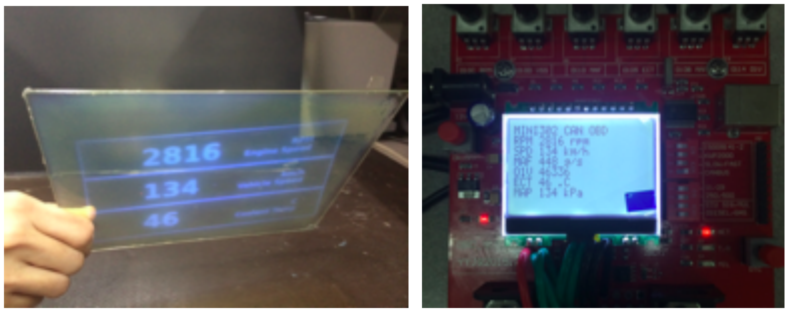


Figure 6.1.a (left) Displaying data on glass. (right)Displaying data on the OBD simulator

When switching data on OBD simulator, it changed accordingly on both glass and LCD screen. Therefore, the data transfer and display ware reliable.

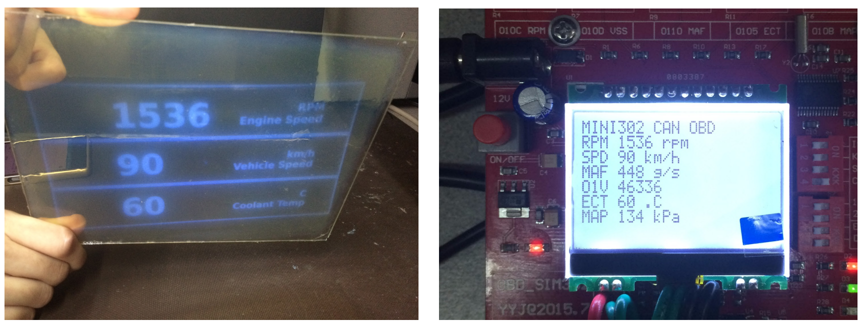


Figure 6.1.b (left)Changing data on glass (right) Changing data on the OBD simulator

6.2 Difficulties encountered

6.2.1 Different protocols

The ELM327 Cable supports several different OBD protocols. For a real car, user may never have to choose which one it should use (since the factory settings cause an automatic search to be performed for you). However, since an OBD simulator was used in this project, we have to specify a protocol

Code snip: obd_io.py

```
debug_display(self._notify_window, 2, "atz response:" + self.ELMver)
self.send_command("ate0") # echo off
debug_display(self._notify_window, 2, "ate0 response:" +
self.get_result())
self.send_command("AT SP 00") # telling the program to
automatically search and save it in internal EEPROM
```

```
debug_display(self._notify_window, 2, "atsp response:" +
self.get_result())
self.send_command("0100")
ready = self.get_result()
```

Protocol	Description
0	Automatic
1	SAE J1850 PWM (41.6 kbaud)
2	SAE J1850 VPW (10.4 kbaud)
3	ISO 9141-2 (5 baud init)
4	ISO 14230-4 KWP (5 baud init)
5	ISO 14230-4 KWP (fast init)
6	ISO 15765-4 CAN (11 bit ID, 500 kbaud)
7	ISO 15765-4 CAN (29 bit ID, 500 kbaud)
8	ISO 15765-4 CAN (11 bit ID, 250 kbaud)
9	ISO 15765-4 CAN (29 bit ID, 250 kbaud)
A	SAE J1939 CAN (29 bit ID, 250* kbaud)
B	User1 CAN (11* bit ID, 125* kbaud)
C	User2 CAN (11* bit ID, 50* kbaud)

*user adjustable

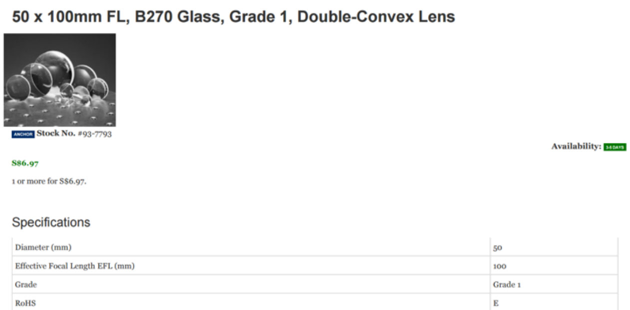
Table 6.2.1.a "AT SP h" command summary and description

6.2.2 Sensors to be displayed

Since the data was transmitted to processor in serial - a list of sensors info (name, value, unit) in binary data stream, it's impossible to restricted the acquisition of data into specific sensors. Therefore, to limit the sensors to be displayed down to 3 necessary sensors: Engine Rpm - Vehicle Speed - Coolant Temperature, which changes are performed on the obd_gui.py program.
This change is practically important since in real life environment (real car driving), no button is provided for driver to press on to switch the gauge. Another version of all sensors displayed are retained under the name obd_gui_fullsensors.py

6.2.3 Limitation in understanding optics and obtaining good quality lenses

Experimental quality lenses used for the projector are cheaply manufactured and do not include much specifications. This limits the amount of control over designing an optical system. In order to use professional optical design softwares like Zemax, other specifications like radius of curvature are needed. Also, information on multiple-elements systems online are based on good quality lenses and involves highly complicated optical knowledge. Therefore, we were unable to make full use of the resources available online to design a simple, effective and affordable lens system. We had to work around with only the fundamentals of optical theory, to understand how image is formed in the lens, and make modifications such that they do not challenge the limits of an inefficient lens system. The final image produced by our modified lens is a huge improvement over the single lens initially prototyped, however, the image still has some spherical aberration at the edges, which may have arisen due to the insufficient diameter tolerance of the third lens. It is also not as feasible to opt for well-made lenses for prototyping, as cost of a single lens can easily equal price of an entire projector.


50 x 100mm FL, B270 Glass, Grade 1, Double-Convex Lens

ANCHOR Stock No. #93-7793

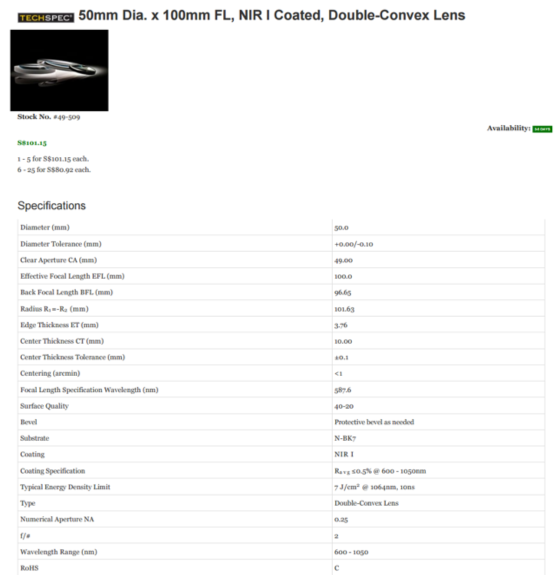
Availability: 3-5 DAYS

S$6.97

1 or more for S$6.97.

Specifications

Diameter (mm)	50
Effective Focal Length EFL (mm)	100
Grade	Grade 1
RoHS	E


TECHSPEC® 50mm Dia. x 100mm FL, NIR I Coated, Double-Convex Lens

Stock No. #49-509

Availability: 3-5 DAYS

S$101.15

1 - 5 for S$101.15 each.
6 - 25 for S$80.92 each.

Specifications

Diameter (mm)	50.0
Diameter Tolerance (mm)	+0.00/-0.10
Clear Aperture CA (mm)	49.00
Effective Focal Length EFL (mm)	100.0
Back Focal Length BFL (mm)	96.65
Radius R_1 = $-R_2$ (mm)	101.63
Edge Thickness ET (mm)	3.76
Center Thickness CT (mm)	10.00
Center Thickness Tolerance (mm)	±0.1
Centering (arcmin)	<1
Focal Length Specification Wavelength (nm)	587.6
Surface Quality	40-20
Bevel	Protective bevel as needed
Substrate	N-BK7
Coating	NIR I
Coating Specification	R_{avg} ≤0.5% @ 600 - 1050nm
Typical Energy Density Limit	7 J/cm² @ 1064nm, 10ns
Type	Double-Convex Lens
Numerical Aperture NA	0.25
f/#	2
Wavelength Range (nm)	600 - 1050
RoHS	C

Figure 6.2.3.a (left) Specifications of experimental quality lens
Figure 6.2.3.b (right) Specifications of a highly refined lens

The data sheet of two 50 x 100mm double convex lens from Edmund Optics are compared above. On the left is an experimental quality lens that was utilised in our project. Its costs $6.97, and 4 specifications were given. For the good quality lens on the right, all specifications of the lens were given however, it costs $101.15. Therefore, there will always be a compromise between quality of an optical system and its costs.

CHAPTER SEVEN – TEAMWORK AND PROJECT MANAGEMENT

7.1 Grouping

The projection on glass system design was divided into two parts. One part was to modify the traditional projector which was widely used in the classrooms and could project on the large-sized screens. The main objective for this part is to make the projector adapt to the car environment by adjusting the image size and light source. The other part focused on the data acquisition and processing. It included the data transfer system design and HUD GUI design. In order to launch the project effectively, the team was divided into two sub-groups of three students. Each sub-group worked on one part – namely optics team and software team.

7.2 Project Timeline

7.2.1 The project timeline of the optics team

Week No.	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Recess Week	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Research and Planning	■	■	■	■			■	■	■		■			
Analysis of components				■	■		■							
Measurements and Calculations				■	■	■			■			■		
Testing And Lens Prototyping						■								
Experiment to find focal length								■						
Testing and 2nd Lens Prototyping										■				
LED and other component prototyping											■	■		
LED testing and Integration													■	
System Integration													■	■

Table 7.2.1.a Project timeline of the optics team

7.2.1.1 Research and Planning - Understand more about optics theories, projector and its optical components, single lens and multiple elements system. Concluded researches, made plans to achieve objectives. Review of implementations (success and limitations), and made improvements on plans.

7.2.1.2 Analysis of components - Analysed physical hardware of projector and projector lens, understand its design and working principles.

7.2.1.3 Measurements and calculations - Measured units such as distances and light intensity, of the projector, and made use of formulas to calculate variables in the system.

7.2.1.4 Testing and Lens Prototyping - Tested out single lenses that was based on calculations and measurements, and noted various positions to achieve ideal image distance and size. Made observation of results and conclusion of implementation.

7.2.1.5 Experiment to find focal length - Conducted physical experiment to find focal length of individual lens in multiple-lens system.

7.2.1.6 Testing and 2nd Lens Prototyping - Tested out new three-lens system which was designed based on calculations. Made adjustments to position of lens to improve on image distance and size.

7.2.1.7 LED and other component prototyping - Roughly integrated LED into system to test. Designed and implemented cooling system for LED. Designed prototype for reflector, improved on and finalised prototype. Constructed final prototype.

7.2.1.8 LED Testing and Integration - Integrated LED and reflector into projector, tested out components as part of projector system.

7.2.1.9 System Integration - Integrated with software and demonstrated projection on glass. Made adjustments for better display quality.

7.2.2 The project timeline of the software team

Week No.	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Recess Week	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Project Planning	■	■												
Arduino Study and Demonstration		■	■	■	■									
OBD System Study and Design				■	■	■								
Raspberry Pi Initialization							■	■						
Python Programming Study							■	■	■		■	■		
LINUX Terminal Control									■	■				
OBD Data Acquisition and Processing									■	■	■			
HUD GUI Design and Modification										■	■	■	■	■
System Integration													■	■

Table 7.2.2.a Project timeline of the software team

7.2.2.1 Project Planning – Brainstormed the possible approaches for the data transfer system. Listed out all the components needed in the project and sourced the suitable types and vendors. Came up with a rough project timeline.

7.2.2.2 Arduino Study and Demonstration – Researched on the basis of Arduino programming and control. Demonstrated the single 7 segment 4 digits LED display and double 7 segment 4 digits LED display. Understood the difficulties on using Arduino as the microcontroller unit (MCU) and looked for a more feasible MCU for future system design.

7.2.2.3 OBD System Study and Design – Studied on the working principle of OBD system and OBD simulator. Understand the communication protocols through the OBD simulator and OBD cable. Studied on the different connection means from OBD to external devices – cable and Bluetooth. Searched for data acquisition samples.

7.2.2.4 Raspberry Pi Initialization – Understood the work principle of the Raspberry Pi as the MCU. Installed the operating system and necessary applications.

7.2.2.5 LINUX Terminal Control – Studied on how to use the terminal window to write the commands.

7.2.2.6 OBD Data Acquisition and Processing – Amended the reference code files for appropriate data acquisition by defining the different serials and sensors, and setting up the communication between input and output.

7.2.2.7 HUD GUI Design and Modification – Designed the GUI to display the required data acquired from OBD simulator. Enable the system to auto-start when rebooting the system.

7.2.2.8 System Integration – Integrated with the optics system and demonstrated the projection on the glass. Made the necessary amendments for the better display quality.

CHAPTER EIGHT – CONCLUSION AND RECOMMENDATION

The Heads-up Display (HUD) system was implemented by integrating the optics system (modified projector) and the data processing system (MCU and OBD system). The optics system successfully shortened the focal distance and fit the image into the desired size. The data processing system was able to acquire the data from OBD simulator to MCU, and visualise the selected data into graphical form which was displayed on the glass. Students have understood the essential concepts and learned the useful knowledge of the optics, MCU (Arduino and Raspberry Pi), and python programming. It will benefit them for their future projects and studies. Based on the wide research and the experience gained throughout the project, the possible improvement opportunities for future groups carrying on this project were recommended as well.

The image quality, especially for the edges, could be further enhanced by fabricating a more sophisticated lens system which will correct the spherical aberration better, such as replacing cheap spherical lenses having high aberration issues with aspherical lenses. A proper lens housing should also be designed to properly integrate the lenses.

Cooling of the LED needs to be further improved, to ensure that the LED is driving at its optimal temperature so that its performance will not degrade rapidly. Currently, the LED is only being cooled by an aluminum-copper heatsink and a few IC chips heatsink. The centrifugal fan in the projector only removes heat from the LCD panel, therefore, a fan may be added to cool the LED, such as attaching a DC axial fan behind the heatsink. Reflectivity of the reflector should also be further improved, such as by adjusting angle of each surface, or replacing it with an acrylic mirror which can be easily cut and is quite sturdy.

Our projector's application is very unique as it demands a short throw distance while maintaining a small image size. Therefore, it is possible to reconsider new layout of the components and make the projector even more compact. Placement and orientation for mounting projector in the car should be assessed so as to include it as a design aspect, to avoid late issues like the projector body blocking the image or it being too bulky to be placed on the dashboard.

The DC step down transformer of the projector can be integrated together with the LED. Since projector still needs to be powered by an AC voltage, portability can also be taken into consideration for future designs. It may be possible to power the projector using batteries, however size will be compromised.
As there are a few heat and dust sensitive optical elements, such as the LCD panel which has to be maintained below 40°C, care should be taken when making modifications to any part of the optics.

For the data processing system, more functionalities could be added into the GUI such as navigation, calendar and weather information from the external device. Another direction falls on making the GUI interactive with users. The projection could be designed to be a platform to pop up the notifications for calls and messages from the smart phone. In this way, it could eventually enable drivers to use the critical functions of the smartphone without looking at it, hence achieves a safer drive.

In conclusion, the project execution and the final demonstration met the initial objectives of. The experience obtained and the prototype designed will be the useful reference for the subsequent projects.

REFERENCES

http://castle.eiu.edu/~ddavis/chapter_18/x_18.18b.jpg

http://www.ebay.com/gds/Understanding-the-Differences-between-LED-LCD-and-DLP-Projectors-/10000000177630814/g.html

http://www.odec.ca/projects/2005/dong5a0/public_html/ConvexLens.png

http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/

https://www.navdy.com/

http://www.pas.rochester.edu/~pavone/particle-www/telescopes/ComputerCommands.htm

http://digitalphotographylive.com/chromatic-spherical-aberration/

http://www.iap.uni-jena.de/iapmedia/Lecture/Design+and+correction+of+optical+systems1351638000/DaCOS12_Lecture_Part_12_Correction_of_aberrations_1_112.pdf

http://www-ferp.ucsd.edu/LASERLAB/TUTOR/lensdesign.shtml

http://www.physicsclassroom.com/class/refrn/Lesson-5/Converging-Lenses-Object-Image-Relations

http://www.physicsclassroom.com/class/refrn/lesson-5/the-mathematics-of-lenses

https://www.christiedigital.com/documents/presentations/simuniversity2014/simu2014-01-fundamentalsofprojectiontechnology-alenkoebel.pdf

http://www.edmundoptics.com.sg/resources/application-notes/optics/advantages-of-fresnel-lenses/#applications

https://www.cis.rit.edu/class/simg712-01/notes/10-ray_model.pdf

http://www.prophotonix.com/resources/technical-overviews/about-chip-on-board.aspx.

http://www.digikey.com/en/articles/techzone/2014/mar/the-rise-of-chip-on-board-led-modules

http://canopyairportparking.com/green-parking/led-lighting/

http://www.homepower.com/articles/home-efficiency/electricity/led-vs-cf-vs-incandescent

APPENDIX A CAO TONTONG'S INDIVIDUAL REPORT

1. OBD-II simulator and ELM327

1.1 On Board Diagnostics System

On Board Diagnostics-II (OBD-II) is a vehicle network for the diagnosis and checking of a vehicle. Produced cars are equipped many sensors for measurement and regulation which are controlled by the Electronic Control Unit (ECU). As advances in vehicle and computer technologies, all parts of the vehicle are controlled by the ECU including the drive system, braking system, engine system and steering system. Among them, the engine is the most significant part of the vehicle, main features such as ignition timing, idle threshold setting, variable valve timing and fuel injection requires a precise control system. OBD-II provides access to data from the engine control unit (ECU) and offers a valuable source of information when troubleshooting problems inside a vehicle.

All light duty vehicles less than 8,500 pounds sold in North America since 1996 are required to equipped with the OBD-II system and the medium duty vehicles up to 14,000 pounds must also be equipped with OBD II systems beginning in 2004. In addition, since 2010, all heavy-duty vehicles over 14,000 pounds are required to support OBD-II diagnostics as well.

The SAE J1962 specification standardised two hardware interfaces for OBD called type A and type B. Both interfaces are female, 16-pin (2x8), D-shaped connectors. Among the 16 pins, the function of 7 pins are specified in the SAE J1962 including pin4 for chassis ground, pin 5 for signal ground, pin 16 for battery voltage and pin 2, 6, 7, 10, 14, 15 for reading and writing using different signal protocols. The rest of coins are left for manufacturer discretion.

Currently, there are five OBD-II signal protocols are in use: SAEJ1850 PWM (pulse-width modulation) 、SAEJ1850 VPW (variable pulse width) 、ISO14230 (KWP-2000) 、ISO9141 、ISO15765-4/SAEJ2480(CAN). The table below shows the properties of each protocols.

	SAE J1850 PWM	SAE J1850 VPW	ISO 9141-2	ISO 14230 KWP2000	ISO 15765 CAN
Data Rate	41.6 kbit/s	10.4 kbit/s	10.4 kbit/s	1.2 to 10.4 kbit/s	250 kbit/s or 500 kbit/s
Utilised Pin	pin 2 : Bus + pin 10: Bus -	pin 2: Bus +	pin 7: K-line pin 15: L-iine	pin 7 : K-line pin 15: L-line	pin 6 : CAN High pin 14 : CAN Low
Voltage Level	Hight voltage : +5V	High voltage : +7V Decision point: +3.5V	K-line idles high	High voltage: +12V	CANH: 3.5V CANL: 1.5V
Message Length	12 bytes, including CRC	12 bytes, including CRC	max 260 bytes (data field max 255)	data field max 255	Base format: 11 bits Extended format: 29 bits

Table 1 Comparison of five signal protocols

A list of On-board Diagnostics Parameter IDs(PIDs) were defined by the SAE J1979 standard. PID codes are used as a diagnostic tool to request data from the vehicle. Technicians can use PID with a scan tool to communicate with the controller area network of a vehicle such as CAN-bus. When a PID code is reported by the scan tool , the controller will recognise the PID report the value for that PID to the bus. Then the scan tool reads the response and displays it to the technician. Table 2 shows the 10 modes of operation described in the latest OBD-II standard SAE J1979. Each mode contains multiple commands to control specific sensor.

Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

Table 2 Description of 9 modes of PID

In ‘obd_sensor.py’, we defined a series Sensors to assign each PID command to its correspond sensor name, value function and unit for displaying. Moreover, the DTC codes are also defined in ‘obd2_codes.py’.

```
class Sensor:
    def __init__(self, shortName, sensorName, sensorcommand, sensorValueFunction, u):
        self.shortname = shortName
        self.name = sensorName
        self.cmd  = sensorcommand #pid code
        self.value= sensorValueFunction# above are the difinitions
        self.unit = u

SENSORS = [
    Sensor("pids"                , "Supported PIDs"                              , "0100" , hex_to_bitstring ,""       ),
    Sensor("dtc_status"          , "S-S DTC Cleared"                             , "0101" , dtc_decrypt      ,""       ),
```

1.2 OBD-II Simulator

In our project, we use an OBD-II simulator to replace the vehicle and ELM327 cable as the scan tool to read data from the simulator. The switches on the right of the LCD screen can choose the OBD protocol. The available protocols including

ISO 15765 CAN (250kbit/s 11bit ID),
ISO 15765 CAN (250kbit/s 29bit ID),
ISO 15765 CAN (500kbit/s 11bit ID),
ISO 15765 CAN (500kbit/s 29bit ID),
ISO 9141 (5 baud init),
ISO 14230 KWP (fast init),
ISO 14230 KWP((5 baud init).

The working voltage for our simulator is 12 V with DC current 2A. It supports real-time data stream (Mode 1) and the freeze frame data (Mode2). The knobs at top of the simulator can adjust the main parameters of engine and these data displays on the LCD screen dynamically. Users can choose DTC mode and no-DTC mode by pressing the DTC button (Mode3, Mode4). It can also produce the vehicle VIN code and other vehicle information (Mode9)

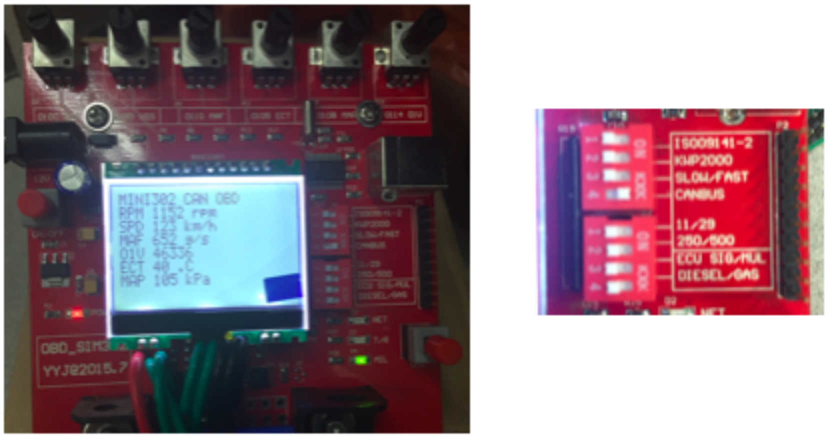


Figure 1 OBD-II simulator

1.3. ELM327 Cable

The scan tool used in this project is an ELM327 cable. To communicate with ELM327, our software program need to be set properly. A proper setting includes a proper baud rate (9600 baud if pin 6 = 0V or 38400 baud if pin 6 = Vcc), 8 data bits, no parity bits, 1 stop bit and the correct port.

```
class OBDPort:
    """ OBDPort abstracts all communication with OBD-II device."""
    def __init__(self,portnum,_notify_window,SERTIMEOUT,RECONNATTEMPTS):
        """Initializes port by resetting device and gettings supported PIDs. """
        # These should really be set by the user.
        baud     = 38400 #or 9600 |
        databits = 8
        par      = serial.PARITY_NONE  # parity
        sb       = 1                   # stop bits
        to       = SERTIMEOUT
        self.ELMver = "Unknown"
        self.State = 1 #state SERIAL is 1 connected, 0 disconnected (connection failed)
        self.port = None

        self._notify_window=_notify_window
        debug_display(self._notify_window, 1, "Opening interface (serial port)")

        try:#
            self.port = serial.Serial(portnum,baud, \
            parity = par, stopbits = sb, bytesize = databits,timeout = to)

        except serial.SerialException as e:
            print e
            self.State = 0
            return None
```

There are two command sets supported by ELM327, one is the PID command set which is used to communicate with a vehicle or the OBD-II simulator, another kind of command is the Hayes AT command set which is considered as the internal commands.

A ‘send_command’ function was wrote to pass the commands (AT commands or OBD commands) to ELM327 cable.

```
def send_command(self, cmd):
    """Internal use only: not a public interface"""
    if self.port:
        self.port.flushOutput()
        self.port.flushInput()
        for c in cmd:
            self.port.write(c)
        self.port.write("\r\n")
        #debug_display(self._notify_window, 3, "Send command:" + cmd)
```

ELM327 requires OBD commands containing ASCII codes for hexadecimal digits (0 to 9 and A to F) and all AT commands must start with ‘AT’ which means attention, so type of cmd is either a hex string or a string starting with ‘at’.

An example of sending AT command is as follows:

```
self.send_command("atz")   # initialize
```

This command reset the ELM327 chip and all setting are returned to their default values. Because ELM327 is not case-sensitve and will ignore the spacing so the commands ‘AT Z’, ‘atz’, and ‘AtZ’ are the same to an ELM327 chip. In our codes, we also use ‘ate0’ to turn off the character echo and ‘at sp 00’ to speed up protocol initiation and detection for the suitable protocol.

An example of sending OBD command is as follows:

```
self.send_command("0100")
```

in which ’01’ indicates Mode 01 and ’00’ means PID 00. This command requests the information of availability of PIDs [01 - 20] in Mode 1. We can also call ‘send_command(sensor.cmd)’ for OBD commands that are defined in ‘obd_sensors.Sensor’ mentioned before.

The function ‘get_result’ shown below is to read data from ELM327.

```
def get_result(self):
    """Internal use only: not a public interface"""
    #time.sleep(0.01)
    repeat_count = 0
    if self.port is not None:
        buffer = ""
        while 1:
            c = self.port.read(1)
            if len(c) == 0:
               if(repeat_count == 5):
                   break
               print "Got nothing\n"
               repeat_count = repeat_count + 1
               continue

            if c == '\r':
               continue
               #'>' inELM 327 indicates the devic is ready to receive characters
               if c == ">":
               break;

            if buffer != "" or c != ">": #if something is in buffer, add everything
               buffer = buffer + c

        #debug_display(self._notify_window, 3, "Get result:" + buffer)
        if(buffer == ""):
           return None
        return buffer
    else:
       debug_display(self._notify_window, 3, "NO self.port!")
    return None
```

The response to OBD command received from ELM327 are hexadecimal digits in pairs. The first 4 bits will repeat the command and the rest of data is the requested data from OBD. Because of the response echo the command, the mode value in the response would be added with 40 to distinguish with a command. For example, the response to "0100" may be "41 00 BF 9F B9 90". The first byte '41' represent Mode 01 and "00" represent PID 00. The rest 4 bytes in digital bits are a series of 0(not supported) and 1 (supported) to indicate whether the correspond PID is supported.

It is obvious that the data returned in 'get_result' is the combination of the echo of the command and the data requested. Therefore, the function 'interepret_result' was added to filter out unnecessary information.

```
def interpret_result(self,code):
    """Internal use only: not a public interface"""
    # Code will be the string returned from the device.
    # It should look something like this:
    # '41 11 0 0\r\r'

    # 9 seems to be the length of the shortest valid response
    if len(code) < 7:
        #raise Exception("BogusCode")
        print "boguscode?"+code

    # get the first thing returned, echo should be off
    code = string.split(code, "\r")
    code = code[0]

    #remove whitespace
    code = string.split(code)
    code = string.join(code, "")

    #cables can behave differently
    if code[:6] == "NODATA": # there is no such sensor
        return "NODATA"

    # first 4 characters are code from ELM
    code = code[4:]
    return code
```

2. Arduino — LED

One way to approach our objective is to use LED light reflection on the glass. Based on an Arduino Uno board, we managed to show 4 digits' data using LED lights. The connection between the Arduino uno and the 7 segment 4 digits LED is shown in the figure below.

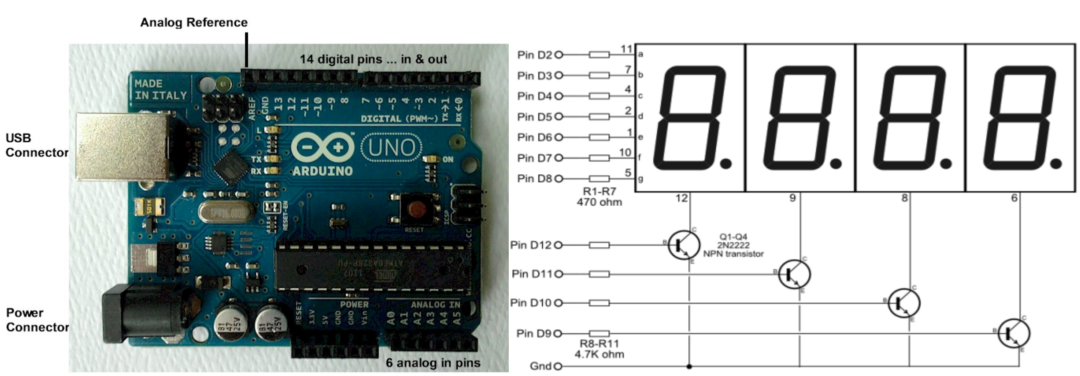


Figure 2 Connection between Arduino Uno to 7-segment 4-digit LED

The Uno is a micro controller board based on the ATmega328P. It has 14 digital input/output pins, 6 analog in pins, a USB connector, a power connector, a 16 MHz quartz

crystal an ICSP header and a reset button. The working voltage is 5V and the recommended input voltage is 7V - 12V.
Set pin D2-D12(connect to LEDs) and pin D1(TX, connect to OBD) and as output and pin2(RX, connect to OBD) as input.

Because of the digits will be reflected on the glass, we need to redefine the on/off mode of the segments for each number (0-9). Thus, the digits displayed on the LED look like figure below.

Figure 4. Inverse digits display

The correspond code for each digits is shown in the table below:

Decimal	7 Segment Display						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	0	1	1	0	1	1
3	1	1	1	1	0	0	1
4	0	1	1	0	1	0	1
5	1	1	0	1	1	0	1
6	1	1	0	1	1	1	1
7	0	1	1	1	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	1	0	1

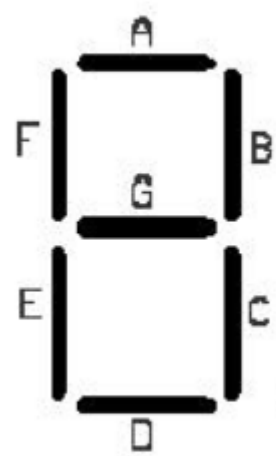


Table 3 Seven-segment Display

3. Raspberry pi

In this project, the Raspberry Pi 2 Model B was used as the processor. The board has 4 USB ports, 40 GPIO pins, a full HDMI port (connect to the projector in this project), an Ethernet port and a Micro SD card slot. Raspberry Pi 2 Mode B is a very powerful mini computer using a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor and1GB RAM. Figure below is the block diagram of Raspberry pi.

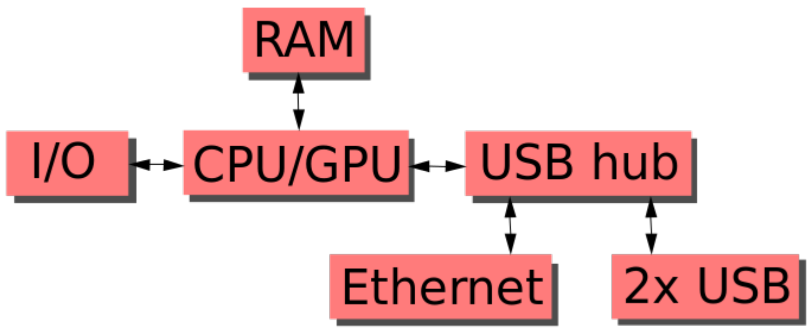


Figure 5 Block diagram for Raspberry Pi

An SD card with the Raspbian Operating System was prepared for our Pi. The SD card was making on a Mac for our case. After downloading and extracting the image file of Raspbian, it expanded into a folder "Raspberry-PI-SD-Installer-OS-X-master ". The image file was also moved to this folder. The Pi installer was run in the Terminal by typing

sudo ./install raspbian.img

where 'raspbian.img' is the name of the image file. The SD card was then inserted into the card slot of Raspberry pi.

Usually, Pi will start with a command line. The command 'sudo raspi-config' make us able to change some default settings for the Pi. In the Raspi-config interface, there are several terms can be adjusted. For example,'Expand_rootfs' enable the Pi to use the whole SD card and by disabling 'overscan', the Pi would use the whole screen when connecting to any displaying devices. We also select starting desktop automatically in 'Boot behavior' options.

Reference

[1] Sung-hyun B and Jong-Wook J, "Implementation of integrated OBD-II connector with external network," ScienceDirect, 2014.

[2] Hu Jie, Yan Fuwu, Tian Jing, Wang Pan and Cao Kai,"Developing PC-Based Automobile Diagnostic System Based on OBD System", IEEE，2010.

[3] ELM Electronics, "ELM327 OBD to RS232 Interpreter" www.elmelectronics.com

APPENDIX B PAY SHI YUN CARIN'S INDIVIDUAL REPORT

Abstract

Sir Isaac Newton was a well-known physicist and mathematician who discovered the theory of optics. A few experiment was done by him and the aim was to discover theories of reflection and refraction. He studied the behavior and properties of optics and concluded light as a stream of particles. Light is known as electromagnetic waves. Optics is the branch of physics whereby light can be classified as ray, wave and particles, including its interactions with matter and the construction of instrument. An optical instrument produces light waves to project the light out of the instrument.

In this project, projector will be used as an optical instrument to study on how it works to form image. According to research, when projector placed at a short distance, it will create a large image also most projectors come standard with a variable focal length of zoom range from 6cm to 10.5cm. The objectives of this experiment are to design a short throw projector which described how light travel when it interacts with lens to form an image size of A5 size on a special fuming glass.

1. INTRODUCTION

1.1 Background

Projector forms an image through the lens by reflection and refraction, this leads to a real image can be formed on the screen. Understanding of geometric optics is important in this project, when light interact with medium it causes light to change direction. Light can be clearly seen when travel through opaque material however when light projection on glass it depends on the distance. Characteristic of image are being studied in this project so as to observe what type of image is produced on the screen through the type of lens used by the projector. It is found out that converging lens can produce real image.

1.2 Individual Objectives

This project has been divided into two parts which involved hardware and software. Scope of this individual report will be focusing on optics hardware research and single lens experiment which causes spherical aberration. The objectives are to prove on how the research has guided the team in achieving the final result for the experiment. This individual report task which include the following:

1. Research on projectors and optics system
2. How Projector works

3. Single lens observation: spherical aberration
4. Solutions to solve the problem

2. RESEARCH

2.1 Comparison between DLP, LCD and LED projector

	Advantages	**Disadvantages**
LCD	Higher Contrast	Colour decay
	Sharper image	Limited lifetime
		Screen-door effect
		Dead Pixels
DLP	Higher Contrast	Rainbow effect
	Less pixilation	
	Filter free design	
LED	Longer life for lamp	More expensive, price per lumen
	Energy efficient light source	

Table 1. Comparison between DLP, LCD and LED projectors

Projector Type	**Technology Type**	**Light Source**
DLP	Reflective	LED or Standard lamp
LCD	Transmissive	Standard lamp
LED	Reflective and Transmissive	LED

Table 2. Factors of DLP, LCD and LED

2.2 How Projector works

The projector lamp produces diffuse light source that will travel over a wide range of angle. Providing efficiency and good angular resolution in one particular direction so that projector can work rightly. The aim is obtained by using condenser mirror which collects the light emerging from light source over a wide range, providing an even illumination over the transparent object and concentrating the light as it passes through the projector lens. The condenser must have a short focal length so that object plane is well illuminated by light traversing it over a wide range of angles. Fresnel lens is used for the light source of the lamp to concentrate it into the projector lens while a polarizer is used to form a sharp image.

Figure 1. UC 40 Projector

Figure 2. UC 40 Projector Zoom lens

3. METHODS

3.1 Type of Lens

UC 40 zoom lens is formed from two plano-convex lenses and one concave lens in the middle.

Research and studies on different type of lens so as to find the most suitable lens to fix into UC 40 projector in order to achieve clearer image projected on the screen. According to the different type of lens, converging lens produce real image whereby there is thickest at the centre of the lens and curved outward which is known as the convex lens. After discussion about the different lens, the team decided to use double convex lens in the experiment. Three double convex lenses of different focal length that bought from Edmund optics are combined with the original UC40 projector lens in order to get the best image result. Double convex lens length include focal length of f = 6.6 cm, f = 8.7 cm, f = 10 cm.

3.2 Problem on the Type of Aberration

It is important to note that the image formed on the screen depends on the light source and position of the lens. The objective is to show how the light converges to form an image by the projector. Problem encountered while working with single lens, it is found that it formed blur image when projected on the screen. Research is carried out to find exactly what type of aberration and to provide a solution to problem. Studies on different aberrations which include chromatic aberration and spherical aberration that would most probably affected the image but it is found out that spherical aberration is the problem that affected the image. Spherical aberration result in focal points of rays far from the principal axis of a spherical lens which are different from the focal points of rays of the same wavelength passing near the axis.

Figure 3: Double Convex Lenses

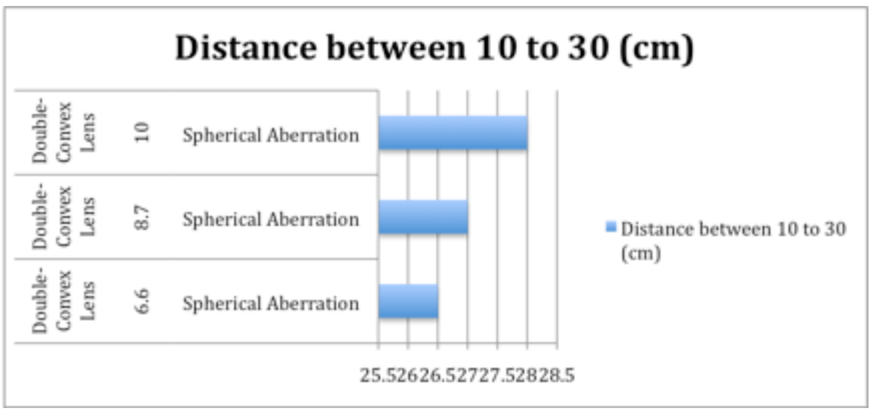


Table 3. Single Lens observation

3.3 Solution to solve the Aberration

According to the research, solution to solve the problem is to use combination of lenses. It depends on using multi or three lens system of combination lens that can be used to solve spherical aberration problem. Using spherical surface lenses and same focal length can minimize spherical aberration whereby lens shaped so that refraction is shared roughly equally between the two surfaces. There are two ways to solve the problems this include the following:

1. Reduced by limiting the aperture
2. Using a combination of lens of different materials and focal lengths in order to partly correct for the spherical aberration.

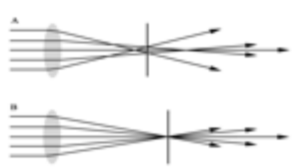


Figure 4. Spherical Aberration

4. CONCLUSION AND RECOMMENDATIONS

4.1 Conclusion

UC40 LED projector has been used in this experiment to change the lens so that it can be projected at the focal length range from 10cm to 30cm on a special fuming glass which is in good legibility under sunlight. It is to observe how light travel through a distance of vacuum and interact with matter. This individual report described the contribution to the team in order to achieve the project goal. By doing research and experiment, it helped in gaining knowledge and experience for the optics system. In order to achieve the result, trial and error in testing to find the problem encountered and then do research to solve the solution. The most important in participating for this Design and Innovation Project is teamwork and cooperation also school laboratory safety rules should be strictly followed. The key to complete this project is to have a well-planned time management. Further improvement can be made to achieve by having smaller size of the projector and also using Bluetooth to send the signal to the projector instead of using cable.

5. References

[1] Graham Smith and Terry A.king. (n.d.). Optics and Photonics An introduction.John Wiley & Sons.

[2] Microsoft Encarta. (n.d.) Isaac Newton Institute for Mathematical Sciences. (4 March 2016) Retrieved from https://www.newton.ac.uk/about/isaac-newton/life

[3] Understanding the differences between LED, LCD and DLP (7 April 2016) http: //www.ebay.com/gds/Understanding-the-Differences-between-LED-LCD-and-DLP-Projectors-/10000000177630814/g.html

[4] Google image (11 April 2016) http://www.optics.rochester.edu/workgroups/cml/opt307/spr11/greg/var12.jpeg

APPENDIX C CHONG CHEAN HWA'S INDIVIDUAL REPORT

Introduction

Head up display (HUD) is defined as any form of transparent data display without having the viewer look away from their view point. HUD is no longer a new technology as the aviation technology has been using HUD in their aircraft. This not only serves as a convenience but also a safety measurement as pilots can read useful information projected on their screen without shifting focus away from the sky.

Research Problem and Objective

Our project team aims to incorporate this into cars. Using the same concept, the setup we plan to develop aims to project data such as engine rotation speed, speedometer etc onto the screen, which allows driver to read such data without looking away from the road. Modern cars use an On Board Diagnostic (OBD) to provide such data in real-time. Thus, our goal is to project OBD data using a projector onto a glass.

Current OBD system does not display data that a projector can read, an intermediary is needed between the OBD and projector to convert the format. We have chosen to use a market available microcontroller (Raspberry Pi) to translate said data. On the other hand, market available projectors project at a distance ranging from 1 - 4m, depending on the projector. We require a projector of a much shorter throw distance, i.e. 15 – 25cm for it to be placed on the dashboard of a car. To realise this, we need to modify the lens of the projector to focus at a shorter distance.

Project Overview

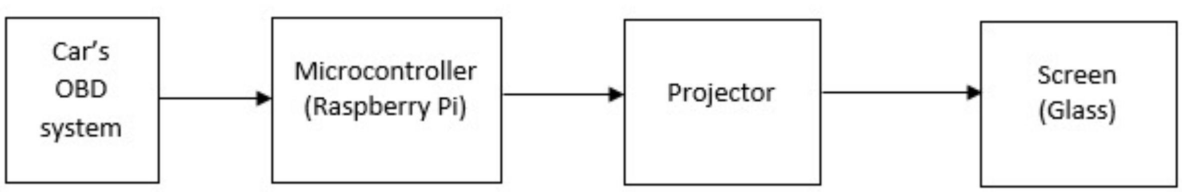


Figure 1 Project overview block diagram

Figure 1 shows the brief overview of our project.

The projector feeds the data from the Raspberry Pi through a HDMI cable to the projector. The projector then displays this readable format onto a glass at a fixed distance.

Personal Role

The project team is divided into 2 subgroups, which will work on software (OBD to Raspberry Pi) and optics (projector to screen) respectively. My contribution, as part of the optics group, is modifying the lens of the projector such that its throw distance of a few metres is reduced to tens of centimetres. This will be the focus of this report.

Optics Overview

- **Understanding optics**

Before we can modify the projector throw distance, we must first understand the basics of optics. Optical path can be mapped by drawing a ray diagram that traces the object through the lens to the image. Figure 2 shows an example of ray diagram of a convex lens, which will be the lens we will mainly use for this project.

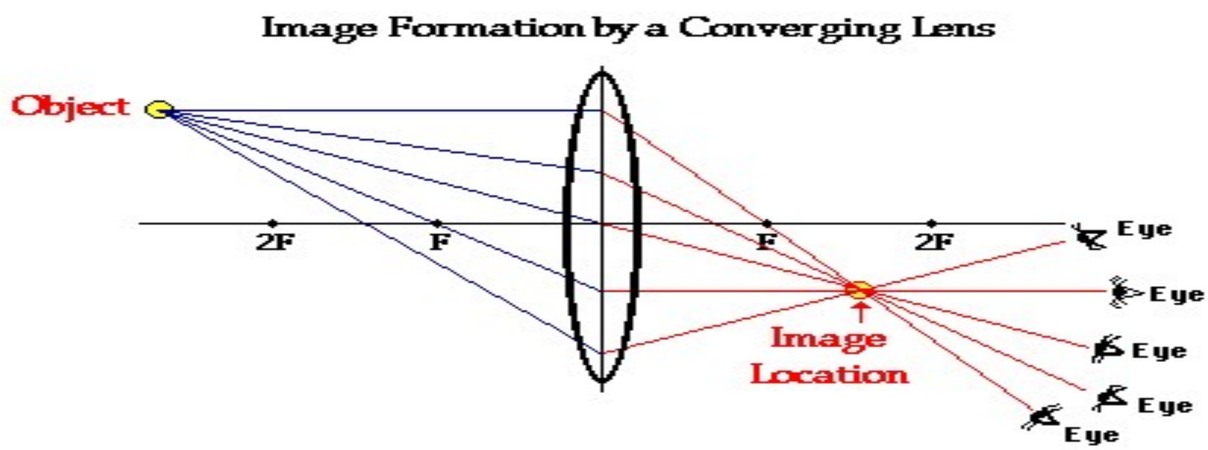


Figure 2 Ray diagram for a convex lens

The distance of object, u from the lens and its image distance, v through the lens of focal length, f are related by a general equation known as the thin lens equation which can be derived from the ray diagram in Figure 2. The thin lens equation is given by

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

Object distance, u and image distance, v can also be related by the ratio of their heights, i.e. height of image, h_i and height of object, h_o known as the magnification factor, M given by

$$M = \frac{h_i}{h_o} = -\frac{v}{u}$$

Furthermore, lenses are prone to aberrations due to its spherical nature. Light rays passing through the lens may not converge at the same point due to the varying thickness of the lens from its centre to its edge, thus, refracting light rays to varying degrees. As a result, image formed will be out of focus. Figure 3 shows a graphical representation of spherical aberration and a comparison to an ideal case [1].

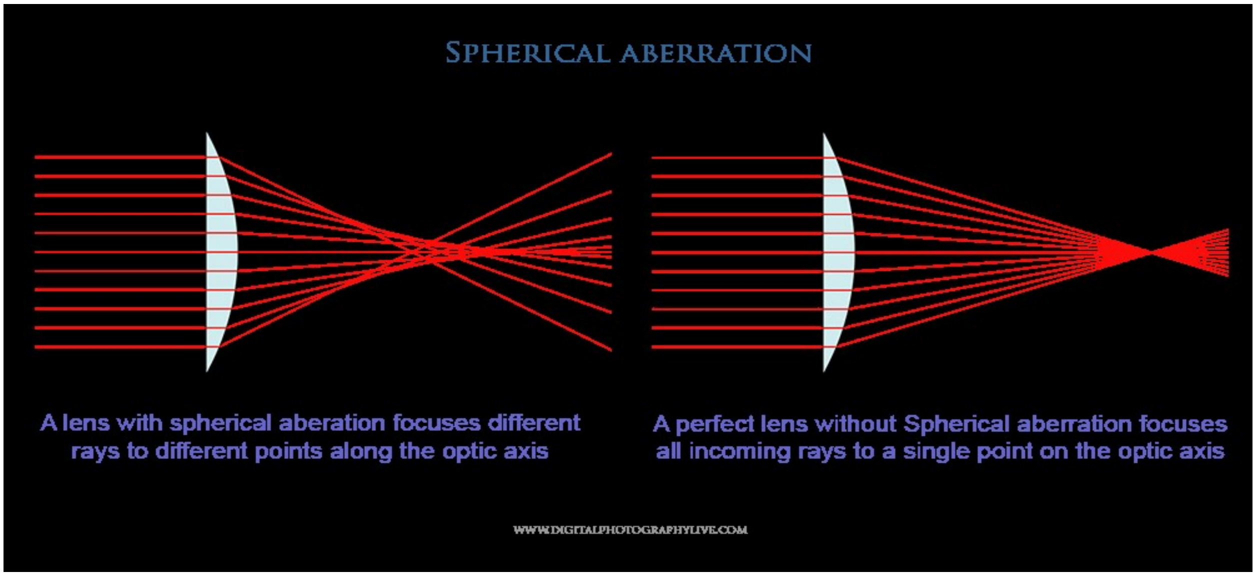


Figure 3 Spherical aberration and ideal case

- **Projector**

To begin experimentation, we first procured a projector. The model we decided upon was the UNIC UC40 mini projector with its technical specifications listed in Appendix A. Our main concern was the brightness, measured in lumens. Commercially available pico projectors often requires a room significantly dim to produce an image bright enough for viewers. We required a projector able to produce a bright image in a car, in the outdoors. Thus, we chose UNIC UC40 due to its higher than average brightness of 800 lumens.

  - Obtaining ***u***

Through its user guide, we were able to extract object distance, ***u*** derived from its throw distance and projection size as stated in the user guide. Throw distance is a term used by the projector to measure the distance from the tip of the lens to the screen, which we will take as image distance, ***v***. We will use throw distance and image distance, ***v*** interchangeably throughout the report. To reassure our object distance, ***u*** obtained, we used a different approach to find object distance, ***u*** by measuring the LCD screen diagonally and taking the ratio of projection size to LCD screen, i.e. height of image/ height of object = magnification factor, m. From there we can easily derive object distance. The mathematics can be found in Appendix B.

Object distance, ***u*** measured from two different methods agrees with both methods yielding 13.33cm and 13.70cm, only a slight discrepancy of 0.63cm.

- **Lens**

After finding out object distance, ***u***, we can steer back to our main objective of shortening the throw distance of the projector. Taking a look at the thin lens equation,

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

We require a shorter throw distance or image distance, ***v***, we can have object distance, ***u*** increased or focal length, ***f*** decreased. In practical sense, we hypothesised that we can move the lens of the projector out of the projector, effectively increasing object distance, ***u***, or simply use a different lens of lower focal length to achieve a shorter image distance, ***v***.

  - Increase ***u***

We first tried the former method. Experiment confirmed with our hypothesis. When we move the lens out of the projector to a distance of 6cm away from its original position, the image distance, ***v*** was reduced to 36cm. Though it reduced the throw distance significantly, having the lens protrude 6cm away from the projector proved to be an unnecessary design and bulky contraption. Not to mention the lack of space efficiency when placed on a dashboard of a car. Therefore, we decided to go with replacing the lens with one of lower focal length.

  - Decrease ***f***

Firstly, we found out the focal length we require by working out the image distance, ***v*** we need and with the object distance, ***u*** from which we previously calculated. Using the thin lens equation, to obtain the desired image distance, ***v*** of 15-25cm, and an object distance, ***u*** of 13cm, which we may vary if need be, we required the new focal length to be of the range 6.9-9.4cm. Since custom made lens was out of our budget, we procured a set of 3 biconvex lenses of focal length, 6.6cm, 8.7cm and 10cm which were available on the market.

Using the new lenses, we first tested with the single lens individually. The resulting throw distances obtained were much shorter than the original throw distance of a few metres. However, the image produced was far from presentable as the spherical aberration were too prominent. The image was found to be out of focus near the edge.

There are many ways to correct spherical aberration: lens splitting, power splitting, cementing, [2] to name a few. We have decided to use the same method used by the default design, by using 3 lenses of both positive and negative, i.e. convex and concave, separated by some distance. This multiple lens system is able to minimise spherical aberration. To better understand this, we have dismantled and study the original lens of the projector. The setup is as shown in figure 4.

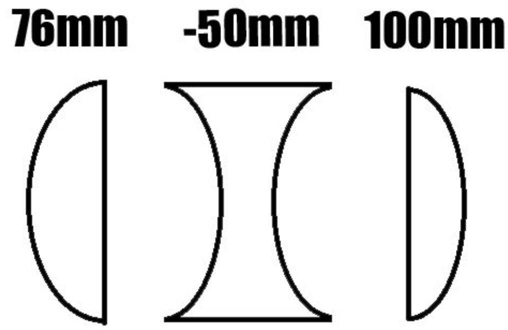


Figure 4 Original lens system of projector UNIC UC40

The original lens system consisted of 2 plano-convex lenses on each end, with focal length 7.6cm and 10cm respectively. A biconcave lens of -5cm focal length sits in the middle. The focal lengths were measure using a parallel light source through the individual lenses and marking the point which they meet.

The effective focal length of a 3 lenses system [3] is given by

$$f_{eff3} = \frac{\frac{f_1+f_2-d_{12}}{f_1 f_2} + f_3 - d_{23}}{\frac{f_1+f_2-d_{12}}{f_1 f_2} f_3}$$

Where f_1, f_2, f_3 are the focal lengths of lenses, and d_{12}, d_{23} are the distances between lenses 1 and 2, and lenses 2 and 3 respectively.

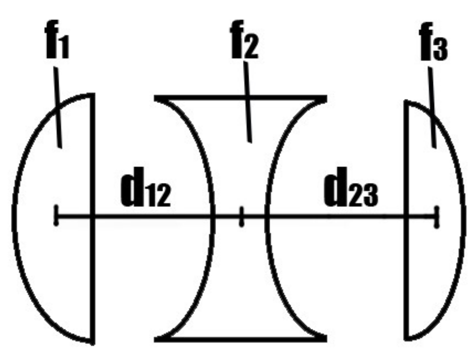


Figure 5 Effective focal length of 3 lenses calculation

After measuring all required variables of the original lens system, we now experiment with different sets of lenses to come up with the desired effective focal length. Using the same setup as the original, we keep d_{12} and d_{23} constant while swapping the outer convex lenses. The results are tabulated in table 3 in the appendix. Although a few combinations fit our desired focal length of 6.9-9.4cm, we finally decided with using the system shown in Figure 9 shown in the Appendix D. The modified lens system used the original plano convex lens of 7cm as the inner lens, biconcave lens of -5cm as the middle lens and changing the outer lens to a biconvex lens of 6.6cm.

Results and Discussions

Using the effective focal length formula for 3 lenses, taking f_1 = 7.6cm, f_2 = -5cm, f_3 = 6.6cm. The effective focal length is 7.6cm. The rationale behind this combination is the position of new lens system is at approximately the same position as the original lens, while its throw distance is reduced to a mere 23cm. Thin lens formula would place the image at 17cm. The discrepancy is due to many factors, mainly human error as most measurements done were only close estimations due to the complex build of lens system. The effective focal length formula of 3 lenses used were also an estimation derived from the effective focal length formula of 2 lenses. Besides that, from Appendix C, it can be seen that the formula is very sensitive to slight variation.

Image produced is of presentable quality as spherical aberration has been corrected significantly due the usage of multiple lens system. Figure 6 shows the distinct difference before and after spherical aberration correction near the edge of the image where it is most prominent.

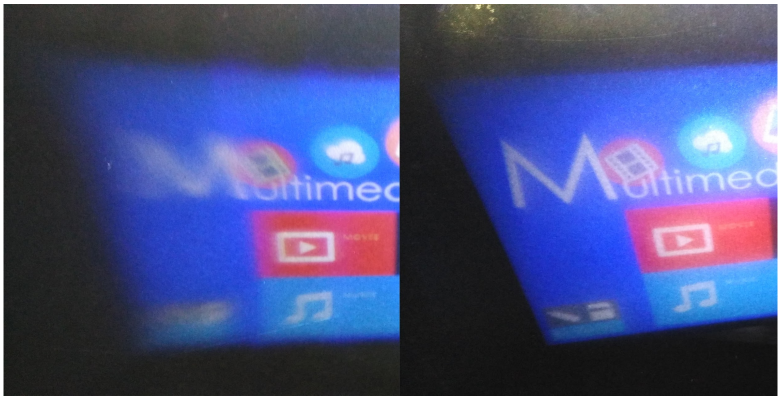


Figure 6 (Left) Before correction (Right) After correction

Acknowledgement

I would like to thank Assoc. Prof Chen for the opportunity to work on this project and for giving us ideas to improve upon our project. I enjoyed working on this project a lot and not to mention I have learnt a lot about optics. This gives me a new found appreciation for lens makers such as Nikon and Canon. I would also like to thank the mentors, Liu Weizhong and Paul Zhen, for their guidance throughout the 13 weeks. Last but not least, much appreciation goes to our team for working together and completing this project.

Conclusion and Recommendation

The throw distance of a projector can be reduced by modifying the lens of the projector. By reducing the effective focal length of the lens system, the throw distance is reduced significantly to meet our needs. Spherical aberrations is apparent in a single lens system and thus multiple lens system is needed for correction. Future project may focus on the keystone effect of the projected image to account for the tilted screen of the car if need be. Else, further studies on improving its portability is also in order to make it feasible to be used in a car.

Reference

[1] Understanding Chromatic & Spherical Aberration of Lenses, Digital Photography Life, [online], Available: http://digitalphotographylive.com/chromatic-spherical-aberration/

[2] H. Gross (2012),"Design and Correction of Optical System", [online] available: http://www.iap.uni-jena.de/iapmedia/Lecture/Design+and+correction+of+optical+systems1351638000/DaCOS12_Lecture_Part_12_Correction_of_aberrations_1_112.pdf

[3] Effective Focal Length Calculator, Advance Energy Technology Group Centre for Energy Research, [online], available: http://www-ferp.ucsd.edu/LASERLAB/TUTOR/lensdesign.shtml

Appendix

Appendix A

Figure 7 Technical specifications of UC40

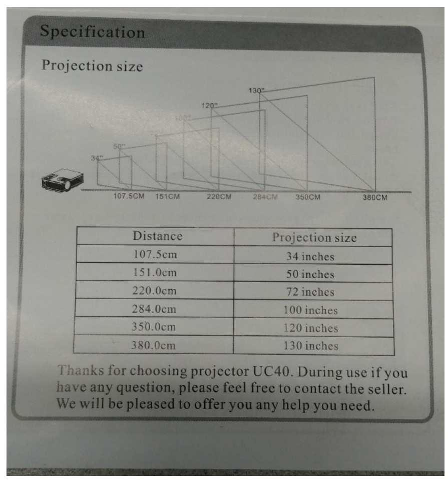


Figure 8 Projection distance and size obtained from user manual

Appendix B

	From f given, **f = 125mm**	From u
Distance, v (cm)	**Object Distance, u (cm)** **(1/f = 1/u + 1/v)**	**Calculated Magnification Factor, |M| = (-v/u)**
107.5	14.14	7.6
151	13.63	11.08

220	13.25	16.6
284	13.08	21.72
350	12.96	27
380	12.93	29.4
Average:	13.33	
Notes:	u varies as zoom lens moves	As u approaches f, image is more magnified

Table 1 Object distance, u from projection distance and focal length given

		Measured LCD screen = 11.66cm	
Projection Size (inch)	**Projection Size (cm)**	**|M| = (Projection Size / Object Size)**	**Object Distance, u (cm) = (v/M)**
34	86.36	7.41	14.51
50	127	10.89	13.86
72	182.88	15.68	14.03
100	254	21.78	13.04
120	304.8	26.14	13.39
130	330.2	28.32	13.42
		Average:	13.70

Table 2 Object distance, u from projection size and object size

Appendix C

f1	f2	d12	f3	d23	f' = (f1*f2)/(f1+f2-d12)	EFL = (f'*f3)/(f'+f3-d23)
7.6	-5	1.6	10	1.6	-38.00	12.84
7.12	-4.8	1.6	9.9	1.6	-47.47	12.00
7.6	-5	1.6	6.6	1.6	-38.00	7.60
10	-5	1.6	7	1.6	-14.71	11.06
7.6	-5	1.6	8.7	1.6	-38.00	10.70
10	-5	1.6	6.6	1.6	-14.71	10.00
8.7	-5	1.6	7	1.6	-20.71	9.47
8.7	-5	1.6	6.6	1.6	-20.71	8.70

Table 3 Effective focal length calculation for 3 lenses system

Appendix D

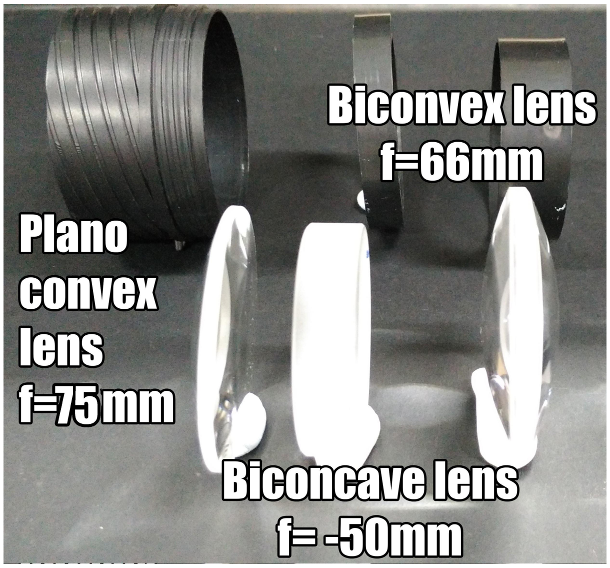


Figure 9 modified lens system

APPENDIX D CHUAH XIAO FEN'S INDIVIDUAL REPORT

1. Overview

The project objective is to retrieve real-time statistics from vehicles (ie. Car) through the On-Board Diagnostics (OBD) system in-built in the vehicle, and display the information through projection by a video projector. The projector would be placed on the dashboard of the vehicle, and a special translucent film will be adhered to the windscreen of the vehicle to act as the projection screen. The focal distance of the projector has to be short, and the projected image has to be bright enough to be viewed under direct sunlight. The projector also has to be as portable and small as possible.

2. Initial Research & Planning

Our group felt that modifying and improving on an existing projector would be more feasible than designing one from scratch, due to the limited optical knowledge and expertise we would be able to obtain within a limited time frame. We would purchase a mini projector and modify its lens, to achieve a short focused image distance between 10cm to 30cm. We would also change its light source to a brighter light source.

After doing research on projector types, we decided on a LCD (Liquid Crystal Display) projector illuminated by an LED. This projector type has a simpler setup that would require a smaller space, thus are commonly used in small projectors. We also decided on a mini projector, instead of a pico projector. Mini projectors are relatively small, lightweight and portable, but are bigger than pico projectors which are pocket-sized. However, mini projectors are much cheaper, and it would be more possible to work with and modify the components than with a pico projector, which has a more complex and compact setup.

We also decided to replace the projector's lens with a single biconvex lens to get an image focused at a nearer distance.

3. Analysis, Design & Testing

<u>3.1 Calculations and First Prototyping</u>

Firstly, we wanted to replace the projector's original lens with a new lens. We opened up our mini projector and analysed its components and layout.

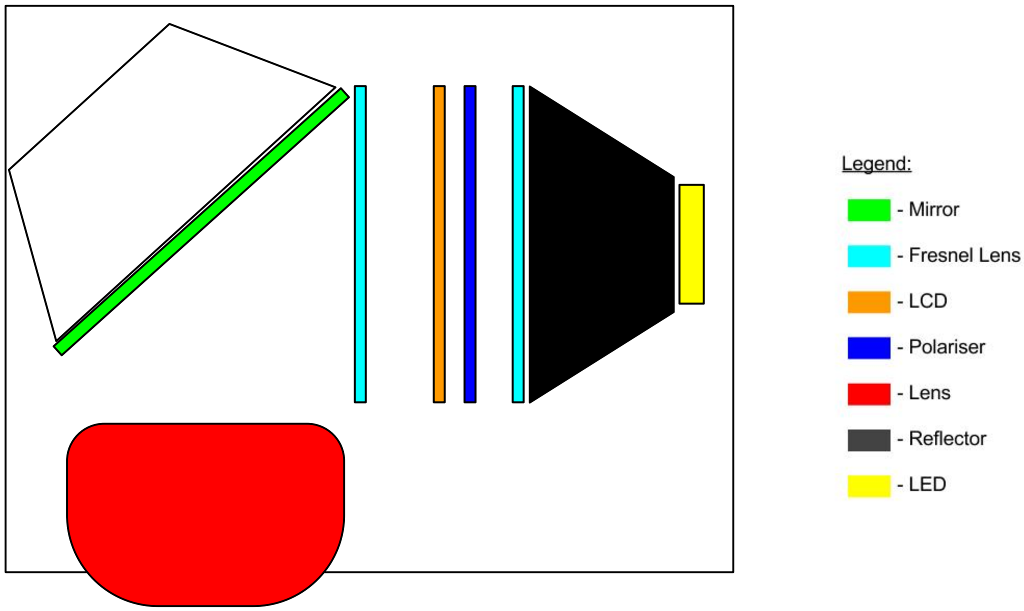


Figure 3.1.a Diagram of internal components of projector

How the projector works will be covered more in Chapter 3.3.

We made measurements of our projector, noting that the object distance from the LCD to the projector lens is 10.8cm, and from the LCD to the front of the projector is 14cm.

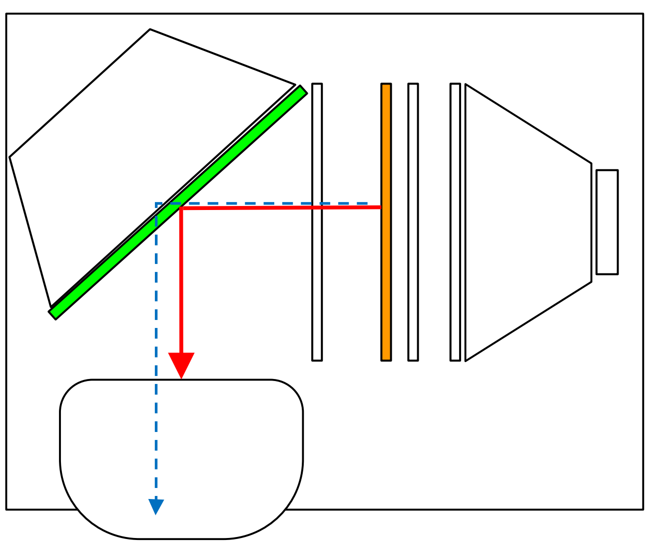


Figure 3.1.b Red arrow represents distance to projector lens
Blue dotted arrow represents distance to front of projector

We were then able to make calculations using Thin Lens Equation ($\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$), by substituting in our measured object distance u (14cm, represented by blue arrow Figure

3.1.b, as the single lens is flushed to the front of the projector) and range of desired image distance *v*. We were able to estimate a range of lens focal length *f* that would work for our application, and made purchase for 3 experimental quality biconvex lenses of *f* = 6.6cm, 8.7cm and 10cm. We also chose different grades of glass for all three lenses.
After obtaining our lenses, we tested out various positions of the lenses to find the shortest image distance, while maintaining a sizeable image (A5 size, 25.7cm) and a feasible *u* between 10cm to 30cm.

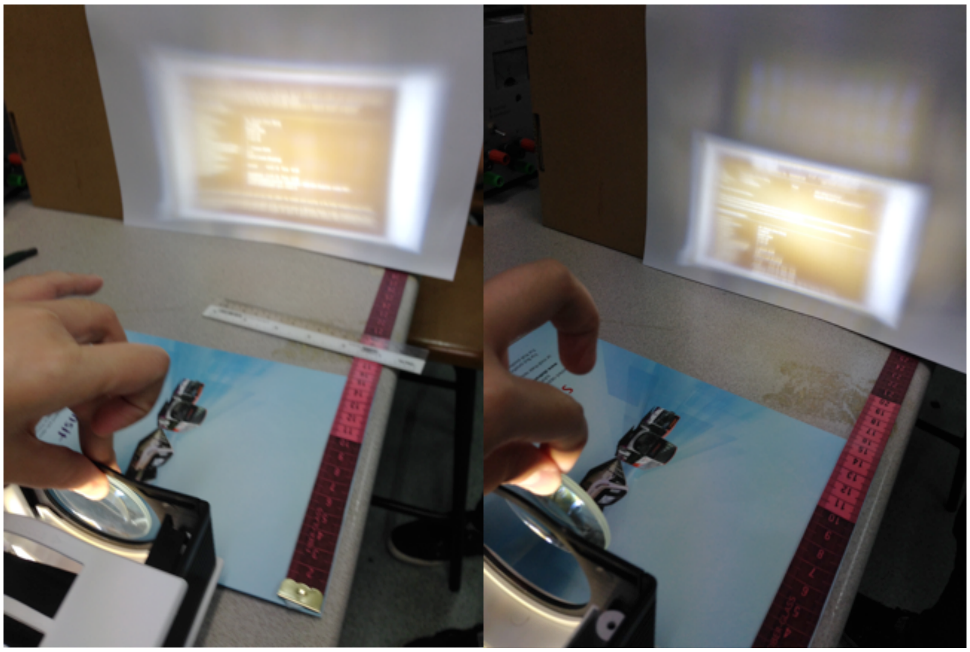


Figure 3.1.c A single lens tried at different positions and its resultant image

We made the following conclusions from our tests:

1) If the lens is placed further inside the projector, the focused image distance *v* becomes further away, and the image size increases. This agrees with biconvex lens image formation theory and lens magnification theory. When the lens is placed inside the projector, object distance *u* decreases. As *u* decreases towards lens focal length *f*, the image becomes larger however, *v* also increases. As long as *u* is larger than *f*, the image will be an enlarged version of the LCD.
2) If the lens is placed further out of the projector, the *v* is nearer, but the image size decreases. This also agrees with biconvex lens image formation theory. When the lens is placed outside the projector, *u* increases. As *u* increases towards *2f*, the image becomes smaller however, *v* also decreases. As long as *u* is smaller than *2f*, image will still be larger than size of the LCD.
3) Our ideal image size can only be achieved with lenses of focal length *f* 8.7cm and 10cm. This is because due to the design of the projector, lens with *f* = 6.6cm cannot

sufficiently move into the projector for u to be close enough to f, to get a image that is large enough.

4) For lens f = 8.7cm, it has to be placed around 1cm inside the projector to get an image of about A5 size, and v = 27cm
5) For lens f = 10cm, it has to be placed around 1cm outside of the projector, v = 28cm
6) Different grades of glass have little visible effect on the quality of image. This may be because the experimental grade lenses have less refined glass no matter which grade of glass, and have no coatings.
7) Image is very clear at the center, but gets unfocused away from the center

We did some research to theorise why the resultant image is unfocused away from the center. We concluded that our single lenses are not refined enough to get images with even sharpness throughout. Spherical aberrations occur due to the spherical surfaces of the lenses. This aberration may be corrected with an aspheric lens, however aspheric lenses are expensive and would thus make our prototype not cost-effective. The aberration however may be corrected with a simple multiple-lens system which would be cheaper to implement.

3.2 Improvements and Second Prototyping

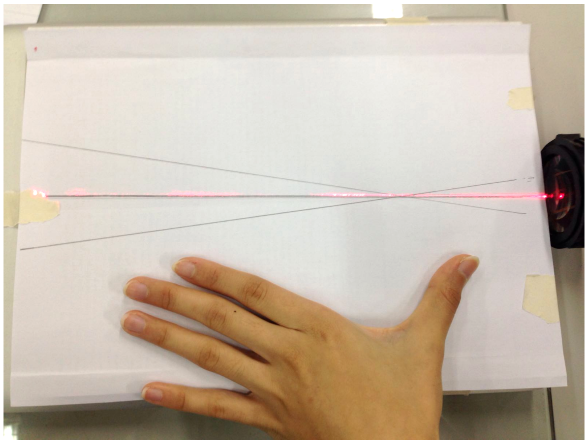
We first opened up our original projector lens, to understand the multiple-lens system it utilises, as the image projected from the lens is consistently sharp. Our original projector lens consists of three lenses: a biconcave lens between two biconvex lenses.

Plano convex Lens 1 (left), Biconcave Lens 2 (middle), Plano convex Lens 3 (right)

Figure 3.2.a Rough representation of lenses and their positions relative to each other

3.2.1 Experiment to find lens focal length

We conducted an experiment to find the focal length of each lens. We shone a collimated ray beam (laser) through each lens at various positions horizontally. We then marked out the path of the resultant rays.



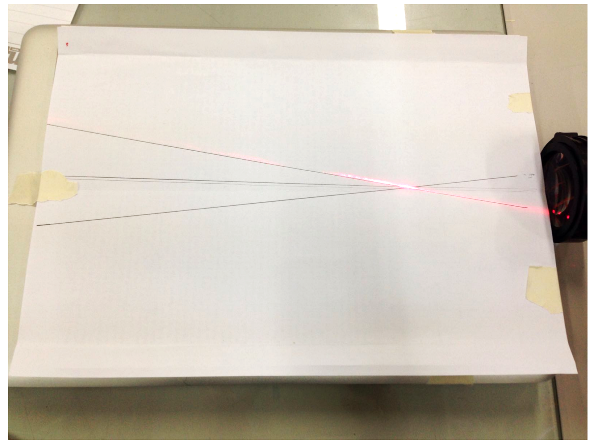


Figure 3.2.1.a (left) Laser beam shone through center of lens goes in a straight path
Figure 3.2.1.b (right) Laser beam shone on the left of the lens bends towards the right

After obtaining at least 3 paths, we were able to find a point where all rays converge. The focal length of the lens is then the distance between the lens to that point.

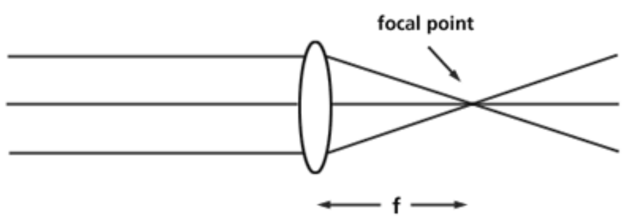
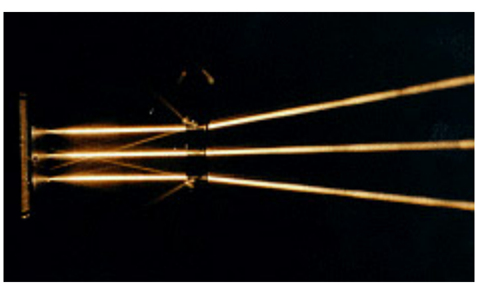


Figure 3.2.1.c shows a schematic drawing of a similar experiment

For concave lenses, light rays behind the lens diverge, and the virtual image is formed in front of the lens instead. If we trace the diverging rays backwards, we are also able to find a point where all rays converge. The focal length of the concave lens is the distance from that point to the lens.



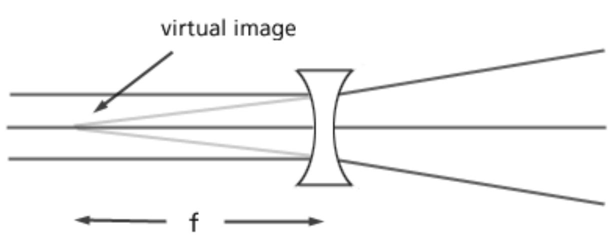


Figure 3.2.1.d (left) shows a similar experiment where collimated light rays are shone through a concave lens, and the rays diverge behind the lens
Figure 3.2.1.e (right) shows a schematic drawing of the experiment

From our experiment, we were able to find that our lenses are of f = 7.5cm, -5cm and 10cm respectively. Measurements of thickness and distances between each lens were also made with vernier calipers for precision.

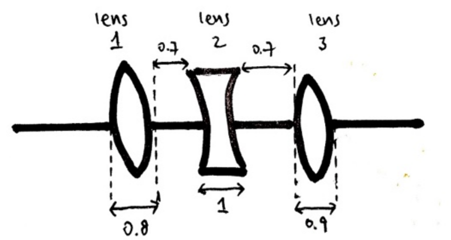


Figure 3.2.1.f Rough drawing of measurements of lenses

Afterwards, we are able to understand how each of the lens in the multi-lens system help the resultant image to be projected with even sharpness. The first lens converges light rays that focuses behind the second lens, since the convex lens' focal length is longer than distance between the lenses. The concave lens diverges the virtual object and widens the light beam. This wider light beam then passes through the third lens and converges, forming the final, real image on a screen.

3.2.2 Three-lens system

Multiple-lens system makes use of an effective focal length rather than focal length of each lens. For a three-lens system, the effective focal length can be calculated as follows:

$$\mathrm{EFL} = \frac{\left(\frac{f_1 * f_2}{f_1 + f_2 - d_{1\text{-}2}}\right) * f_3}{\left(\frac{f_1 * f_2}{f_1 + f_2 - d_{1\text{-}2}}\right) + f_3 - d_{2\text{-}3}}$$

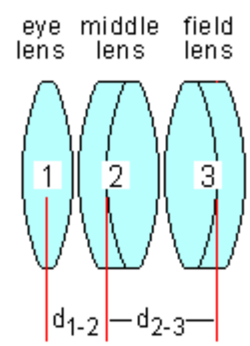


Figure 3.2.2.a (left) is formula for calculating effective focal length of the system

Figure 3.2.2.b (right) shows each lens element and their positions relative to each other

After the first lens testing, our group decided to make adjustments to our design and to compromise the size of the image for a closer image distance. Therefore, using the lenses we have and the three-lens system equation, we matched the different focal lengths to get a reasonable effective focal length. All lenses were spaced 1.6cm from each other, and the object distance used is 10.8cm, to take into consideration the thickness of the whole projector lens system.

Lens 1 (cm)	Lens 2 (cm)	D_{12} (cm)	Lens 3 (cm)	D_{23} (cm)	EFL (cm)	v (cm)
6.6	-5	1.6	8.7	1.6	8.7	44.7
6.6	-5	1.6	10	1.6	10	135
7.5	-5	1.6	6.6	1.6	7.5	24.5
7.5	-5	1.6	8.7	1.6	10.5	378
7.5	-5	1.6	10	1.6	12.5	-79.4
8.7	-5	1.6	10	1.6	16.8	-30.2

Table 3.2.2.a Different combination of lenses, their EFL and resultant image distance v

The combination of f = 7.5cm, -5cm and 6.6cm results in an effective focal length of 7.5cm, and an image distance of 24.5cm, while the rest of the combinations are not feasible for our application. We tested out our new three-lens system, and made adjustments to the position of the 6.6cm lens. If we shifted the lens till D_{23} = 2cm ie. shift the lens outward by 0.4cm, we got an image 2.1 times the magnification of our LCD at an image distance of 23.5cm. The new EFL is 7.4cm. The image projected from our new lens system is quite clear, except at the extreme edges of the image. This may be due to the quality of our 6.6cm lens at the diameter tolerance of the lens.

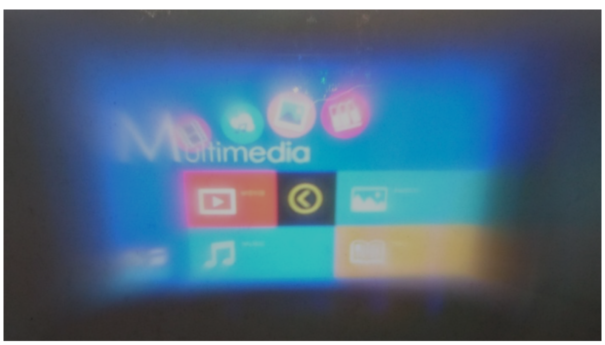


Figure 3.2.2.c (left) Image projected through single biconvex lens
Figure 3.2.2.d (right) Image projected through new modified 3-lens system

<u>3.3 Improvement to brightness of projector</u>

When we projected an image onto our special-made glass film under ambient lighting, the image was not very visible. Therefore, we wanted to improve on the light source of our projector.

Firstly, we need to understand how the projector works, to determine. The LED is placed at the back of the projector, and is shone through a reflector, which will gather and shape the beam forward. The light will then pass through a fresnel lens, thus collimating the beam. It then passes through a polariser, so that only polarised light (light in a single direction) can hit and illuminating the entire LCD panel. A fresnel lens then converges the beam towards the mirror, so that a converging beam will be reflected off the mirror into the projector lens. This is so that the entire reflection can be captured by the first lens through its aperture.

The original light source is a $24W$ COB LED. We purchased a $50W$ COB LED and a $50W$ LED driver that runs on $1.5A$ constant current. Higher wattage LEDs are commonly manufactured in 50W or 100W however the 100W LED would need a highly efficient cooling system which would take up too much space in our application, and also need to draw a large amount of power.

3.3.1 Comparing brightness of LEDs

Firstly, I compared the brightness of both LEDs. I used a multi-meter to measure the voltage across the $24W$ LED when it is driven by the projector's internal circuit, and the $50W$ LED when it is driven by its own LED driver. I also measured their currents, and used the power law $P = IV$ to calculate power drawn.

LED	Voltage (V)	Current (A)	Power (W)
$24W$	19.45	1.7	33
$50W$	30.2	1.5	45.3

Table 3.3.1.a Measurements and calculations for $24W$ and $50W$ LED

Then, I secured each LED in a black box, and attached the lux meter directly opposite the LED. When the LED is switched off, the reading on the lux meter is 0. I used a voltage generator to supply power for both LEDs, and maintained 1.5A for them.

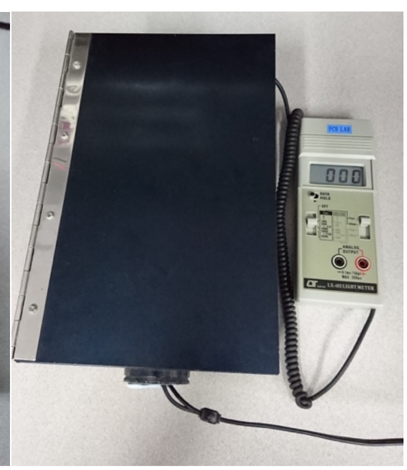


Figure 3.3.1.a (left) Setup of lux measurement, LED turned off, ambient lighting recorded as '103 *lx*'

Figure 3.3.1.b (right) Black box when closed, no light can enter

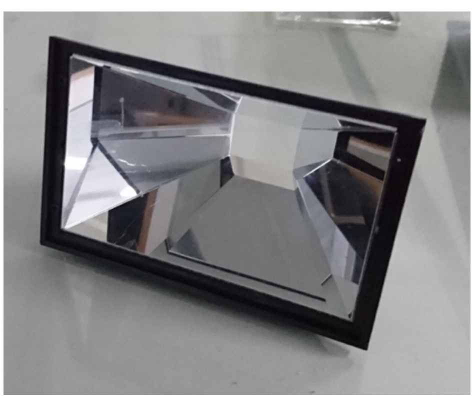
LED	Voltage (V)	Power (W)	Lux (lx)
$24W$	19	28.5	21500
$50W$	30.3	45.45	35000

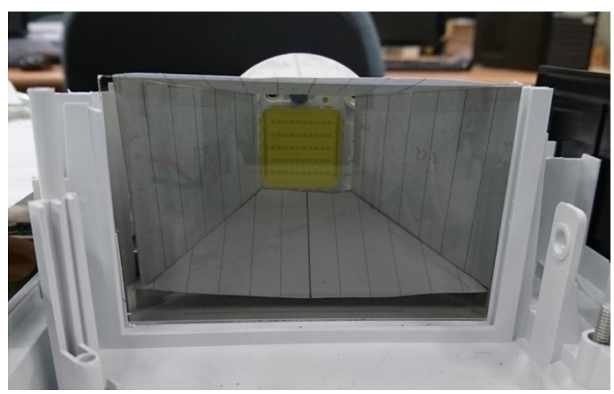
Table 3.3.1.a Comparison of LED and their lx

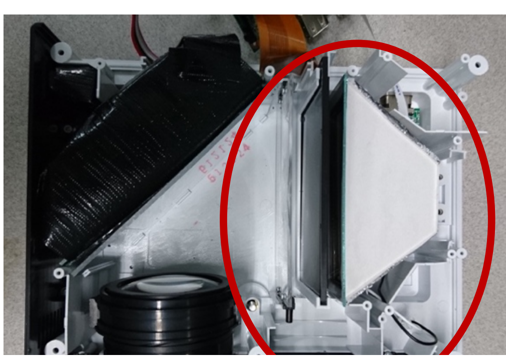
Lux measures luminous flux per unit area (lm/m^2). As the area is constant, the $50W$ LED's power of light (lm) is around 1.67 times stronger than the $24W$ LED, and therefore is brighter.

3.3.2 Integrating LED

The new LED was much bigger than the original LED. Therefore, I had to modify the projector's internal casing to make space for the $50W$ LED. I sawed away some of the plastic, and added an aluminum plate between the LED and the heat sink to improve contact for heat dissipation. I also added a few more IC chip heat sinks (small heat sinks) behind the main heat sink as the $50W$ LED would heat up very rapidly. After fitting the new LED into the projector, I had to construct a new reflector as the projector's original reflector is too small the new LED. I first measured the space available between the polariser and the designed a prototype out on paper, before moving on to thin acrylic. I then pasted silver reflective paper on the acrylic, and glued the four pieces together. Acrylic was used as it is more sturdy than other materials such as cardboard. Aluminum was also chosen as it's reflectivity is around 80% to 88% while acrylic mirrors which are used in the original reflector has reflectivity of 85% to 90%.







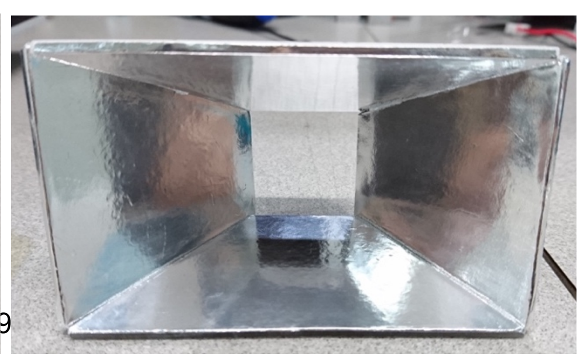


Figure 3.3.2.a (clockwise from top left) Original reflector, 3.3.2.b case fitting of paper prototype, 3.3.2.c final prototype, 3.3.2.d case fitting of final prototype

Figure 3.3.2.e (left) Image projected onto screen, illuminated by the new LED
Figure 3.3.2.f (right) The same image projected onto glass

The new LED and heatsinks fitted well into the projector casing together with the reflector. They were able to illuminate the LCD very well, and the resultant image was very bright when projected onto screen, although intensity is still reduced when projected onto glass.

4. Conclusion and Reflections

Modifications made to the optical components of the project were successful, however, improvements still can be made in its implementation.

This project has been very nurturing experience both in working towards achieving its initial objectives, and working with people with very different working styles and problem approach from myself. I have gained more knowledge in optics, particularly in the working principles of the projector. I have also experienced how it is like to do research and development, and pick up new knowledge through the resources available around us.

5. References

Figure 3-lens formula http://www-ferp.ucsd.edu/LASERLAB/TUTOR/lensdesign.shtml
Figure experiment http://www.optics4kids.org/home/content/other-resources/articles/lenses-and-geometrical-optics/

APPENDIX E VU VIET LINH'S INDIVIDUAL REPORT

Overview

Head up display (HUD) is defined as a form of transparent data display without requiring the viewer look away from their usual view point. In the current traffic environment, drivers often have to look away from the road to observe their velocity and other data on their vehicle dashboard. The big advantage of the transparent HUD system is that it projects information within driver's front view so they can keep their eyes forward to make a safer drive.

The project is divided into 2 parts: Projector – which takes the role of displaying data, and Processor – capture, process and transmit data. This report is the description of the student's own role in helping the team to achieve its goal in relation to the team report.

The task is to design a system which takes a car's data (vehicle speed, coolant temp, engine rpm…) as input, process it through a microcontroller and transmit it to the projector for display.

Group assignment:

- Projector: Pay Shi Yun Carin, Chuah Xiao Fen, Chong Chen Hwa
- Processor:
    - Wu Jitong: connection between devices
    - Vu Viet Linh: software programming
    - Cao Tongtong: hardware setup

Procedure

(Processor subgroup)

Week 1-2: Briefing and planning – assign groups

Week 3-5: Practice on hardware – OBD simulator/OBD cable/Arduino microcontroller

Week 6-8: Practice on Raspberry Pi – study Python/wxPython

Week 9-12: Build & finalize the complete program

Week 13: Work on final report

Microcontrollers

In this project, an OBD simulator was used to generates data instead of a real car. In order to capture and process the data generated by this OBD simulator, a microcontroller was used as the main processor, programmed to bring out specific tasks.

a) Arduino

Arduino is an open-source prototyping platform on easy to use hardware and software. By connecting the pins on board to specific devices, the Arduino can operate many different tasks due to user's programming. Arduino language is based on C.

In this project we manage to build an Arduino which can control two boards of 4-digits-7-segments LED simultaneously, displaying the data captured from OBD simulator. The LED digits were displayed in a mirrored manner so they can be observed through a reflection.

b) Raspberry Pi

Raspberry Pi functions as a small-sized CPU, which plugs in a monitor or projector and uses a standard keyboard and mouse. It has the ability to interact with the outside environment through many different channels (USB/HDMI/Ethernet cable/pins), which greatly simplify and enhance the connection between devices in our project.

In this project we use Python as the language to build our main program, including capture, process and display. The user interface is constructed using wxPython, a GUI toolkit for Python.

Py-OBD programming

The program will connect through the OBD-II interface, display the gauges available dependent on the particular vehicle and display realtime engine data to the cars aftermarket head unit in an interactive GUI.

· **obd_utils.py:** serial scan for available ports and return a list of serial names

· **obd_sensors.py:** decrypt and translate the data generated by OBD simulator into a list of sensors: Engine RPM, Vehicle Speed, Coolant Temp, Fuel System Status…

· **obd_io.py**: the heart of the whole program: capture data, control output and handle exceptions by sending proper commands.

· **obd_capture.py:** find supported sensors by getting PIDs from OBD. This is an executable file which can print out values of sensors in a Python shell

· **obd_gui.py:** user interface

❖ Different protocols:

The ELM327 Cable supports several different OBD protocols. For a real car, user may never have to choose which one it should use (since the factory settings cause an automatic search to be performed for you). However, since an OBD simulator was used in this project, we have to specify a protocol

Code snip: obd_io.py

```
debug_display(self._notify_window, 2, "atz response:" + self.ELMver)
self.send_command("ate0") # echo off
debug_display(self._notify_window, 2, "ate0 response:" + self.get_result())
self.send_command("AT SP 00") # telling the program to automatically search and save it in internal
EEPROM
debug_display(self._notify_window, 2, "atsp response:" + self.get_result())
self.send_command("0100")
ready = self.get_result()
```

❖ Autorun when start:

Since car drivers wouldn't have the device to control the Raspberry Pi – we embedded the program to automatically run when the engine/RaspberryPi is turned on

In /home/pi/.config create autostart/ScanTest (a text editor file):

```
[Desktop Entry]
Name=ScanTest
Comment=My Python Program
Exec=python /home/pi/ScanTest/Scan.py
Icon=/home/pi/ScanTest/Scan.png
Terminal=false
MultipleArgs=false
Type=Application
Categories=Application;
Development;
StartupNotify=true
```

- Sensors to be displayed:

Since the data was transmitted to processor in serial - a list of sensors info (name, value, unit) in binary data stream, it's impossible to restricted the acquisition of data into specific sensors. Therefore, to limit the sensors to be displayed down to 3 necessary sensors: Engine Rpm - Vehicle Speed - Coolant Temperature, which changes are performed on the obd_gui.py program.

This change is practically important since in real life environment (real car driving), no button is provided for driver to press on to switch the gauge. Another version of all sensors displayed are retained under the name obd_gui_fullsensors.py

APPENDIX F WU JITONG'S INDIVIDUAL REPORT

1. Project Overview

Windshield projection technologies could one day replace the current dashboard display due to its transparency and flexibility. We have seen a lot of scenes in the science fiction movies implementing this idea for information display and sharing, such as in a meeting or during driving. Nowadays, some of the research and development institutes have put funds and manpower to bring this idea into reality. For example, the latest version of Buick Lacrosse 2016 has integrated a glass display system in the car drive system. This head-up display (HUD) system could update speed information instantaneously in several layout designs which drivers are able to customize. Besides that, communicating with radar system, it helps to alert the drivers when pedestrians are walking around the car and assists to back off the car. Another independent electronic devices company, Navdy, also has developed a small HUD system. Apart from the basic car on-time information display and navigation, it also brings notifications from the apps on driver phone into view allowing you to keep your eyes forward.

The big advantage of the transparent HUD system is that it projects information within drivers' field of view so they can keep their eyes forward to make a safer drive. Also, making good use of the front glass space to project all the critical information which currently display on the instrument panel may lead the revolution and innovation on car internal design, especially for the driver's part.

In our project, the basic information projection system was designed. It aimed to provide three critical data for driving – vehicle speed in km/h, engine speed in rpm and coolant temperature in degree Celsius. These three data displayed on the screen at the same time and update with a preset time interval. To simulate the real car environment, a piece of glass with a special film was acted as the car windshield and a on-broad diagnose (OBD) simulator helped to generate the series of data representing the car digital control system.

2. Project planning

2.1 Grouping

The glass projection system design was divided into two parts. One part was to modify the traditional projector which was widely used in the classrooms and could project on the large-sized screens. The main objective for this part is to make the projector adapt to the car environment by adjusting the image size and light source. The other part focused on the data acquisition and processing. It included the data transfer system design and HUD GUI design. In order to launch the project effectively, the team was divided into two sub-groups of three students. Each sub-group worked on one part – namely optics team and software team.

2.2 Timeline

As part of the software team, at the initial stage of the project, I helped to plan the project timeline which is shown in the below chart.

Week No.	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Recess Week	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Project Planning														
Arduino Study and Demonstration														
OBD System Study and Design														
Raspberry Pi Initialization														
Python Programming Study														
LINUX Terminal Control														
OBD Data Acquisition and Processing														
HUD GUI Design and Modification														
System Integration														

Table 1 Project timeline of the software team

Project Planning – Brainstormed the possible approaches for the data transfer system. Listed out all the components needed in the project and sourced the suitable types and vendors. Came up with a rough project timeline.

Arduino Study and Demonstration – Researched on the basis of Arduino programming and control. Demonstrated the single 7 segment 4 digits LED display and double 7 segment 4 digits LED display. Understood the difficulties on using Arduino as the micro-controller unit (MCU) and looked for a more feasible MCU for future system design.

OBD System Study and Design – Studied on the working principle of OBD system and OBD simulator. Understand the communication protocols through the OBD simulator and OBD cable. Studied on the different connection means from OBD to external devices – cable and Bluetooth. Searched for data acquisition samples.

Raspberry Pi Initialization – Understood the work principle of the Raspberry Pi as the MCU. Installed the operating system and necessary applications.

LINUX Terminal Control – Studied on how to use the terminal window to write the commands.

OBD Data Acquisition and Processing – Amended the reference code files for appropriate data acquisition by defining the different serials and sensors, and setting up the communication between input and output.

HUD GUI Design and Modification – Designed the GUI to display the required data acquired from OBD simulator. Enable the system to auto-start when rebooting the system.

System Integration – Integrated with the optics system and demonstrated the projection on the glass. Made the necessary amendments for the better display quality.

3. Personal performance and contribution

3.1 Two approaches for glass projection

There were basically two approaches to project the information to the glass. First approach was reflecting the image from LED broad to the glass. For example, during the project demonstration as shown in the below, data was transferred to two 7 segment 4 digit LED display broad and controlled by the Arduino Uno. To make it reflect the correct number, the original data must be converted into a special order. The relationship between normal numbers and converted numbers have been discussed in the previous chapter.

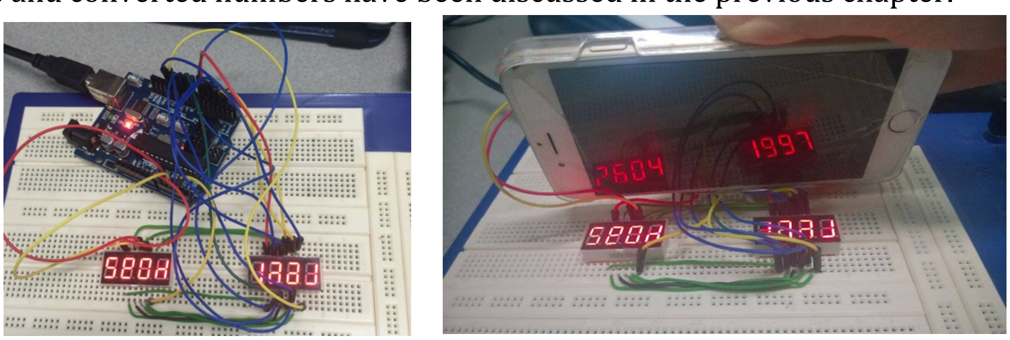


Figure 1 Data displaying with 7-segment 4-digit LEDs directly and with the reflection

The other approach was projecting the image from a projector directly. It provided the flexibility since nowadays the advanced projectors could received the graphic information from laptops, smart phones and etc. through HDMI cable and USB. In this project, Raspberry Pi MCU was selected to be connecting the modified projector as the source of information. In this way, the data could be acquired and processed on the Raspberry Pi platform first, and then transferred to the projector and displayed accordingly.

Figure 2 Projected image on the blackboard

3.2 Connections between components

From previous section, it was shown that two different approaches were used to achieve the data projection on the glass (screen). Since the MCUs and display media were different, the construction of the systems was different as well. Below show the structures of each approach.

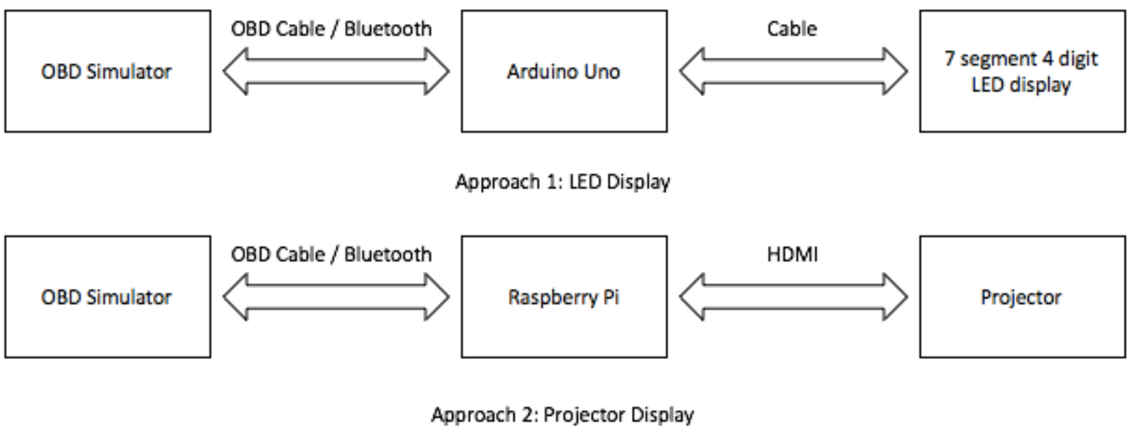


Figure 3 Two approaches of data display

3.3 Comparison on Arduino and Raspberry Pi

At the beginning, we considered to use Arduino as the MCU to process and transfer the data. However, after examining the project purpose and system configuration, we decided to use Raspberry Pi instead. Below provides the comparison between Arduino and Raspberry Pi. I will explain the main reason.

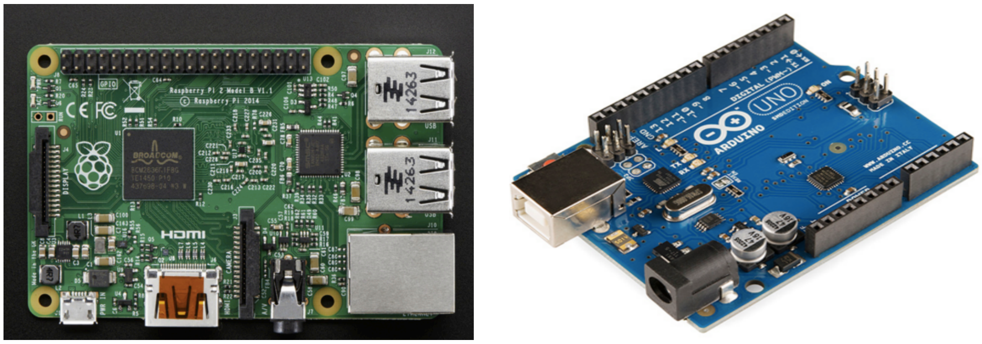


Figure 4 Raspberry Pi 2B Model and Arduino Uno

1. Arduino boards are micro-controller while Raspberry Pi is a fully function computer with LINUX OS.
2. The main purpose of the Arduino board is to interface with sensors and devices while Raspberry Pi is with a dedicated processor, memory, USB ports and a graphics driver for output through HDMI
3. Arduino's 16MHz processor falls a little short of the Pi's 900 MHz chip.

Considering these features, we decided to use Raspberry Pi for its easy connection with other devices through USB ports and powerful processing capability.

3.4 Terminal control

Giving commands in the terminal can simplify the control and be free from the mouse. During the project, we grabbed the opportunity to learn basic and useful commands under LINUX terminal to achieve our purpose. I listed out few commonly used commands as below.

- Sudo reboot: to restart the system
- Man XXX: brings up the online LINUX command manuals to help understand the usage of each command
- Pwd: Shows what directory/folder you are in. In Raspberry Pi LINUX, the home directory is /home/pi
- Cd XXX: Change directory
- Python XXX: run the python code in the terminal shell
- Ls XXX: lists out all the files in the current folder

For example, in the below shown command, the first command lists out all the files in the current folder and then goes to the desktop folder.

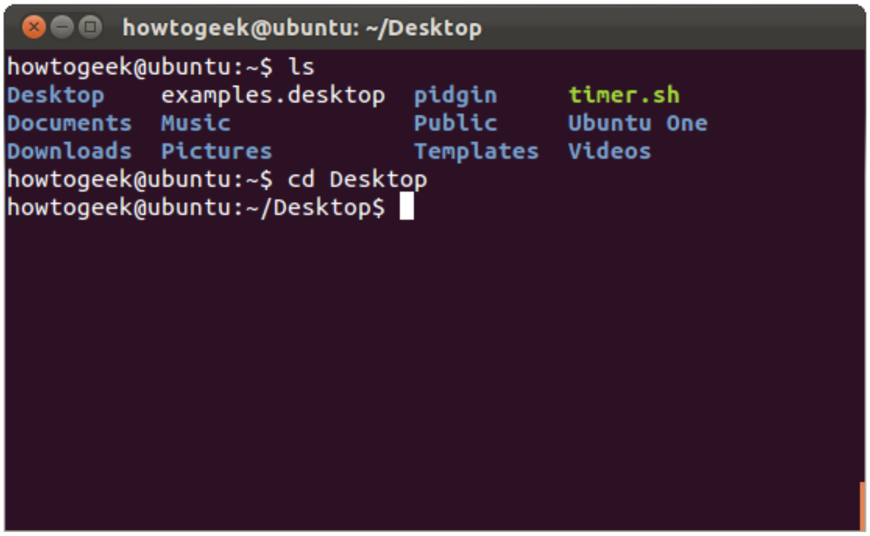


Figure 5 Terminal control example

3.5 Auto-Start

The GUI design in this project was basically the shell result of a programming file in python. During the modification process, it was fine to adjust the code and use mouse and keyboard to control the file running. However, in order to fit the car working environment, the auto-start of the finalized file was required. It was related to the rebooting setting of the Raspberry Pi. There were several methods to achieve this goal. In this case, a separated folder 'autostart' was created under /home/pi/.config directory and a command file was built within it and with the content shown below. Noted that .config directory was a hidden file. Hence, it must be directed into by typing the full address in the address line.

```
[Desktop Entry]

Name=ScanTest

Comment=My Python Program

Exec=python /home/pi/ScanTest/Scan.py

Icon=/home/pi/ScanTest/Scan.png

Terminal=false

MultipleArgs=false

Type=Application

Categories=Application;Development;

StartupNotify=true
```

The two important lines for this short programming were discussed as following.

1. Exec = python /home/pi/ScanTest/Scan.py: define which file will be run after system is rebooting

2. Icon = /home/pi/ScanTest/Scan.png: choose the icon image for this auto-start file

Upon rebooting the system, the system was able to open the destination file and display the desired information.

3.6 Data acquisition

Data acquisition could be divided into three steps. Firstly, the obd_io.py file defined and sent the command which contain the hex numbers from the device to the cable. Secondly, in the obd_sensor.py, hex values were converted back to the decimal values presenting different meaning data. Sensor names and units were also defined in this file. Last but not least, the obd_utils.py files enabled the Bluetooth and USB connections. It allowed the data to be transferred to the external devices.

3.7 GUI Design and Demonstration

Two phases of the information were shown after running the GUI file. First phase was the loading window. It was the initializing stage of the program which was in the white background color and displayed the project title and team members' names.

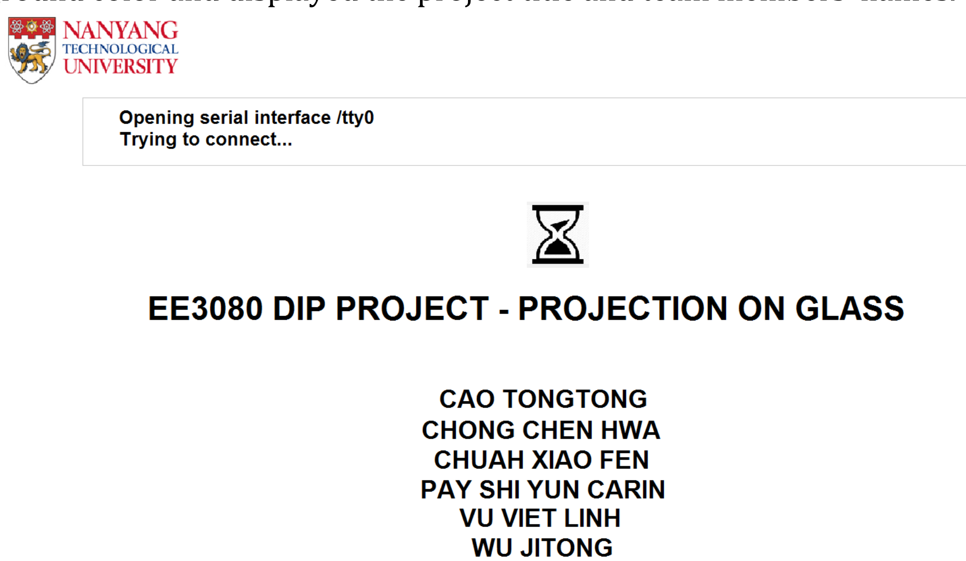


Figure 6 Loading panel

The second phase was displaying the required data – engine speed, vehicle speed and coolant temp respectively in the vertical order. For better display quality, background was in black while the content was in white, and later will be changed to blue series due to the sensitivity of the glass film used in the project. Within each display box, left hand side showed the number in a very big size while right hand side showed the units and data names. The data shown on the LCD screen of the OBD simulator could be used as the reference for data checking. Below two images shows that three data on projection glass and LCD screen were the same.

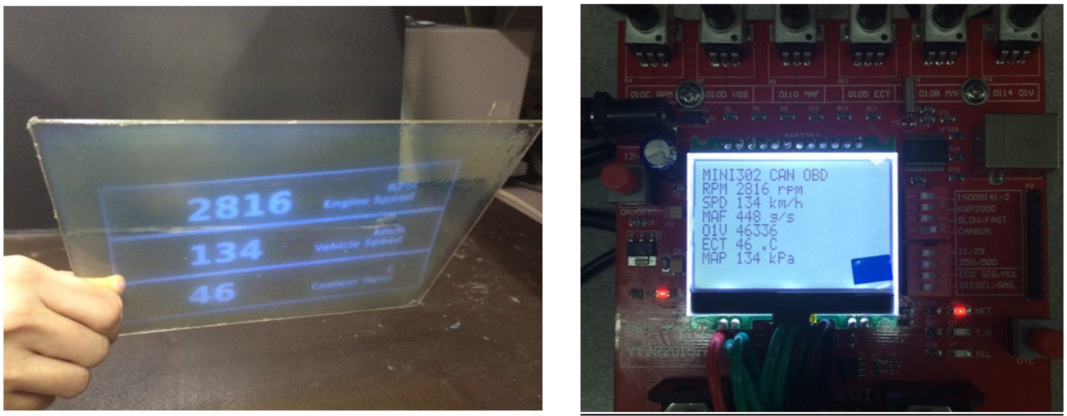


Figure 7 (left) Displaying data on glass. (right)Displaying data on the OBD simulator

When switching data on OBD simulator, it changed accordingly on both glass and LCD screen. Therefore, the data transfer and display ware reliable.

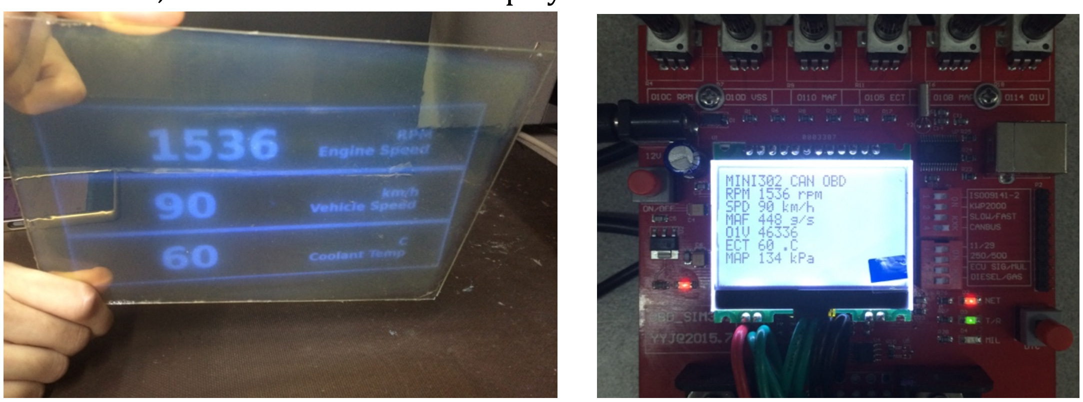


Figure 8 (left)Changing data on glass (right) Changing data on the OBD simulator

4. Self-reflection

As the project leader of the team, I helped to plan the project fund usage, coordinate the project execution and communicate with supervisor and mentors. Together with all the team members, we planned and discussed the project timeline, how to design the system and achieve the initial objectives. Along the way, we solved the technical issues and overcame the challenges together. We also have obtained a deep understanding of the optics system in the projector and data acquisition and processing flow. It was a great opportunity for the software team to learn the new programming languages which might benefit the FYP project and even future career. However, since the project was newly initialed from our team, we have spent a lot of time on project planning which lead to the time shortage in the last few weeks. Fortunately, the research time devoted will not be wasted, it could be a guideline for future DIP groups to continue on our project and begin at a higher starting point.

5. Reference

http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/
https://www.navdy.com/
http://www.pas.rochester.edu/~pavone/particle-www/telescopes/ComputerCommands.htm