High-resolution LED Headlamps Testing System with Web GUI

Abstract

Due to the advantages and security improvement brought by ADB systems and high-resolution LED headlamps, the high-resolution LED headlamps are becoming more and more popular among the markets and manufactures.

In the meantime, with the development of high-resolution headlamps, the controlling signals for headlamps are becoming more and more complex. A headlamp testing system can both adapt to this situation and have a lower cost is required.

In this thesis, a system based on Arduino and Raspberry Pi is presented. The system is intended to control and monitor headlamps of car from a website.

The main idea of this system is use Node.js framework to send CAN signals to car regarding to different user instructions at the same time receive the CAN message from the car to translate this message to visible data for the users.

The simulation and real car testing results are also presented in this thesis.

KEY WORDS: Rapsberry Pi, Arduino, CAN bus, Node, High resolution headlamps, LED lighting, car controlling

目　录

Chapter 1 Background

1.1 High-resolution LED headlamp

1.1.1 Concept of ADB headlamps

Headlamp technology has been evolving at an increasing speed in last thirty years. The first improvements concerned the light sources, then the headlamp components, and finally the lighting functions which have recently been melding and evolving into a driver-vision lighting system including sensors, ECUs, software and hardware actuators, and the headlamps themselves.

The development of headlamps is mainly in three aspects, new light sources, the lighting systems, and adaptive driving beam (ADB).

The ADB system is aimed to fix the problem between how to resolve the conflict between seeing and glare. High beam can provide enough lighting for the drivers but it could lead to glare of the incoming cars. In some circumstance, the high beam only used appr. 3% of the total night driving time, this indicates the potential safety improvement could be achieved by improve the lighting system.[1]

The ADB system in general has a camera, an ECU, lighting electronic controlling system and the corresponding headlamps.

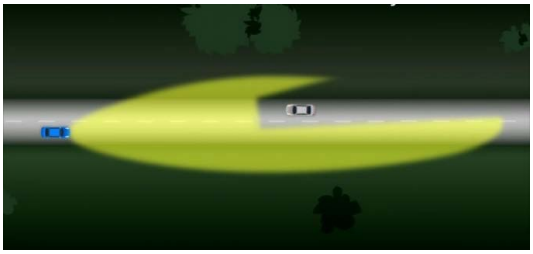

Figure 1-1 General idea of ADB system[2]

Figure 1-1 gives a general idea of ADB systems, the headlamps can automatic create dimming area to avoid potential harm to the incoming car. It can increase visibility distance and comfort in all driving conditions without increasing glare for other drivers.

The ADB system is more like a controlling system combines with lighting system. And ADB system relies on a complexes lighting system. The system should be able to switch from high to low in certain areas independently that is why ADB system leads to the need of high resolution LED headlamps.

1.1.2 Current Solution for ADB Headlamps

There are currently three solutions for ADB system. They are divided by the lighting systems and headlamps used to function with the controlling system[2].

Mechanical movement solution, it is using a mechanical switch to control the beam. This is the most economic and affordable solution, however, the lighting system is too simple that half of the light is lost in one side of the headlamp when dimmed. And a mechanical system is not as reliable as electronica ones.

Matrix beam, the main idea of this system is to create glare-free high beam with a fixed LED matrix

array. By switching off each chip of LED, the dimming area can be created. This is a higher-resolution system that gives additional possibilities such as lighting between shadowed-out vehicles. This solution has a very good smoothness, and provides the possibilities to shadow several cars crossing. Because of the electronica solution, the reaction time is quite rapid.

Pixel light, this is an even more high-resolution solution comparing to LED matrix, it provides in total 192 LED pixels. This would enable smooth change and can shadow multiple cars ahead while producing light between the cars in the meantime continues to provide light above the shadowed cars with a fast reaction time. But, this solution would be expensive and the development of the electronics will be very complex.

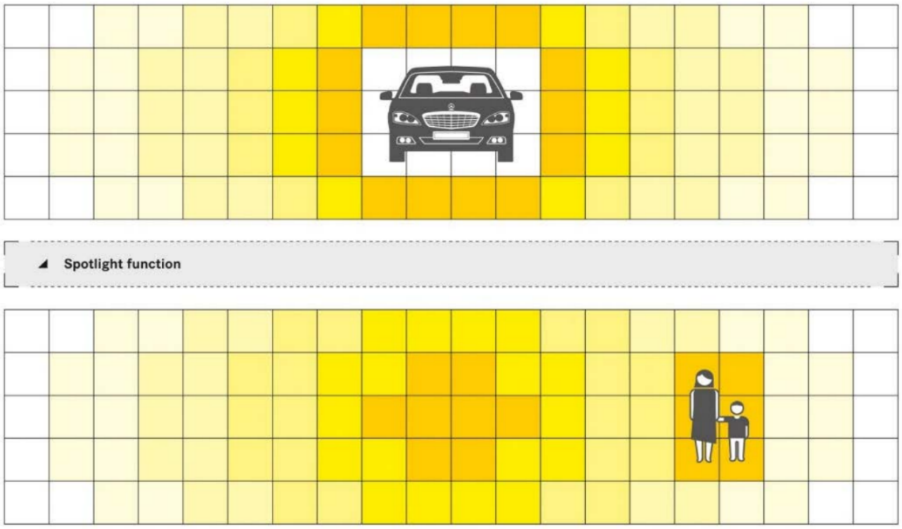

Figure 1-2 General idea of pixel light[2]

From the above three solutions, we can find out that the development of LED lighting systems is to becoming higher-resolution in order to provide more flexible lightning area and able to dynamic diming or lightning certain area according to the traffic condition. With higher-resolution the headlamps itself is surely becoming more and more complexes. Another trend is the electronic solution is better than the mechanical solution in all aspects expect for the cost. The perforce of lightning and reacting time, as well as reliability of electronic headlamps is much better.

After that, the electronica controlling part will also grow more and more complexes due to the demand of the headlamps.

1.1.3 Advantages of ADB Headlamps

The safety improvement of ADB system is also discussed. A few tests have been performed in tracks to show the safety improvement of LED headlamps.

From these tests, data proved that ADB LED headlamps can increase the security of the cars. The accident rate depending on speed may minimize form 74%@80km to 58%@100km/h when the glare free high beam is activated. [1]

The safety level increases due to the fact that the average light flux for illumination of the high beam

area above the cut off is increased by 33% compared to automatic high beams and that the illumination range can be excellent, even when other traffic participants and some areas have to be dimmed.

The data from a measurement performed on a 900 km track in both country side roads, additional highway and urban road proved the LED high beam activation time of more than 98% at speed levels higher than 60 km/h during night drive illustrates the safety benefit remarkably. [3]

LED high-resolution lighting is a new technology and because the security reasons, it still faces the law and social difficulties. The marketing success of high-resolution LED system relies on its security and the willing of acceptance of the drivers[4].

The development of high-resolution LED system also require new methods in both designing and testing. Some statistic method was presented to develop high-resolution LED system. From the consideration to test the system as well as show the system to the potential customs for marketing use, the high-resolution LED system has a completely new controlling system due to its variety of changes in light distribution, in general, the commands sent to the headlamps should contain the description of the dimming or lighting area for both sides, usually in angles, and sometimes also contains the lighting mode and strength of the light. The traditional testing methods requires a set of expensive equipment and a computer. This neither economic nor elegant for testing and demonstrating to costumes. The system presented in this thesis is aimed to solve this problem.

1.2 CAN Protocol

1.2.1 Brief introduction

A controller area network as known as CAN bus, it is a network standard with the intention to apply in automotive and embedded developing field. There is no center or host in this network to improve its reliability. The ECU or cameras, sensors, processing units can communicate with this protocol in vehicles. It is designed for automotive application originally but due to its high reliability it also has other application areas.

CAN is a message based and event driven protocol. And in general, CAN protocol is widely used in embedded developing areas, like in robotics or even in airplanes.

Robert Bosch GmbH firstly started the research of the CAN bus 1983. In the year of in 1986, it is at the Society of Automotive Engineers (SAE) congress where the protocol was firstly officially released. The first generation of controlling micro device followed CAN protocol was made by the companies like Intel and Philips. And in the year of 1987, the chips came into the market. [5]

The current version of CAN protocol is CAN 2.0 released also by BOSCH in the year of 1991. The International Organization for Standardization (ISO) released the CAN standard ISO 11898 in the year 1993.

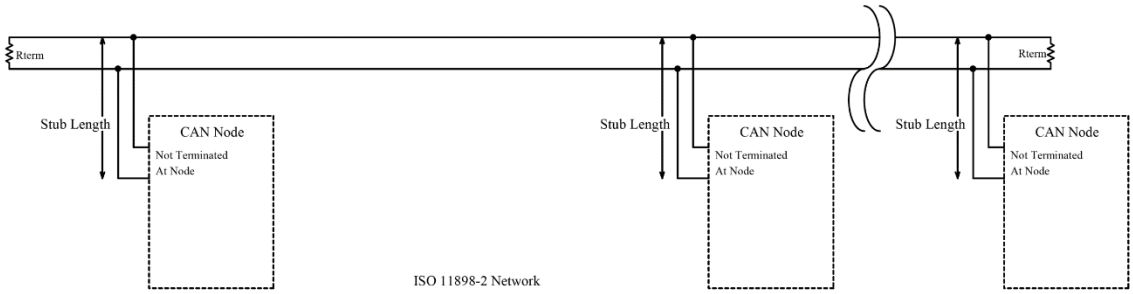

Figure 1-3 an example of CAN network

CAN network is a simple mix with a set of CAN nodes and the bus. All these nodes are connected by a physical transmission medium (CAN bus). As it shows in figure 1-3. In the CAN node, it usually contains a CAN controller and a Transceiver for the communication use.

Every single node follows the CAN protocol should have the ability of both send and receive CAN messages. The CAN node does not have the ability of both sending and receiving the CAN messages at the same time. A CAN message necessarily has a CAN ID which represents its priority. The specific description of a CAN message will be presented in the following contexts.

After the node, the CAN network also must have a bus to interchange signals, the physical signal transmission in a CAN network is based on voltages, or to be more clearly, by calculating differential voltages between the two voltages the node can get the signal. This effectively eliminates the negative effects of interference background noise which may be introduce by moto or other devices. Consequently, the transmission medium (CAN bus) consists of two lines: CAN high line (CAN-H) and CAN low line (CAN-L).[5]

Because of the application scene of the CAN bus, where safety is always the priority concern, like in cars or power chains. So it would be dangerous to assign responsibility for bus distribution to just a single bus node. If this node is no longer functional, the whole communication is cut off. A much more elegant solution is to decentralize bus access, so that each bus node has the right to access the bus.

That is the concern why CAN network is based on a combination of multi-master architecture and line topology: essentially each CAN node is authorized to place CAN messages on the bus in a CAN network. The transmission of CAN messages does not follow any predetermined time sequence, rather it is event-driven.

This is kind like ALOHA protocol, the bus is only bus when one node tries to send messages. This would allow quick access to the bus when a node wants to send CAN message. When using a CAN bus protocol to transmit messages, usually the applications require a real-time data transmissions to deal with emergency, while in this case, milliseconds data transmission is no problem in a CAN network, because of the ability to quickly react to asynchronous events and very high data rates up to 1 MBit/s.

CAN network uses a technology called receiver-selective addressing to prevent dependencies between nodes. Every CAN node is sending every single message to every other nodes connected on the

same bus, or using the term in computer networking to call it broadcasting. Despite broadcasting every message looks kind of inefficient, but this action reduce extra process when trying to add a new node to the network and also makes the whole network simple and reliable.

1.2.2 CAN Frames

CAN has four frame types: Data frame: the basic frame for simple data transmission between nodes inside a CAN network, Remote frame: the frame for request certain data transmission, usually by giving its ID, Error frame: the frame for errors, it can be sent by any kind of nodes, Overload frame: the frame simply used to manually create delay between other kinds of frames.

Table 1-1 is a simple description of the data frame, in this thesis, the base data frame is enough for the system.

Table 1-1 Fields in a base data frame

Name of fields	Length(bits)	Description
Start-of-frame	1	start of frame transmission
Identifier (ID)	11	unique identifier which also represents the message priority
Remote transmission request (RTR)	1	Set true when it is a remote request frames
Identifier extension bit (IDE)	1	Set 0 for base frame format with 11-bit identifiers
Reserved bit	1	Reserved bit. Set false
Data length code (DLC)	4	Number of how many bytes of data are in the data field
Data field	0–64	Data
CRC	15	Cyclic redundancy check
CRC delimiter	1	1
ACK slot	1	CAN transmitter sets 1 and any receiver can assert with 0
ACK delimiter	1	1
End-of-frame (EOF)	7	1

Chapter 2 System Overview

2.1 System Structure

The core idea of the system is to provide the user with a friendly and responsive design website for controlling the headlamps. This will require a web server which is capable of hosting a normal website. This system will process all user requests and send this commands to the car in the meantime, receive the status from the car and present them to the user. In order to communicate with the car, a micro controller is also required.

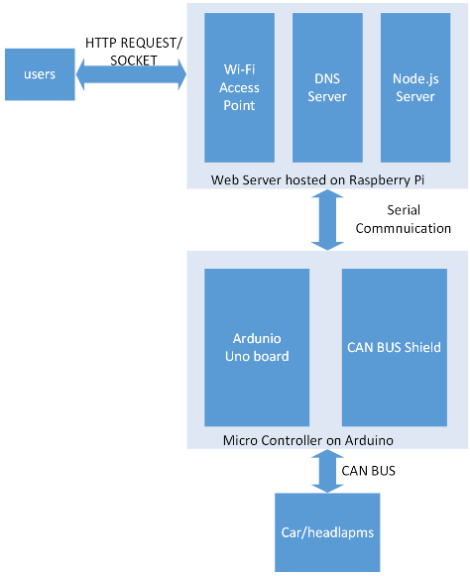

Figure 2-1 Structure of the System

Figure 2-1 CAN Controller and enables the system to communicate with the car, the CAN BUS gives a general structure of the lighting testing system. The system in total has two parts which are the web server and micro controller.

The web server is running on a Raspberry Pi which is an open source arm based tiny computer, powered by normal 5V USB port. It runs an arm version of Linux and it is possible to install Node.js environment on it. The server side code is based on Node and written in pure and simple JavaScript code. A Wi-Fi network is also generated from the web server so the user can directly connect to it.

And giving concern to the micro controller, an Arduino with Can Bus shield is added to the system. The micro controller is used to receive commands from the server and send it to the car, in the meantime, it should also receive messages send from the car and get the data from them in order to send back to the server.

The Raspberry Pi and Arduino are connected by USB cable. The USB cable is both used for communication between the two parts and power the Arduino.

The Arduino is connected to a CAN BUS shield. The CAN BUS shield is based on MCP 2515. This chip is a CAN controller and enables the system to communicate with the car, the chip itself communicate with the Arduino with SPI interface.

The website and micro controller are communicated by serial communication.

2.2 Hardware Usage

For the web server, a Raspberry Pi is chosen to be the server. Besides, a USB Wi-Fi adaptor is also required for the system to provide access over Wi-Fi network, through some configuration, the web server can also work as a router and generate its own Wi-Fi network and thanks to that the external Wi-Fi network is no longer required.

Raspberry Pi can provide a desktop like environment; it is more like a little computer than just an embedded controller. With an Arm version of Linux, it is possible to host a normal website in this low cost system. This can save time on developing and testing because it is possible to directly run the same code on Raspberry Pi and a normal windows or mac computer.

Raspberry Pi provides two USB ports, which are enough for this system, one for USB Wi-Fi adaptor and one for Arduino. The power supply on Raspberry Pi is sufficient for powering the system and the micro USB port is the only power supply required for the whole system.

An Arduino Uno and CAN BUS shield is used for micro controller. The CAN BUS shield is provided by SPARK FUN. It uses a MCP 2515 chip as CAN controller. The CAN BUS shield can be directly plugged into the Arduino Uno board. Its manufacturers already provide the code used to power this shield. The only concern will be the content in the CAN messages. The technique used to transfer information will be introduced in the following chapters.

2.3 Communication between Layers

The communications between layers are the most important parts in this project. The system in total is divided into two components, the web server and the micro controller. Apart from that, the micro controller will be connected to the car to control the headlamps. And the web server has the ability to generate Wi-Fi network for every user to get access to the lighting testing system. For the user and the server, they are connected by Wi-Fi network, the user can visit the website hosted on the server.

As it shows in Figure 2-1, two different methods are used in interacting with the user. The users' commands are sent by HTML get requests and the temperature data are sent to the user by socket due to its real-time demand.

The sever will automatically search for the Arduino and try to establish a serial connection with Arduino. The serial communication is used for all the data exchanges between the micro controller and the web server.

In this circumstance, the Arduino will get the temperature data from the car and compile the data into certain format and then send it to the web server by serial communication. With the same principle the communication also functions in the other direction, which means the web server will also send the command of their users by serial communication.

And the micro controller should have the ability of sending and receiving CAN messages. By a CAN

cable, the system is connected to a car, the car will keep sending can messages to the system and the system will send different CAN messages containing the control signals according to the users' commands.

Chapter 3 Developing History

The idea of how to implement this system have changed during the time of developing. These ideas of how to develop this system will be discussed in this chapter. This would give a clear idea of why the current solution is adapted and its advantages comparing to other solutions.

3.1 Arduino with Matlab Solution

This method has been presented at the very beginning of the project. Because Simulink and Matlab provide native support for Arduino and Wi-Fi shield and meet all requirements. So it seems to be a decent choice to realize the whole system by this technology.

The hardware requirement for this system is Arduino alone and its Wi-Fi shield plus CAN BUS shield. The cost is actually higher because the Wi-Fi shield is more expensive than the Raspberry Pi B+.

The first burden of system is that CAN BUS shield and Wi-Fi shield use the same pins. Even if SPI interface can be shared the SPI chip select pin of the shield still need to be modified to make them work at the same time. A simple attachment of the shield is not possible.

Since the Simulink software does not provide existing model for webpage. A huge number of S-functions like html pages server must built to achieve this system. The official Mathworks Arduino supporting packet is using an old version of Arduino IDE, which contains known issues with Wi-Fi client function. The website base on Arduino is neither stable nor efficient.

In this plan, even the basic html page only contains several lines of html codes costs seconds to access. Besides, Simulink does not provide official support for CAN BUS shield, which should be powered only by S-function. This means the major part of the whole program will be built by S-function and this will lost the advantages of building program with Simulink, graphical programming.

Building pure C code to S-function is making things more complicated. Because Arduino Uno does not support real-time feedback to Simulink, the whole programming is like a black box. It is not convent when considering debugging and testing of the whole system.

The conclusion is this route map is not the best way to achieve the system.

3.2 Arduino Alone Solution

Working with Arduino IDE, makes program with the system more directly and simple. The website works fine but the problems of delay still remains. Arduino is only a micro controller, hosting a website is not what it is supposed to do. The website will become even slower if JS and CSS file are added to the website.

And as we all known, Arduino works in a loop model. This means all other actions would be blocked if one action is being done. In this circumstances, dealing with network request is a very time consuming action and this blocks Arduino from any other actions including sending and receiving CAN signals. Blocking the whole system for seconds is not a pleasant thing when plenty of signals need to be transmitted. It is low efficient and even unstable.

Changing the hardware to Arduino because of a faster 32 bits arm processor will reduce the time for dealing with network requests and maybe help improve the performance of the whole system. However, due to uncertain reasons, in this case, the Arduino will lose connection with the Wi-Fi shield after a few minutes even with the official demo. This force me to reconsider that Arduino alone may be not the best solution for this system. Because Arduino is only designed to be a micro controller more than a real pocket computer. Hosting a rich-text website is not the best application for it. It cannot provide even basic web server functions but only write a simple string response to the client. In this case, a mobile first responsive design website with JavaScript interactive assists is required.

A Raspberry Pi is considered as an alternative solution for hosting a website because Raspberry is using a normal Linux environment, perfect for hosting website, and a standard USB Wi-Fi adaptor will guarantee its stability. And Raspberry Pi B+ is even cheaper when comparing to a Arduino Wi-Fi shield. And using a Raspberry Pi to host website also solve the problem of collision in pins because the only need is CAN BUS shield.

The remaining question is how to make Arduino and Raspberry Pi working together. And the technology used for communication between layers will be presented in following chapters.

Chapter 4 System Components

4.1 Web GUI

The web GUI is designed to be an interface for the user to test the headlamps. It should be able to give useful data to the testers, and give a responsive design website for the switchers and sliders to control the headlamps in the cars. There are three demands for this website. First of all, it should be able to fit any resolution because testers may access this web site through tablet or smartphones. Secondly, it should provide a real-time and pleasant visualization of the data feedback from the car. Last but not least, interactive logic and a user-friendly slider is also very important for the testers to give command. The following context will give a clear and detailed idea of how this website is designed. In order to build a responsive design website, Bootstrap css framework is used in this website. Because of the tag provided by Bootstrap, the website can easily replace the content according to different size of the screen. By setting "col-xs" and "col-sm" in tag to appoint the layout in different resolutions, the website will automatically fit.

4.1.1 Data Visualization

The system will receive the temperature data from the cars. This data rely strongly on real-time otherwise, the data will make no sense. Giving concern to this, d3 framework and socket.io is used to present data. Socket.io is used for real-time data transmission. It is driven by event. The website will initialize the socket and establish socket connection with the server when the webpage is loaded by user.

A socket.on event trigger is set in the website, and it is binding to the function used to update the gauges. When the server receives data from the micro controller, it will emit the data in no time, and the website will react to this event by updating the gauges using the new arrived data.

The d3 framework realizes the gauges. The style of the gauges is provided by official website of this project.

D3.js is a JavaScript library provides special and useful function to organize the website based on data and to present data in elegant form. In this website, the createGauges function will be called when the page is successfully loaded by the client. And as soon as the new data emitted by the web server arrives, the socket event handler will pass the two arguments to the update function to update the gauges. The transferred data contains two different fields in a json format, the name of the source of temperature and the actual value of it.

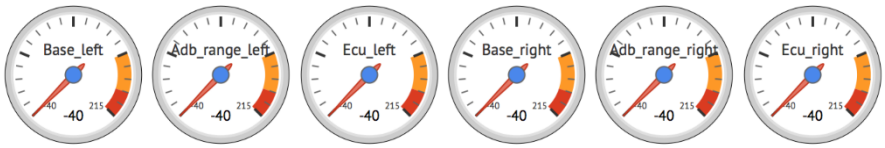

Figure 4-2 (a) The gauges created when loaded

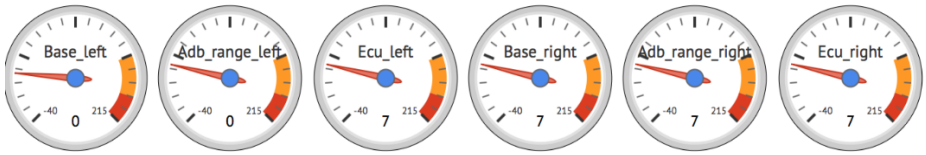

Figure 4-2 (b) The gauges after receiving the data from the web server

4.1.2 Interactive Logic

The webpage uses slider to set the values for the configuration of the headlamps; the sliders are created by the JS library bootstrap slider. The slider will follow the configuration preset in the <input> tag specified by the designer.

Event trigger is added to get the exact value of the slider, the value will displayed in a textbox, and fix to a certain format. The user is also granted to modify the value more accurate in the text box. The textbox value attribute is set by document.getElementById() value function and this change is binding to the slider onChange event. As it is shown if figure 4-3, each time the slider moves, the value of the textbox will be refreshed and the data is to fixed length.

Because the car side programs' requirements, an enable signal must be sent in order to remind the car this system is enabled.

In order to make this website act more natively for the user; the HTTP request is binding to the slider's stop event. In order to avoid any potential confusion of the users or other uncertainties, even the GET request is sent by the slider, and the value sent to the web server is still get from the text box.

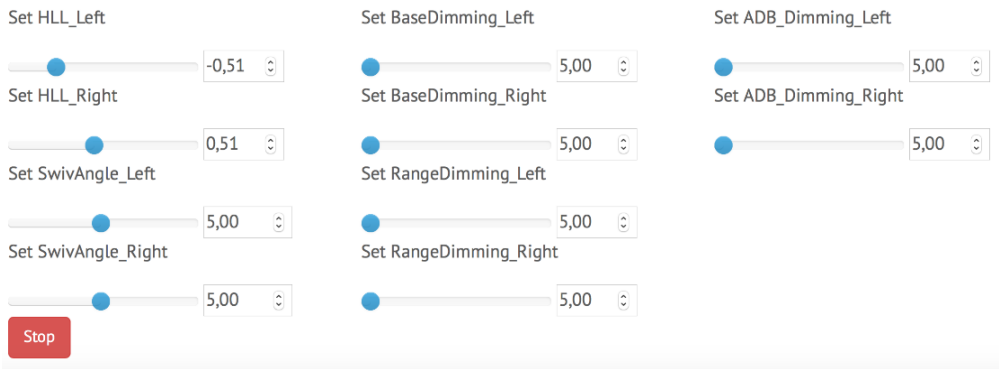

Figure 4-3 The data in textbox when moving the slider

A global variable is added to the website to monitor and log the situation of enable signal, so press the enable button is not necessary when starting the system. If the user use the slider before the activation of enable button, the enable request will automatically be sent to the web server to ask the server to send enable CAN message.

A feedback can also be alerted to the user after the transmission of GET request, the website will wait for the response from the web server. When the response is arrived from the web server, a message will be displayed on the website, otherwise, an error message will be given to the users.

Apart from that, the web page is divided into two tab views. One for headlamps direct drive mode,

and one for AFS and COL mode. The enable commands for each mode are unique and have different data field, to deal with this situation, an onclick event is added to the list view so when user acts to do a switch between tabs, the requests will be automatically sent to the server to stop the former enable command and start the new enable in this mode.

In COL and AFS mode, two selection boxes are added to the web page to let the user decide which mode of COL should be used. The description of all these signals will be discussed and presented in the following chapters. For the selection boxes, an onChange function is also added to the text boxes to send requests after each selection. During AFS mode the two sliders will be disabled because in AFS mode the ADB_Left_Edge and ADB_Right_Edge are not used.

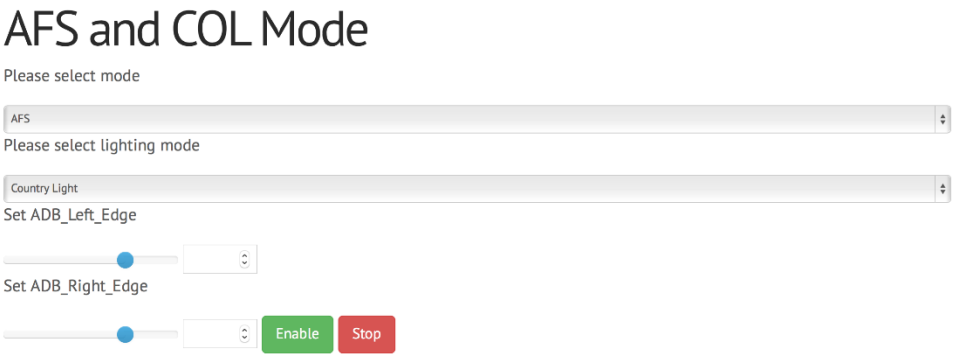

Figure 4-4 The AFS and COL mode controlling panel

While the mode is switched to COL mode, the slider will be enabled, and as it is described in the former text, the system will response right after the slider change and the green enable button is not necessarily to be pressed.

The stop button in each page will send the same request to the server and the sever will stop all the current action and sent a stop message to the car. The button is designed in case of uncertain actions.

4.2 Web Server

4.2.1 Node.js Web Server

An arm version node.js is running on the Raspberry Pi served as a web server framework. The control logic of the system on the Pi is written in pure and only JavaScript code. A few modules are added to the node to achieve the goal of this system.

Including the node-serial dealing with serial communication and express dealing with HTTP request. BodyParser is also used in the program to pray the data got from the request to a proper format to use.

The node.js server follows the principal of event driven, so each request won't block the main program.

The node-serial provides function to search all the USB ports, and an array contains the information of each com port will be given back. The program can search for the Arduino and establish the serial connection with it.

After that, the JavaScript functions to deal with different get requests are defined. The webpage will send different action requests to different address. And the server can get the complied data form the get request.

Socket.io is used in this server, it will listen to the socket when a user is connected and emit socket data when new temperature data is arrived from the Arduino.

The node server will automatic run and listen on the port 8000 as specified in the code, because this won't require root permission to start the program. However, if a server is not running and listening on port 80 which is default port for HTTP request, user cannot direct access by its domain or IP address, instead, they have to add the port number like: 192.168.1.1:8000. This would make users uncomfortable. So the comprise between the two would be setting a port forwarding in the Linux built in firewall, iptables, this action will redirect all requests trying to access port 80 port to port 8000. By this way, user can access to this website directly by its domain and IP address. And the sever script is also added in init.d folder to make it auto boot when power is up.

4.2.2 DNS Sever

In Internet, each IP address cprresponds to different domains, IP address is not easy to remember and not elegant to use, it is usually only numbers with dots. In order to make the users feel more comfortable when they trying to visit this testing website, a DNS server is also added to this little server to enable users directly access to this website by test.hella.com or any domains they want instead of the IP address.

A software called dnsmasq is used for setting up DNS server in Raspberry Pi. The function of this software is actually more powerful, it is not necessary in this case. The dnsmasq is only configured to resolve the domain to Pi's own IP address, because the Wi-Fi network is generated by Raspberry Pi itself and the IP address of the server itself is fixed. The configuration does not need to change. In this case the headlamps testing website can be viewed by test.hella.com.

Figure 4-5 The website can be accessed by domain instead of IP

4.2.3 Wi-Fi network

Because the testing environment in cars cannot always have a Wi-Fi router. It will be great if the Raspberry Pi can generate its own Wi-Fi. Apart from that, generating its own Wi-Fi can make the server to have its static IP address instead of a DHCP dynamic one. By this way the server will be more stable and easy to access. Surely the static IP and MAC binding won't apply to any server. If the web server gets a different IP address each time when it is connected to the network. It will be confusing and inefficient for the users to look up its IP address each time before using it.

Based on these reasons, it is be a good idea to make the web server to set up its own Wi-Fi. Luckily, Raspberry Pi is using a standard USB Wi-Fi adaptor. It is possible for it to generate its own Wi-Fi.

Hostapd is used for Wi-Fi generation. The software is a Linux daemon process for creating access

point. That means it can turn Raspberry Pi into an access point that other computers can connect to. It is also possible to add a password to the network. Hostapd only make the web server generate the Wi-Fi network. This is not enough for this system where the Wi-Fi network still have some packet forwarding job, so in order to allow users to connect to the website, an IP address is required for each device. A DHCP server is also required for distributing the IP address.

Isc-dhcp-server is the Internet Systems Consortium's implementation of a DHCP server. After installed Isc-dhcp-server and all these configurations, the users are able to access to the web server's Wi-Fi network and visit the website by a normal domain. The IP forward option in the iptables is not turning on because the web server is not real served as a router.

Figure 4-6 The Wi-Fi network generated by web server

4.3 Micro Controller

The main task for a Micro Controller is served as a translator for the server and the cars, because the server cannot be directed connected to the cars. The Micro controller should be able to send and receive CAN messages. It can provide some kind of interfaces for the web server to give orders to it and can get necessary information from the CAN message received from the cars, send it back to the server. Regarding all the reasons, Arduino Uno is chosen for the micro controller, because it has an existed CAN BUS shield and the programing of Arduino is very common, Arduino seems to be a decent choice in the implement of the system.

4.3.1 Arduino

An Adruino Uno with a CAN bus shield from Spark Fun plugged on board is used as the micro controller. The program on the Arduino side has two tasks, one is to power the CAN bus shield, to make it functional, and the other one is to deal with the data received from both sides, the web server and the car and help them exchange the necessary data and information.

The Arduino can receive CAN messages and get the data from them and transform them into a readable data then use serial port to print it to the server.

In the other way, the web server will give orders to the Arduino when a CAN message need to be sent. When a CAN message need to be sent, the web server will use the port to print a string contains all the data to the Arduino, it contains the CAN ID and the DATA field inside a CAN message, the data is already complied into the CAN message HEX form. Once Arduino receives the serial message, it will get the first two numbers as a CAN ID, and directed use the HEX data containing in this message.

4.3.2 CAN BUS Shield

The can bus shield is provided by SPARKFUN. It uses a stand-alone CAN controller with SPI interface, MCP2515.

It uses special CAN cable when comparing to the normal vector ones. The CAN high is connected to number 3 and CAN low is connected to number 5. So a special CAN cable should be made to make this system working.

Table 4-1 Pin distribution of Can Bus Shield and Vector Can Case

Designation	Vector pinning	Can Bus Shield
CAN-H	Pin 7	Pin 3
CAN-L	Pin 2	Pin 5

A simple attachment of CAN BUS shield is only possible for Arduino Uno; in this case it is enough.The connecting of the CAN BUS shield to the cars and to CAN case will be discussed later on.The CAN BUS shield uses a SPI connection to transfer data with the Arduino.

Chapter 5 Communication between Layers

5.1 Web GUI with Server

The web GUI is stored and hosted in the web server and different technologies are used in the data transmission between the web server and the web GUI, because different demands are presented in different data transmission.

The server will receive temperature data from the micro controller. And after successfully receiving the necessary data, the server will push this data to the website to refresh it.

This will require a real-time broadcasting to every device and do not need any feedback from the website. While the action code from the users, usually contains exact data and command to the headlamps, is transferred less often but require a feedback of receiving this command or not.

These two are typically different scene of usage in Internet. The first one is a socket, and the latter one is a HTTP request.

5.1.1 Socket.io

Socket.io is an open sourced real-time messaging framework. Its main goal is to deal with the demand of real-time communication. It also supports almost every platforms ranging from Chrome, Safari to IOS and Arduino. Apart from that, it is also light and fast.

Socket.io provides users identification service, this feature is not used in this system because in most circumstance, the user is supposed to be only one person during the demonstration or testing. But this feature provides a possibility to make this controlling system a multi-users one.

One user can be a demonstrator and other users like colleagues or customers can also access to this website. When the demonstrator is using the slider to give commands, the exact parameters will be sent to the server, and the server can broadcast these data to others users, so others can see exact parameters in their own device. It is possible to synchronize all the websites to present the same data in this system. But this feature is not added in the current version of the system.

In this system, the socket.io is initialed both in website and in the server.

The socket will be established when a user accessed to this website. Several events are available for the socket, in this program, it is pretty simple, just log the information of new user connected and emit data immediately when receives.

5.1.2 HTTP Request

The Hypertext Transfer Protocol (HTTP) is designed to handle the communication and data exchange online and it is widely used in the Internet applications no matter web browsers or other applications. The basic and simple client and server model is used in this protocol, the terms client usually refers to the web browser.

HTTP defines methods to describe the functions the clients used to get certain resource stored on the

server. The clients send requests to the server, and the server answer the request with a response.

And HTTP requests have four different methods and GET requests are used to get the specified resource with query data. This fits the scene when the user sends commands to the server, the GET request itself contains the data of the commands, including different parameters setting by the users like the diming filed of the ADB system or in which mode it should operate. The parameters in GET request is compiled and sent in a format of query string, including the name of the data and the value of it. And each requests will get the respond, the respond is defined by the server, in this situation, the respond is served as acknowledgment, the client side code can read the respond to see whether the server have received it or not. And if a correct response is received, the client will give feedback to the user. And the button will be blocked by loading state until a response arrived, a success alarm will be given to user and button will be back to normal when the process is finished by the ACK from the server.

5.2 Web Server with Micro Controller

This section is the most important part of the whole thesis and this is actually the most time consuming task in the whole developing process. The web server needs to exchange information and communicate with the micro controller, and because Arduino only get one USB cable for programming, it will be a good choice to directly connect it to the web server. And this will enable the Arduino to communicate with the web server by serial communication. Each side will send strings and deal with strings to get data.

5.2.1 Serial Communication

In information engineering and computer science field, the term serial communication is sending data in a time sequence and once a time. The concept serial is in direct correspond to parallel communication. And serial communication is widely used in this kind of embedded developing to give string forms feedback to the users when the developing board is connected to a host computer.

Arduino can use Serial.print function to print any messages to the serial, Serial.read to read from the serial. While on the server side, a node module named node-serial is introduced to this program. It enables the node server to deal with serial port in the computer and can both read strings from the serial port and write strings to the serial port. So to be brief, the communication based on serial port is based on strings.

An extra parser option is set in the server side code and this option makes it function as readline.

5.2.2 Initialize and Obtain Connection

The node-serial provide a function to scan the port available in this device. On the server side, serialPort.list function can return an array containing the information for every device connected to the server. So it will be easily to tell which device is an Arduino. The program will search the whole array and get the port.manufacturer area, if this area equals to Arduino. This means that this port is connected to an Arduino and the name of the port will be saved for future use.

When the server get the port name of the port connected to the Arduino, it can start to establish the connection.

From the Arduino side, it will try to establish a serial connection in the setup function, because the Arduino is running by time sequence, it will not start until the server find it and try to start a serial connection. Baud rate on both two sides should be same in order to obtain connection.

5.2.3 Compile Data to HEX

After two way serial port connection is built, the question will be how to transfer data between each side.

The CAN protocol is much more simple than the normal network protocol, micro controller is not expected to receive any kind of responds from the car when it sends commands to it. So when the server is printing commands, it is just like a single connection.

Different solutions can be used to deal with the data transfer. When the server receives the data contains exact parameters from the user, it can send this data to the Arduino and let the Arduino assemble it into CAN message. But it is easy to see this solution is not efficient nor decent, because if we write the data and its name to the serial port, the total length of the string printed is far longer than the exact data itself. The limit of the serial communication is set. Besides, if using this solution, each time if the data structure is changed because of new commands added into the website, the code in the micro controller will need to be changed at the same time.

So, it will be more efficient if the data can be transferred directly rather than using the long string. When sending a CAN message, two fields are necessary, the first one is the CAN ID, and the following one is the data field. The length of data field is 8 bytes (64bits) maximal. Each byte can be expressed into two HEX numbers.

It will be more efficient for the webserver to directly send the content of the CAN message instead of the original data to leave the work and let the micro controller do it.

The web server will receive the order from the user and transfer this data into a string, and the string should contains the CAN ID and the data fields of the CAN message. With a terminal symbol, ’/n’, the micro controller can divided each piece of CAN messages. From the micro controller side, it only receive complied string, and got the first two numbers as the CAN ID and the rest 16 numbers as the content of the CAN message.

The micro controller will transfer this HEX string into real hex value. Because the HEX string is in ASCII code instead of a real HEX value, this value cannot be directed filled in the blanks of CAN messages. Because in CAN message it is the real value.

The convert between string and real HEX is simple, for the numbers only need to minus the ASCII code to get the real data. For numbers, the micro controller will minus ‘0’ and for characters, it will minus ‘A’ and plus 10.

This method have an extra benefits, if the data is compiled by the Arduino, another disadvantage is because CAN messages should be sent by circle time, an extra delay will be required in micro controller due to the cycle time, because Arduino is an only-thread 8 bit controller, the delay or sleep function would block the whole program from running until the delay time is completed, and if several messages is sent at the same time, they will affect each other, and the regular check to receive CAN messages will also be blocked, the update of temperature data will be stopped due to the delay. The cycle time of each message will be far from accurate.

But in this solution, because the CAN message will be immediately sent when it receives the serial string contains CAN ID and data, the timing work is transferred to the web server. This action won't cost any time and the cycling time is more accurate. And all sending and receiving actions will not interfere each other. And if the server want to stop CAN messages, all it need to do is just stop printing the CAN message. The stop action will be performed immediately without any delay.

JavaScript is a script programming language which is relatively high level programing language when comparing to C or C++. It does not support pointers or some directly modify to the memory. The normal process to form a raw data packet in C or C++ would be create a memory space and uses pointers to directly fill in the blanks. However, this is not simple in JavaScript, because JavaScript has dynamic type and that is why usually the users do not care exact size or bitwise things when using it.

That is the reason why in this system, the raw HEX data is not transferred directly to the micro controller while a string is used to symbolize the data.

So when the server receives the exact command from the user, it has to get the parameters for each field, and compile them to the CAN message data field, and then send it to the micro controller.

Take the following CAN message as an example. This is a Headlamps dimming CAN message with a CAN ID of 0x601. Its length is 8 bytes, 64 bits, and the data is actually divided into a few fields as it shows in table 5-1.

Signal	Start bit	Length (bit)	Byte order	Value type
Base Dimming Left	0	10	Intel	Unsigned
Range Dimming Left	10	10	Intel	Unsigned
ADB Dimming Left	20	10	Intel	Unsigned
Base Dimming Right	30	10	Intel	Unsigned
Range Dimming Right	40	10	Intel	Unsigned
ADB Dimming Right	50	10	Intel	Unsigned

Table 5-1 signals contained in headlamps dimming CAN message

In figure 5-1 is a real distribution of the exact data field contains in a CAN message. Each of this parameters are set by the users. By the slider in the website, the users can easily set the value of each parameter.

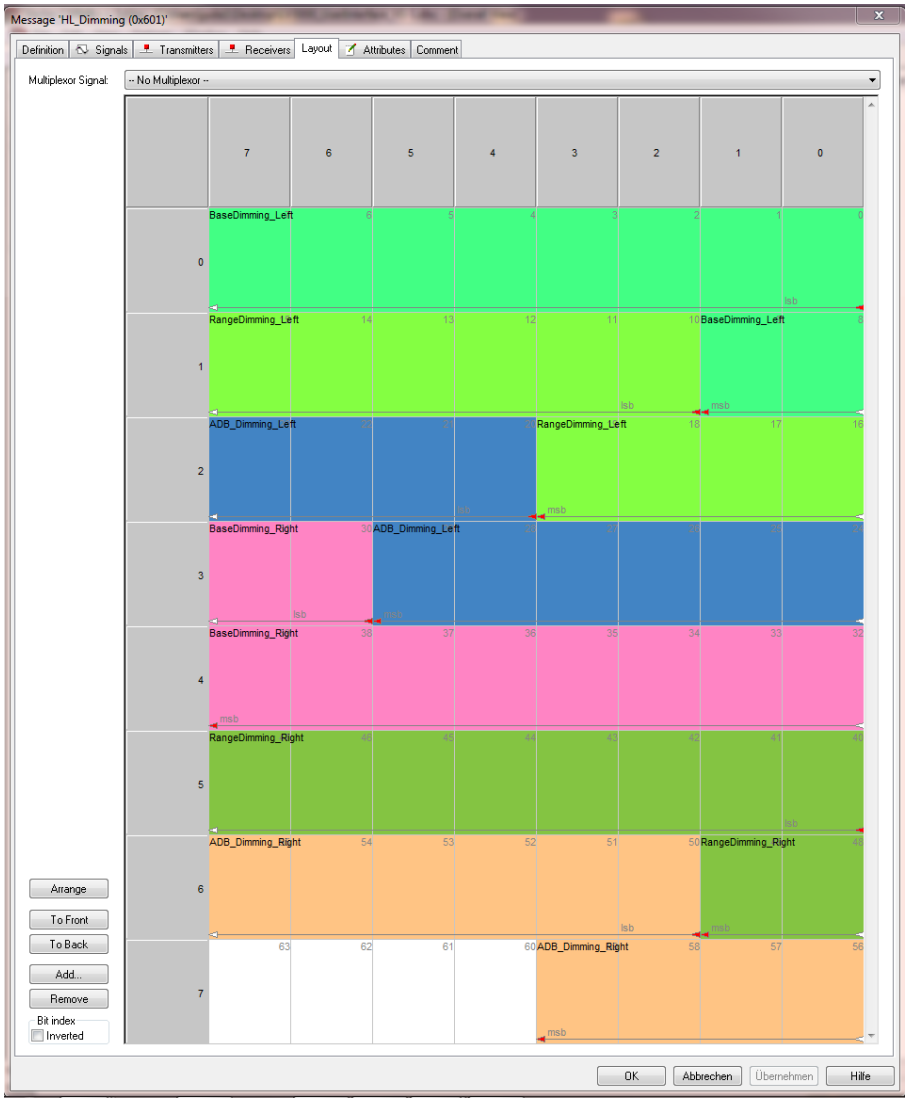

Figure 5-1 Distribution of signals in headlamps dimming CAN message

A 10 bits unsigned BIN number can represent a value from 0 to 1023. And CAN database can also defines how this data to be understood. A factor and offset is defined to translate the figure. In this case, the factor is 0.1 and the offset is 0.

The factor means the min step in the changing of the number. So the cars receive the data, they will understand this data in this way.

$$\text{Number} = (\text{data in Bin}) \times \text{factor} - \text{offset}$$

This means when the server got the data from the user, it needs to transfer it to a unsigned number in BIN by this way,

$$\text{Number} = (\text{data in Bin}) \times \text{factor} - \text{offset}$$

And then transfer this number to BIN. In JavaScript, the convert to BIN can be done by toString(2) function.

In CAN network, bit order is optional for the designer, the Motorola bit order and Intel bit order. Or to be more specifically, endianness.

The terms endian and endianness refer to how the data is stored in bits in the memory or hard disk in the computer. This indicates how the most significant bit or byte is stored. It is stored in a lower address or it is stored in the higher address.

In this project, the little endian as known as Intel bit order is chosen. This means the CAN network will understand the message in a little endian way. As the figure shows, in base dimming left signal, the highest bit is bit 9, the smaller bit will be stored in smaller address.

This raises a problem for a string, which the higher bit is stored in front of the BIN number. So a reverse is required when the covert is finished. The length of the signal is always fixed, but the converted result from the toString(2) function is not fixed. This means if transfer 1 to HEX, the result will be 1 instead of 0000001. But the signal always has a fixed length, so 0 must be added to the converted string to make it become the same length every time.

After that, all the strings of different data will be added together to form the final data field.

Then, a problem occurred, the result is not the same as expected when connected to CANoe software. The software cannot get the right result.

This is due to the length of signal is longer than one byte, which means the one signal may be divided into two bytes. Take base dimming left as example, the bit 8 and bit 9 in the next byte, but the CANoe analyses the byte in Intel bit order, so bit 14 and bit 15 are treated as the rest two bits for this signal. Extra process is required to deal with this problem.

After the result string is assembled, it should be divided into bytes (8 bits) and reverse by byte. Each 8 bits will be seen as a byte, and reversed.

Then, the BIN string needs to be transformed to HEX. The simple convert will not be effective here. The reason is the same, the fixed length of the HEX data. The simple convert will not transfer the BIN data to 0B form. However, in this circumstance, the data field needs to be a fixed length HEX data.

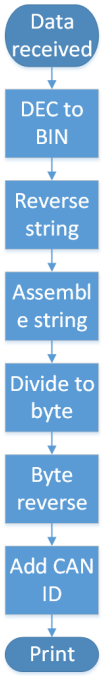

Figure 5-2 Process of compiling data to HEX CAN message form

After all these process, the data field of the CAN message is generated, the CAN ID needs to be added to the front of the string, the total process of action is described in figure 5-2. After that, the web server will print the whole string to the micro controller to make it send the CAN message to the car.

And all the serial printing commands are carried out by the setInterval function. It is a function can repeat the one action in a required time, and a variable contains a handle is return, so the action can be stopped any time by the clearInterval function.

In the control logic from the server, each change of the slider will clear the former timer and start a new one. This can ensure only the latest command is sent. And when the stop button is clicked, all the timers will be stopped and start to send the stop CAN message. The stop timer can be automatically stopped when a new order is given as well.

5.2.4 Receiving Data from Micro Controller

Despite the CAN network does not give any feedback to the system, the system is expected to get the temperature data of the complements of the car and present the data to the user.

In CAN network, the temperature data is sent by broadcasting, there is not query and answer process in this network.

The temperature data is sent by CAN message by ID of 0x600, as it shows in figure 5-3. In this message all data are in byte form. So there is not extra process required. The micro controller can get the number of the temperature when receives the CAN message.

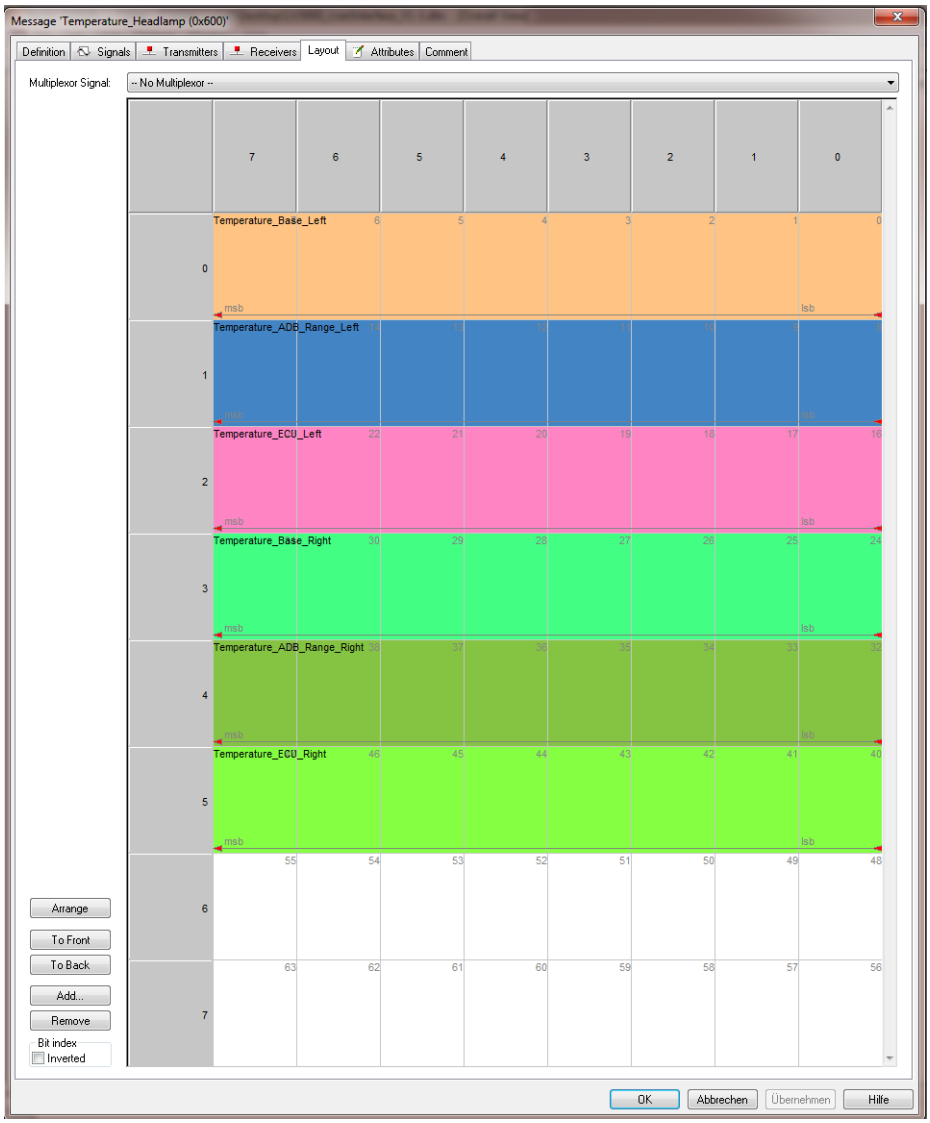

Figure 5-3 Distribution of signals in headlamps temperature CAN message

After getting the data, the micro controller needs to transfer the data to the web server. As it is described, the data transmission will be performed by serial port in the form of strings. The strings' format is fixed as some kind of protocol between the micro controller and the server.

The micro controller will assemble the string in the following format:

N:base_left,T:256.

When the server receives the strings, the server will search for ':'. And the string between 'N:' and the ',' will be treated as the name of the temperature data and in the same way, it can also get the value of it.

When the server receives the value, it will emit a socket event to push all these data to the connected users to refresh and update the gauges.

5.3 Micro Controller with CAN Bus Shield

5.3.1 SPI communication

The Serial Peripheral Interface (SPI) bus is a synchronous serial communication interface, the both Arduino and the CAN bus shield has the SPI interface.[6]

In this system, the SPI communication is used in the communication between Arduino and CAN bus shield. Thanks to its master-slave mode, multiple slave devices are supported through selection with individual slave select (SS) lines so it is possible to add multiple shields to one Arduino board when all the shields have different select pins.

Arduino supports the SPI communication and a simple attachment is available when using an Arduino Uno board with the shield. The Arduino Uno and CAN bus shield is connected by the following pins specialized in table 5-2.

Table 5-2 Pin mapping of Arduino Uno and CAN BUS Shield

CAN Shield	Arduino UNO
VIN	VIN
GND	GND
GND	GND
+ 5V	+ 5V
+ 3.3V	+ 3.3V
RST	RST
D10	D10
D11	D11
D12	D12
D13	D13

All other pins are not necessary for operation of the shield.

Chapter 6 System Testing

6.1 Simulation Testing

6.1.1 Testing environment

To test the function of the whole system, a CAN network is set up with two participants. One of the participants is the micro controller, the other is a CAN Case from Vector. It should be ensured that both parties send and receive.

The CAN Case is a tool to record and test CAN network. A CANoe software is required to make it function.

The type of the CAN Case is VN1640A.

To send and receive with CAN Case, an interactive generator is set in CANoe software. The CAN Case and the code to control the CAN bus shield are set both to 500kbps.

As it is described in the former chapter, in table 4-1. The pinning of the Sub-D connector of the CAN bus shield does not correspond to the usual standard of Vector. To use here-prefabricated cables, an adapter is necessary, the flips pinning accordingly. Usually the CAN High is (hereinafter referred to as CAN-H) to pin 7 of the Sub-D connector and CAN low (hereinafter referred to as CAN-L) to pin 2. On the Spark fun Shield is CAN-H to pin 3 and CAN-L on pin 5.

Structure of this testing environment is described in figure 6-1 that the ends of the CAN lines with a 120-ohm resistor between CAN-H and CAN-L must be completed. Absence of these resistors, there are reflections of the signals at the line ends; thereby communicating on line becomes impossible in the worst case.

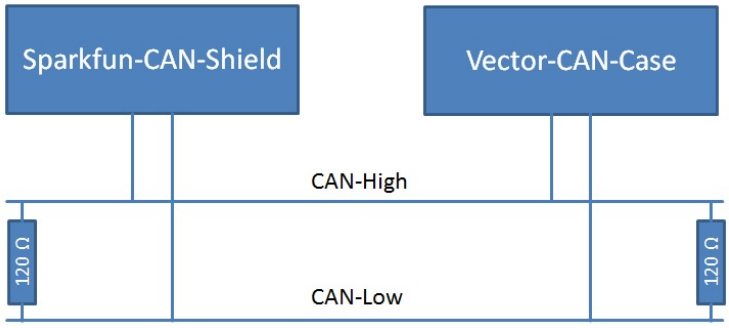

Figure 6-1 Set up of CAN network

After connection, the CAN messages should be able to send and receive from both sides.

6.1.2 CANoe Software

CANoe is a software tool for analysis of the CAN protocol traffic. With CAN Case, it can directly receive, send, log the can messages in the CAN network. It is also provided by Vector.[7]

The system is connected to channel one in the CAN Case.

An interactive generator is added to the software to produce Temperature data. The data field of each temperature data is configured as a sin wave.

The generator will send the CAN messages in a cycle time of 100 ms.

When the system starts sending CAN messages, all these messages can be viewed in the CANoe trace window. The database of how to analysis the CAN messages is also added to the program. The database is provided by HELLA and is the same with the real testing car.

6.1.3 Testing results

All the function are tested during the simulation testing to see whether the CAN messages can be sent and received properly. Including the analysis of the temperature data and the sending of every command, to see whether the system can send the right CAN messages contains the same parameters as the user expected.

1. Receiving temperature data from the CANoe software

In order to test the temperature monitor, the testing data is sent from the CANoe software.

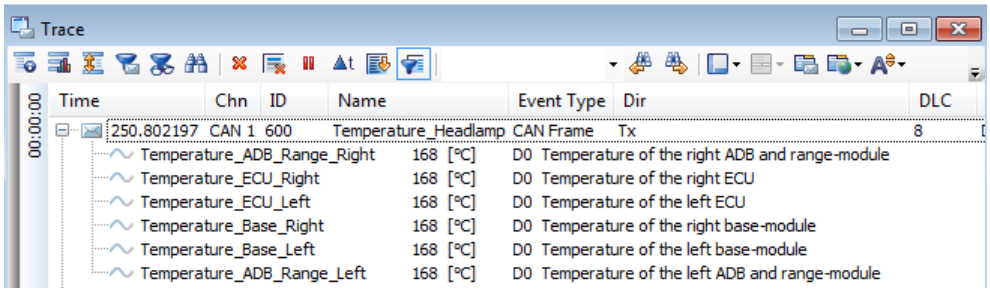

Figure 6-2(a) The CAN message contains temperature data sending from CANoe software

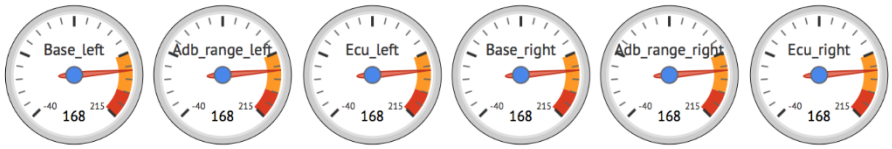

Figure 6-2(b) The gauges on the website

From figure 6-2, the same temperature data can be viewed from the CANoe, the sender, and the website provided by the system, the receiver.

2. Testing headlamps mode direct drive.

During this mode, in order to control the car, the CAN message by ID 0x603 should be sent to enable the headlamps to make it obey the orders from the system. And two other controlling signals 0x602 for headlamps actuators as well as 0x601 for headlamps dimming. The first column for the four sliders is used for headlamps actuators. After dragging them and setting data, the enable signal in direct drive mode is sent automatically. And the 0x603 message contains the actual data is sent in the same time as in figure

6-3.

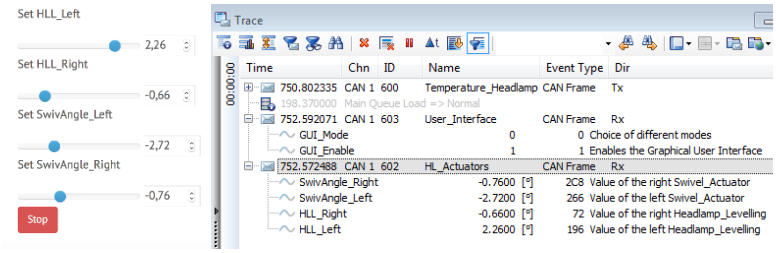

Figure 6-3 The data set in the website and the CAN messages sent by system

After changing the slider, in figure 6-4, the data in the CAN messages is also changed.

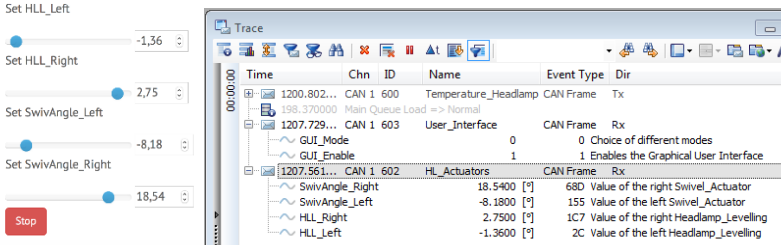

Figure 6-4 The data set in the website and the CAN messages sent by system after moving slider

Since the CAN message usually sent by cycle time, so if a message is not sent any more, the car will still respond to it until it reaches the maximum time, that is why a stop button is required, when press the stop button, the system will send a stop command to the car and in the same time clear all other commands.

In figure 6-5, the stop command, CAN message enable with enable bit set to 0 is sent, and the former CAN message turns into grey because it is no longer received.

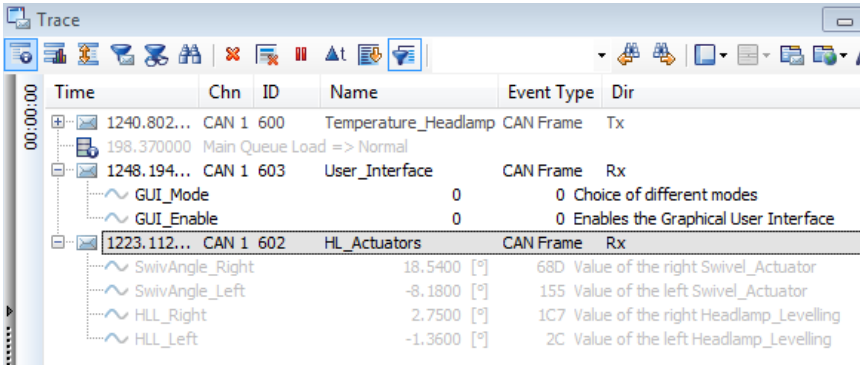

Figure 6-5 CAN messages sent from system after stop

3. COL and AFS mode

In COL and AFS mode, the 0x603 user interface message will tell the mode. And extra signal 0x604 headlamps lighting function is required in this mode to give extra information of the current working mode.

In COL mode, it has four different modes, town light, motorway light, country light and high beam. And the two sliders are not required for this mode.

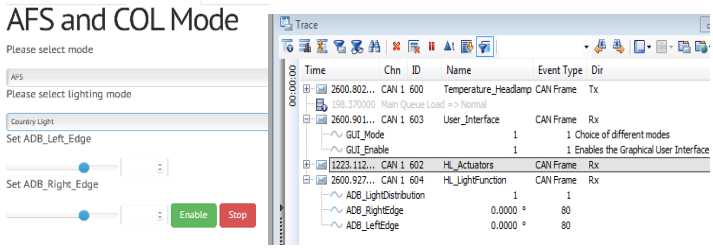

Figure 6-6 CAN messages sent form system after switch to AFS mode

In figure 6-6, we can see the GUI mode changed to 1, stands for AFS mode and the ADB light distribution changed to 1 represents country light. It is the same result as the user configured. All these signals will be automatically changed when a new selection of the select box is made.

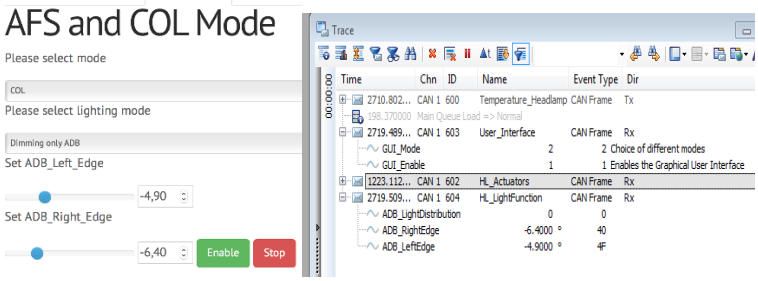

Figure 6-7 CAN message sent from system after switch to COL mode

When changed to COL mode, in figure 6-7, the two sliders will be enabled, and the data for ADB left and right edge is the same in the CAN message as the user expected.

4. Conclusion

After this simulation test, this system is proved to be effective, it can transfer the user command configured on the website into proper CAN messages and send them to the CAN network.

The data in every field of the website is the same with the corresponding field in the CAN messages. And the interactive logic works as expected; the enable command can be sent automatically when a slider or a selection is chosen. And they can be changed when the tab is changed.

The stimulation proved that the system functions as expected.

6.2 Real Car Testing

During the stimulation, the system seems to function normally. A few of real car testing was performed after that. The testing system is connected to a testing car by a standard HELLA CAN. The car is equipped HELLA tool chain for testing. During this test, all the functions of the system has been tested.

6.2.1 Testing Environment

Figure 6-8 is an overview of a HELLA tool chains in a testing car. The MicroAutoBox form dSpace functions as a control system in the testing environment. MicroAutoBox is a real-time system for performing fast function prototyping in fullpass and bypass scenarios. It operates without user intervention, just like an ECU.

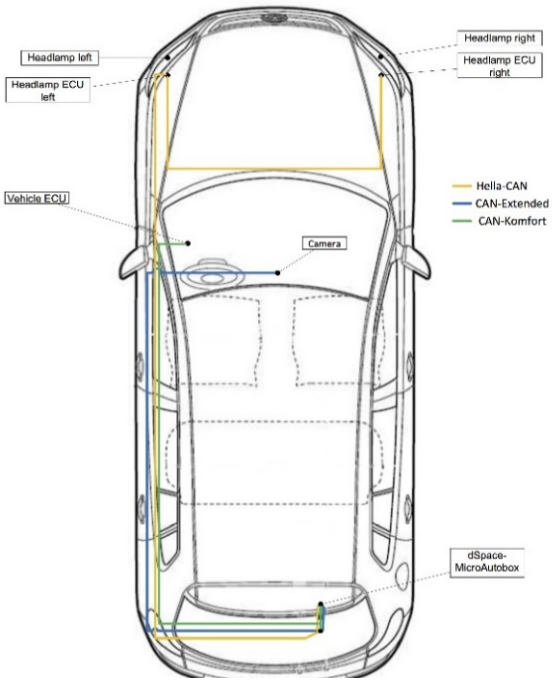

Figure 6-8 Overview of HELLA tool chains

The system is connected to the car from the CAN-H and CAN-L (HELLA CAN) provided by the MicroAutoBox. The MicroAutoBox is programmed to work in the mode that the engine is not working to avoid potential risks. However, the headlamps are functioning in this mode and the power supply is from the battery. The system itself is powered by a cigarette lighter on the car. An adaptor is used to change the 12V cigarette lighter to a normal 5V USB port, as it shows in figure 6-9.

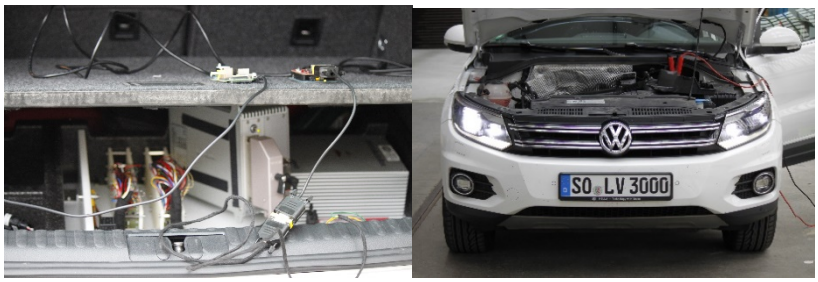

Figure 6-9 Set up of real car testing

As discussed, the Wi-Fi network is generated by the webserver itself, and for testing, an iPhone 6, an iPad 2 and a MacBook Pro as well as an Android phone are used to access this website.

The tests in total be performed 4 times, at the first time a computer is used to run CANoe software to make a record of all the CAN messages interactive between the car and the system. Just like in the

simulation test, in order to see the CAN message's contains and the data its carrying is true or false. After that, the following 3 tests, the computer is no longer necessary because all the correctness of the messages sent by the system have been verified.

6.2.2 Testing Results

1. Temperature monitor

The temperature data is successfully got from the car, as in figure 6-10. The ECU unit is 0 because of the ECU unit is not sending temperature. After checking the record of the signals, the original data is 0 from the CAN message. So the system can analyze the temperature data from real car, the result is normal.

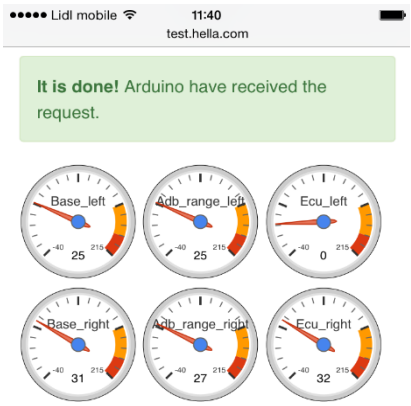

Figure 6-10 Temperature data from the car

2. Controlling Light Distribution

Due to the condition, an accurate measurement of headlamps lighting is not possible and necessary. Because in the former tests, the correctness of the parameters contained in the CAN message have been verified. The main task of the test is to see whether the car responds to all these commands.

Some slightly modifies of the parameters are invisible from picture, a few pictures will be presented to give a general idea of how the system functions.

From the website in figure 6-11, the changes of headlamps from town light, country way light and high beam when selecting different options on the website.

Figure 6-11 Lighting distribution of headlamps

6.3 Conclusion

From the testing results in the simulation tests, the system proves to be effective in transfering all the user commands to the corresponding CAN messages. And in the real car testing, the headlamps responds to the CAN messages correctly.

The system is proved to be effective and functional in all the designed functions.

Chapter 7 Conclusion

In this thesis, a system based on Raspberry Pi and Arduino is presented. The system is low cost, light and environment friendly considering to its power consume. And the software is easy to maintain.

The system is capable of directly connecting to the car and provides access to the user to allow them to control the headlamps in the car through a website. The UI is compatible to devices ranging from PC, smartphones and tables.

Both simulation tests and real car tests have been performed. The system has been proved to be functional and effective.

Acknowledgement

First and foremost, I appreciate my college here in the L-LAB and HELLA who gives me a comfortable research atmosphere and necessary help. Second, I would like to show my deepest gratitude to my supervisor, Mr. Duhme, who helps me walk through the whole project and take care of all the arrangement and hardware. Without his help and patience, this thesis could not have reached its present form. I also want to thank Mr. Hesse for giving all advises about the CAN protocol and help me prepare the real car tests.

I am also very grateful to Mrs. Krause in HELLA and Miss Qin in international affairs of Southeast University who contribute their work to this exchanging program and help me through the whole visa process and of course to CSC for its sponsorship.

I am greatly indebted to my adviser Mr. Li, Mrs. Zhao in Southeast University and Mr. Han, my friend, who help me with the whole graduation process when I am absent.

Apart from that, I want to pay my great thanks to Miss Wang and Mr. Chen, my classmates, as well as my best friend, Mr. Xia, my life couch, Mr. Chen for helping me through the emotional difficulties when I have troubles with the project.

Last but not least, I want to thank my parents for supporting me and to all my teachers who have helped me to develop the fundamental and essential academic competence.

参考文献（References）

[1] Michael Hamm. Safety Improvement generated by Pioneering New Matrix and Direction Indicator Functionalities[C]. In: Tran Quoc Khanh, eds. Proceedings of the 10th International Symposium on Automotive Lighting. Darmstadt: Herbert Utz Gmbh, 2013. 310-320

[2] DrivingVeisionNews, Adaptive Driving Beam Application to Matrix Beam [EB/OL]. http://www.drivingnisionnews.com, 2014-09-23

[3] Armin Austerschulte, Bernd Dreier, Ernst-olaf Rolaf Rosenhahn. Analysis of Safety Aspects for LED Matrix High Beam Functions[C]. In: Tran Quoc Khanh, eds. Proceedings of the 10th International Symposium on Automotive Lighting. Darmstadt: Herbert Utz Gmbh, 2013. 321-320

[4] Michael Kleinkes. New Automotive Lighting Technology: Benefit or Mayfly?[C]. In: Tran Quoc Khanh, eds. Proceedings of the 10th International Symposium on Automotive Lighting. Darmstadt: Herbert Utz Gmbh, 2013. 361-366

[5] O. Pfeiffer, A. Ayre, and C. Keydel. Embedded Networking with CAN and CANopen. Greenfields: Copperhill Technologies Corporation, 2008.203-240

[6]Microchip Technology Inc. Stand-Alone CAN Controller with SPI Interface [EB/OL]. http://ww1.microchip.com/downloads/en/DeviceDoc/21801e.pdf, 2015-03-26

[7] Vector Gmbh. CANoe Installation & Quick Start Guide [EB/OL]. http://cfile10.uf.tistory.com/attach/255B67355214061D0ECE00, 2015-03-15

[8] L. C. Passarini and M. Pinotti. Analyzing In-Vehicle Real-Time Communications: The Controler Area Network (CAN Protocol) [C], eds. SAE International. Warrendale, 1997. 973055

[9] Aktouf, O. Deleuze, C.Wahl, M. Dependability of embedded networks — A case study with system diagnosis of CAN protocol[C]. Eds. Intelligent Transport Systems Telecommunications (ITST). 2009. 552-556

[10] P. R. Burje, K. J. Karande, A. B. Jagadale. Embedded On-Board Diagnostics system using CAN protocol[C]. Eds. Communications and Signal Processing (ICCSP), 2014. 734–737

[11] Tan, Xiao-Peng, Li, Xiao-Bing, Xiao, Ti-Liang. Real-time analysis of dynamic priority of CAN bus protocol[C]. Eds. International Conference on Electronics and Information Engineering, Proceedings: v1, 2010. 1219-1222

[12]Giampiero Campa. Writing a Simulink Device Driver block: a step by step guide [EB/OL]. http://www.mathworks.com/matlabcentral/fileexchange/39354-device-drivers, 2015-02-15

[13] Satoshi Yamamura, Hidetada Tanaka, Noriko Sato, Takao Muramatsu. Glare-free High Beam with Beam-scanning [C]. In: Tran Quoc Khanh, eds. Proceedings of the 10th International Symposium on Automotive Lighting. Darmstadt: Herbert Utz Gmbh, 2013. 340-347

[14] 燕坤善,牛萍娟,付贤松. 汽车前照灯光源及发展趋势[J]. 光机电信息. 2008(11)

[15] Gould, R. Gordon (1959). The LASER, Light Amplification by Stimulated Emission of Radiation[C] In Franken, P.A. and Sands, R.H. (Eds.). The Ann Arbor Conference on Optical Pumping, the University of Michigan, 15 June through 18 June 1959. p. 128. OCLC 02460155.