Title
Development and Field Testing of Laser Photodiode Array-Based Vehicle Detection Systems

Authors
Cheng, Harry H.
Shaw, Ben
Palen, Joe
et al.

Publication Date
2004-10-01

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Development and Field Testing of Laser Photodiode Array-Based Vehicle Detection Systems

Harry H. Cheng, Ben Shaw, Joe Palen, Zhaoqing Wang, Ping Feng, Stephen Nestinger, Bo Chen

California PATH Research Report
UCB-ITS-PRR-2004-38

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

Final Report for Task Order 4310

October 2004

ISSN 1055-1425

CALIFORNIA PARTNERS FOR ADVANCED TRANSIT AND HIGHWAYS

Development and Field Testing of Laser Photodiode Array-Based Vehicle Detection Systems

Harry H. Cheng
Ben Shaw
Joe Palen
Zhaoqing Wang
Ping Feng
Stephen Nestinger
Bo Chen

Integration Engineering Laboratory
Department of Mechanical and Aeronautical Engineering
University of California, Davis
Davis, CA 95616

Contents

Abstract

Over the past year we have researched the development of a network-based real-time laser-based non-intrusive field-deployable detection for the delineation of moving vehicles. The primary goal of this project is to develop a roadway detection system that can be used to gather reliable travel time data non-intrusively. A powerful Rabbit 3200, instead of multi-microchips, is used to control digitally controlled potentiometers (DCP), which adjust the gain of the sensors' signals. Utilizing digitally controlled potentiometers allow for quick and easy adjustment on the highway with only the need of pushing a button. The adjustment, which used to take half or an hour, now only takes several seconds. The Rabbit 3200 is also used as a data sender to a computer through the Ethernet. The rabbit digital input ports are triggered to collect the signals by an interrupt pulse from a PWM signal, which also acts as the laser source trigger. The Rabbit 3200 will then pack all of the data into TCP packages and send them to a remote computer over a network. The Rabbit 3200 is also able to filter noise after having finished the task of adjustment. The software of the system has been modified for 8 channels. Compared with the previous 4 channel code, the new system is able to obtain more information on vehicles, including the profile of a passing vehicle. In order to improve the precision of the system, we improved the mechanical design, optical design and electric circuit design. We also use an interrupt to sample signals instead of a fixed sampling interval strategy. This method ensures every signal generated by the APD will be captured by the Rabbit 3200. The laser sources are pulsed at 10 kHz and the sampling rate of the Rabbit 3200 is synchronized to the 10 kHz laser pulse. This report describes the design and implementation of each functional component of the field-deployable system, the configuration of the field detection system, software design and implementation, and a signal calibration method to obtain higher system precision. It also demonstrates four different ways to test the field-deployable system and the results from each way of testing for future improvement.

1 Introduction

According to the conclusions of CalTrans and the US Department of Transportation, it is impossible to build our way out of traffic congestion. The solution is to run the transportation system more intelligently, which is known as the Intelligent Transportation System. Travel time is a good indicator of other direct constraints on ITS efficiency: cost, risk, and attentive workload. The importance of travel time is verified in Advanced Traffic Information System (ATIS) user surveys, which indicate that what travelers almost always most want from a transportation system is reduced travel time and higher reliability (e.g. reduced travel time variance and reduced risk)[1][2]. Every traveler must implicitly or explicitly make an assessment of these various travel time options before embarking on every trip; therefore, this information is definitely of high value. Because trip travel time is the parameter the public most wants to minimize, this is the parameter that is most important for transportation service providers to measure and minimize.

Speed is commonly used as an indicator of the travel time across a link. In current practice, speed is measured at one or more points along a link and extrapolated across the rest of the link [1]. This extrapolation method is used regardless of the mechanism of detection. Example detection methods are loops [13] which determine speed from two elements twenty feet apart; radar---which can directly determine speed from the carrier frequency shift (Doppler Effect); or video image processing [4], which tracks vehicles across the pixel elements within the field of view. The extrapolation from a point to a line is not necessarily valid. At the onset of flow breakdown, the speed variations along the length of a link can be quite large. Also, the onset of flow breakdown is when routing decisions are most time critical and accurate information has the highest value, so inaccurate extrapolations could have detrimental effects to the traveler.

An alternate method to determine the traverse travel time (e.g. the true link speed) is to use Vehicles As Probes (VAP). A VAP system determines travel time directly by identifying vehicles at the start of the link and re-identifying them at the end of the link, with the time difference being the true travel time. The problem with VAP systems is that they require large numbers of both vehicle tags and tag readers to be effective, and the cost justification of such a system seems unwarranted in the light of other options. The key aspect to measuring the actual travel time is simply to identify some distinguishing characteristic on a vehicle at the beginning of a link and then to re-identify that same characteristic on the same vehicle at the end on the link. This is the basic idea of VAP, however the characteristic does not have to be entirely unique (as in a vehicle tag), and it does not necessitate the infrastructure set-up costs of VAP. If a characteristic can be found to separate the fleet into (say) 100 classifications, “the maximum probability fit” can be determined for the same sequence of classifications at the downstream detector as was identified at the upstream detector. This is what is currently being done in Germany with the low-resolution imaging provided by (new high speed) loops [1]. If a higher-resolution detector is used so that it is possible to get a few thousand classes, then it should be quite possible to perform a 100% upstream-downstream Origin and Destination (O/D) analysis (even if a significant percentage of the vehicles switch lanes) using time gating and other relatively straight-forward signal processing techniques. The mechanism of detection must allow highly resolved delineations between commonly available “commuter” vehicles, because commuter vehicles represent the majority of the vehicle stream during the period that travel time information is most needed (e.g. the peak hours).

Any mechanism to measure travel time, by definition, is only determining the “past state” of the transportation system. Collecting data on what happened in the past has no utility except if it is used to infer what may happen in the future. All decisions, by definition, are based on an inference of future consequences. When a traveler learns that speed on a route is 50 MPH, the traveler generally infers that the speed will remain 50 MPH when she/he traverses it. This may or may not be an accurate inference. Travelers want to know the “state” of the system (in the future) when they traverse it. In the simplest case, this is just a straight extrapolation of current “state”. More sophisticated travelers may develop their own internal conceptual model of the typical build up and progression of congestion along routes with which they are familiar. A major benefit of ITS will be to provide travelers with a much more valid and comprehensive “look ahead” model of the (short-

term) future state of the transportation system. Validation of any traffic model requires (either implicitly or explicitly) traffic O/D data. The lack of valid O/D data has been the major impediment in the calibration, validation and usage of traffic models.

In this research project we are developing a roadway detection system that can directly determine O/D data non-intrusively without violating the public's privacy (as in license plate recognition systems). This detection system employs lasers to non-intrusively detect vehicles under highway conditions. This system has been under development for some time [19][20][21]. More recently we have improved the system further. A feed-back loop, using a Rabbit microprocessor for control, was introduced into the system. The mechanical system was improved to enhance the precision of the system. In addition, software to display information or communicate with the detector was updated. The new system device is shown in Figure 1-1.

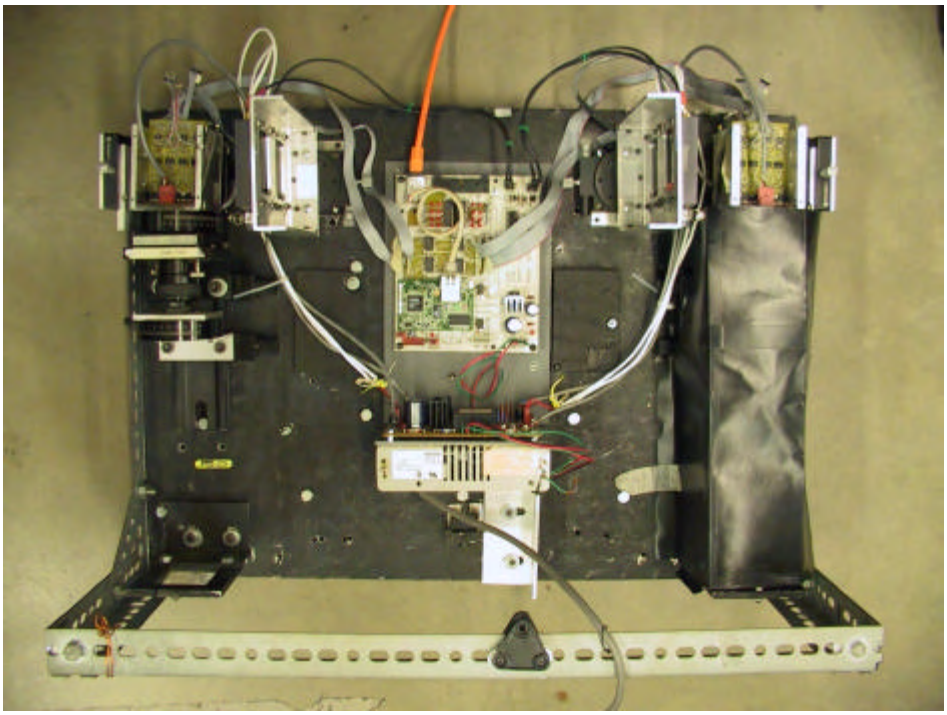


Figure 1-1. Current prototype of the laser-based detection system.

2 Overview of the Laser-Based Detection System (LBDS) Project

Figure 2-1 shows a conceptual drawing of the Laser Based Detection System (LBDS). It is comprised of two laser diode modules and other optical, mechanical, and electrical components. The basic detector unit consists of a laser diode module and a photodiode array positioned above the roadway. The laser diode module consists of a pulsed infrared diode and line-generating optics that projects a laser beam onto a road surface. The imaging lens focuses the reflected laser light onto the active area of the photodiode sensor array. The signal from a photodiode array is amplified and sent to a computer for processing. Vehicle presence is detected based on the absence of the reflected laser light. Two detector units are integrated and placed at a known distance apart. Two pairs of lasers/sensor optics are used for detecting vehicle presence at two points laterally about 4 and 8 feet across the lane. A sampling rate of about 10 kHz is is used in the system. As shown in Figure 2-1, t_1 is the time when the first laser line is blocked, t_2 is when the second laser line is blocked, t_1' is when the first laser is unblocked, and t_2' is when the second laser line is unblocked. We can calculate the length and the speed of a vehicle with D_{ref}, which is the distance between the two laser lines, t_1, t_2, t_1', and t_2'. Details on the calculations can be found in Ref. [30].

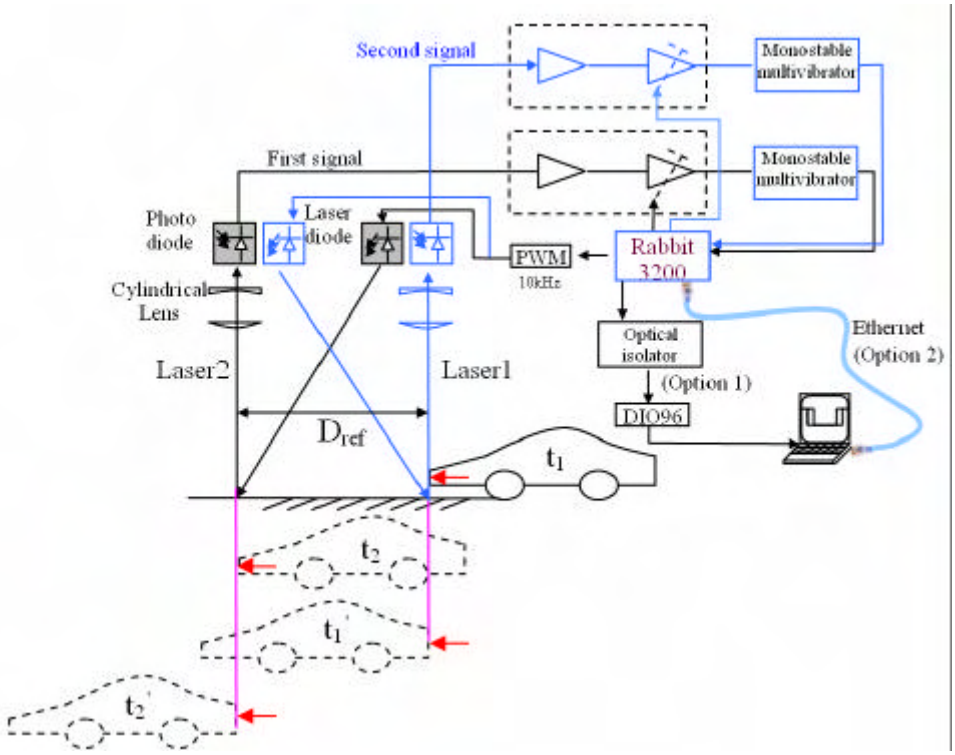
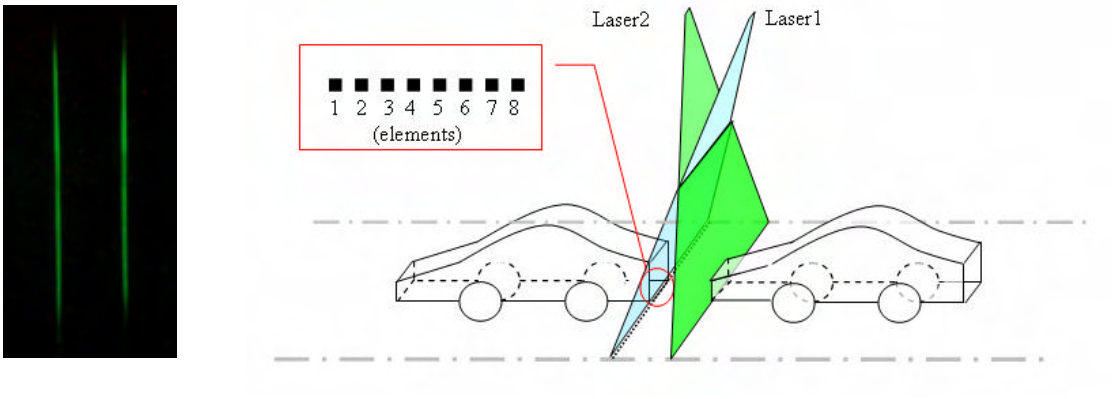


Figure 2-1. An overview of the laser-based detection system.

The two linear laser beams that are used as laser sources are shown in Figure 2-2 (a), which were imaged indoors through the use of a laser viewer. Figure 2-2 (b) shows the two laser beams projected across a lane of a highway when a vehicle is at the edge entering the field of view of the first laser/senor pair. The laser beam is reflected from the ground and is focused onto a 7.5 mm linear APD array through an optical system. The APD is an array of 24 photodiode elements, of which only eight are used in the current version of the LBDS prototype. There are eight signals for Laser1 and another eight signals for Laser2. Each pair of element signals of Laser1 and Laser2 will turn out a set of vehicle parameters, giving a total of eight sets of vehicle parameters. The purpose of acquiring

multiple vehicle parameters is to improve the reliability of the parameters and to acquire the outline profile of a vehicle.



(a) **(b)**

Figure 2-2. Laser-based detection system overview.

When the Rabbit3200 reads the signal data, it packs data into TCP Packages and sends them to a remote computer through an Ethernet port. The vehicle parameters are calculated and displayed in a remote computer.

2.1 Architecture of the LBDS

The architecture of the LBDS is shown in Figure 2-3. It is composed of an opto-mechanical subsystem for laser-beam collection and adjustment, signal amplification and shaping circuitry to pick up the laser signals for conversion to digital signals, a Rabbit 3200 for data acquisition and data communication broker, and a computer for parameter calculation and record/replay/reuse of data

The opto-mechanical laser receiver collects the laser signal reflected from the ground into the APD linear array. The APD linear array generates signals when the laser beam is present. The signals are then amplified by the signal amplification and shaping circuitry. The opto-mechanical laser receiver has a very narrow field of view. When the system is moved to different heights, the laser sources must be adjusted so that the projected linear laser beam stays within the systems field of view. The opto-mechanical system is very sensitive to the signal output from the APD that is mounted to it. Our system should be adjusted accordingly relative to the height of the detector from the ground.

The signal amplification and shaping circuitry is used to pick up the signals generated from the APD array, and then amplify and shape them. It outputs digital signals that are read by the Rabbit 3200 core module. Considering that 48 signals will be read in the future instead of the 16 signals that are currently read, we use bus technology to connect the Rabbit core module to the signal amplification and shaping circuitry.

The Rabbit acts as a PWM generator to trigger the ML20A15 laser diode modules at 10 kHz. The ML20A15 laser diode module produces a linear laser beam with a full fan angle of 15 degrees that is projected towards the ground. Some of the reflected laser beam from the ground will fall onto the front of the receiving lens of the opt-mechanical system, which then falls onto the APD array causing it to output a pulse signal. The Rabbit core is triggered at the same frequency as the laser diodes to read the digital signals coming from the signal amplification and shaping circuitry. Every 10ms, the Rabbit module sends one TCP package containing 100 sets of data to a computer over a network.

The computer calculates the parameters of vehicles as they pass underneath the detector, and then displays, records, or replays the results.

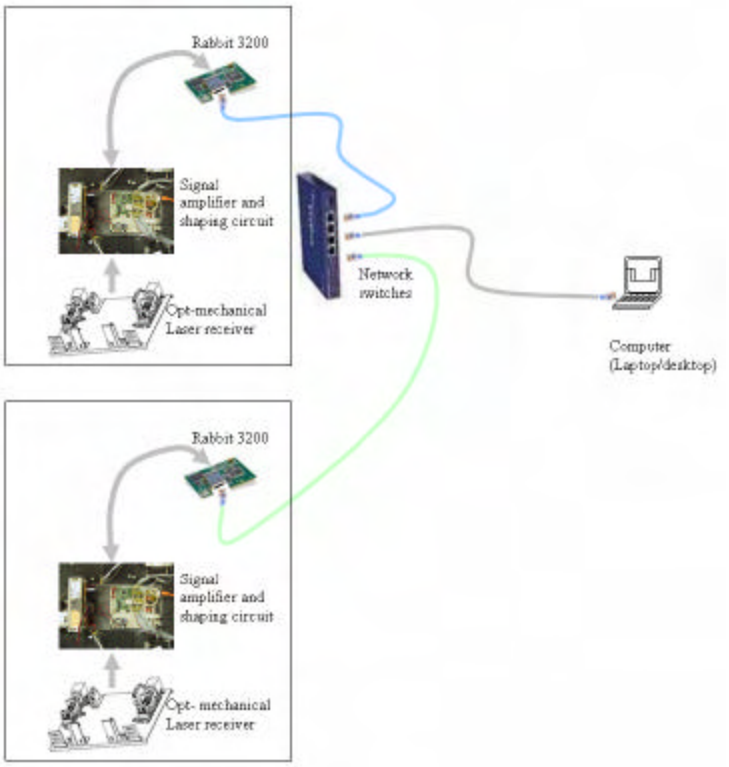


Figure 2-3. The Architecture of the LBDS.

2.2 Opto-Mechanical System

In this section, we will discuss the current optical design used in the LBDS, a proposed optical design for future work, and the development of new laser mounts to increase the alignment precision of the laser sources.

2.2.1 Optical Design of the LBDS

One of the crucial subsystems of the Laser-Based Detection System is the optical system that images the linear laser beam (projected onto the road) onto the optical sensor that detects the presence of a vehicle and which is used to calculate a vehicle's parameters and acquire the vehicle's profile. Any minor deflection of the optical system can cause problems in attaining a suitable signal for overall system operation. The optical system also determines the coverage area that is viewable, the amount of area the image is projected onto, and in accord with the optical sensor unit, determines the resolution of the overall system. The optical requirements of the LBDS are more rigorous compared with standard optical projects. The current optical system produces a useable image but does not satisfy the overall requirements of the LBDS. A new optical system has been proposed and will soon be implemented for testing.

2.2.1.1 Optical Requirements of the Laser-Based Detection System

The first thing to take into consideration when designing an optical system is to know the desired performance characteristics of the system. The optical requirements of the LBDS are more arduous than standard optical systems. The LBDS optical system, at a distance of 8m away from the road surface, must take in the reflected light of a linear laser beam that is 4m high and 5mm wide and converge it into a real image of the linear laser beam that is 7.5 mm high and .3 mm wide due to the active area constraints of the optical sensor, a 25 element avalanche photodiode array. Most standard optical systems are symmetrical in which the required power or magnification as well as the field of view of the system is the same along both perpendicular axes. The LBDS requires an anamorphic optical system in which the magnification and the field of view along the meridian or tangential axis are higher as compared the magnification required along the sagittal axis. The hardest of all requirements is that the optical system should continuously produce a consistent image when the LBDS is placed at different heights given that the distance from the objective lens to the road surface should be able to change from 5.4m to 10m. As the distance from the road surface changes, the height and width of linear laser beam changes due to the laser diodes used.

Magnification

The magnification of an optical system is given by

$$m = \frac{h_i}{h_o} \qquad \textbf{(2-1)}$$

Where h_i is the image height and h_o is the object height [24]. Since the APD array has a specific height of 7.5mm and width of .3mm, the image produced by the optical system should be of the same height and width in order to utilize the full 25 elements. Standard linear laser beams are produced by using a laser diode and a miniature optical system that sits in front of the laser diode. The linear laser beam would then fan out of the laser diode in which the height and width of the linear laser beam would change as a function of distance from the laser diode. Assuming that the LBDS utilizes a laser diode that can produce a 4m high, 5mm wide linear laser beam, the tangential fan angle of that laser diode would then be

$$FA_t = \arctan(4m/8m) = 26.6^\circ$$

and the sagittal fan angle of the laser diode would then be

$$FA_s = \arctan(5mm/8m) = .036^\circ.$$

The linear laser beam height as a function of distance from the road surface is given by

$$h_o = x \cdot \tan(26.6^\circ) \qquad \textbf{(2-2)}$$

and the width is given by

$$w_o = x \cdot \tan(.036^\circ) \qquad \textbf{(2-3)}$$

where x is the distance from the linear laser beam to the LBDS. Since the object height and width change with distance and the image height and width must remain constant, the magnification must

also change with distance. Using Equation (2-1) and Equation (2-2), we find that the wanted tangential magnification is

$$m_t = \frac{h_i}{h_o} = \frac{7.5mm}{x \cdot \tan(26.6^\circ)}. \qquad \textbf{(2-4)}$$

Using Equation (2-1) and Equation (2-3), we find that the wanted sagittal magnification is

$$m_s = \frac{w_i}{w_o} = \frac{.3mm}{x \cdot \tan(.036^\circ)}. \qquad \textbf{(2-5)}$$

The sign of the image height and width are not taken into consideration since the optical system should be axi-symmetric such that it should not matter whether the image is in the positive or negative direction. Table 2-1 gives the desired magnifications at different object distances.

Table 2-1

Object Distance (m)	Magnification	
	Tangential	Sagittal
5	0.0030	0.0955
6	0.0025	0.0796
7	0.0021	0.0682
8	0.0019	0.0597
9	0.0017	0.0531
10	0.0015	0.0477

Field of View

It is desired to have an optical system with a large field of view in the tangential axis in order to cover the 4m requirement while having a narrow field of view in the sagittal axis. The narrow field of view in the sagittal axis is due to the critical height requirement of the LBDS. The calculation of the field of view is given by calculating the slope of the principal ray, which just skims through the field stop [25]. The field of view of an optical system is calculated and taken into consideration during the design processes. Figure 2-4 shows the different fields of view that must be taken into consideration.

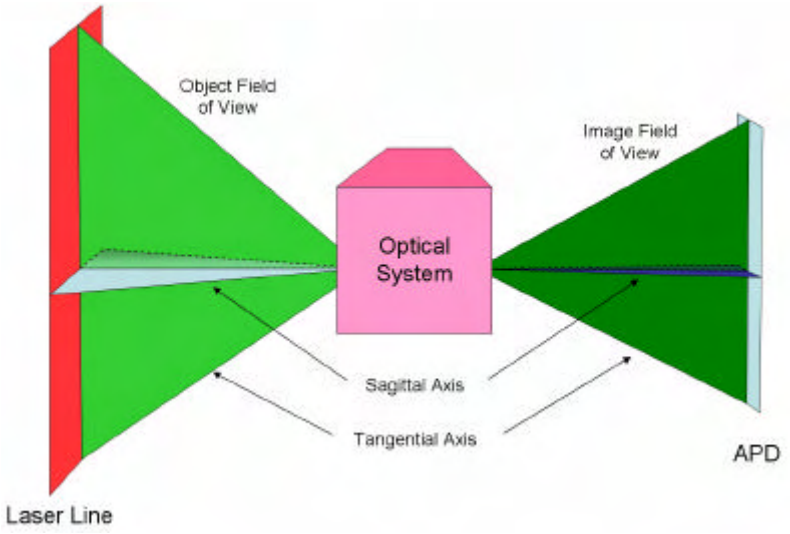


Figure 2-4. Fields of view of the optical system.

Size Restriction

The location of the image is not a real concern to the LBDS although a small f/# would be more appropriate since it would reduce the size of the optical system and help reduce the overall size of the LBDS. There are not mechanical restrictions to the optical system although size is of concern. The size of the LBDS is not only constrained to the size of the optical system but also to other subsystems such as the lasers when dealing with laser safety.

Resolution

The optical system along with the optical sensor determines the overall resolution of the LBDS. Since the optical sensor is a 25-element photodiode avalanche array, this limits the resolution to 25 elements over 4m. The optical system determines which light rays from the linear laser beam hit each individual element. The optical system should take every point from the linear laser beam and converge it to a point on a single element depending on which portion of the linear laser beam the emitting point is in. This in turn would give each element a specific detectable area as shown in Figure 2-5.

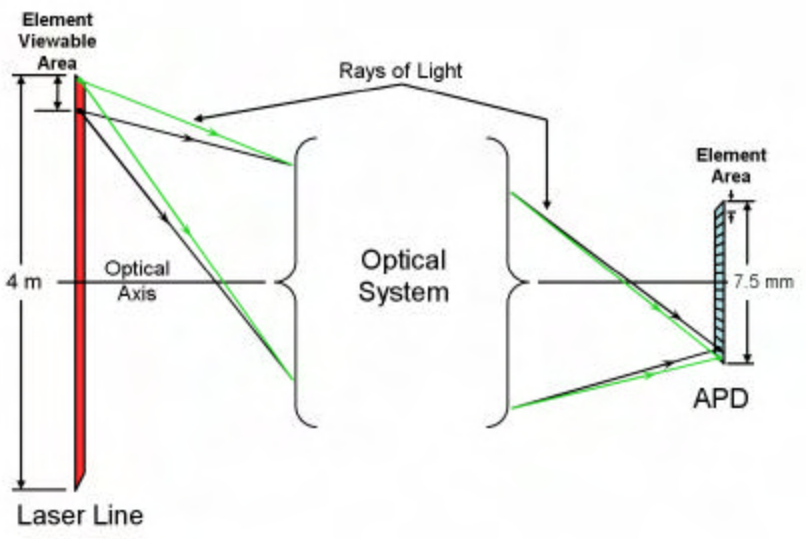


Figure 2-5. Resolution of the optical system.

In Figure 2-5, the green rays of light are those rays, which are emitted from the top of linear laser beam and converge at the bottom of the last element. The figure assumes that the image will be inverted which may not be the case. The black rays of light are those rays, which are emitted from a position of the linear laser beam and converge at the top of the last element. The distance between the positions of the green and black rays of light emitted from the laser is the element viewable area. This is the area that the last element of the APD array can be used to detect the presence of a vehicle and can only be acquired through a one-to-one convergence in which the light emitted from a point of the object converges into a point on the image. Some sensor systems employ a one-to-all method in which light emitted from a point of the object is spread throughout all of the area of the APD array. Having a higher resolution allows the LBDS to acquire a more accurate profile of a passing vehicle. With the APD array having 25 elements, the desired element viewable height is $4m/25 = 160mm$ and width is $5mm$.

2.2.1.2 Current Optics

The current optical system utilizes a simple negative Galilean system and a Tamron imaging lens. Figure 2-6 shows the position of the Galilean system and the Tamron imaging lens with respect to the APD array and the light emitted from the road surface.

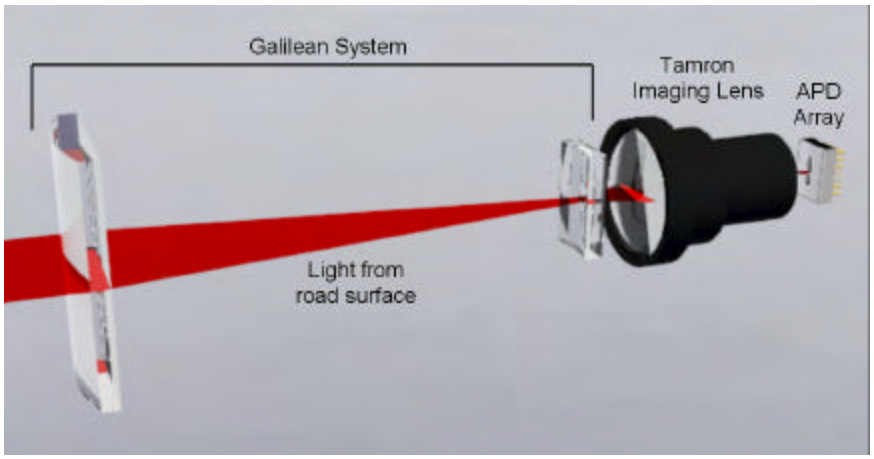


Figure 2-6. Current optical system.

The Tamron Lens

The Tamron lens is a Closed Circuit Television (CCTV) and Factory Automation Lens designed for a 2/3" CCD. This defines the image output area to be a circle with a diameter of 2/3". The Tamron lens is a varifocal lens with a focal length of 8 to 16mm. It contains an iris with a range of F/1.6 to close. With a 2/3 inch (16.933 mm) diameter image output, the Tamron lens has a field of view of 69 degrees and 1 minute in wide mode and 37 degrees and 56 minutes in telephoto mode. The diameter of the circular viewable object area is given by

$$D = x \cdot \tan(69^\circ 1'') \text{ for wide mode and}$$
$$D = x \cdot \tan(37^\circ 56'') \text{ for telephoto mode}$$

where x is the distance from the front of the Tamron lens to the object. Table 2-2 gives the calculated viewable diameter for different object distances.

Table 2-2

Object Distance (m)	Wide Mode	Telephoto Mode
	Viewable Diameter (m)	
5	13.025	3.906
6	15.631	4.688
7	18.236	5.469
8	20.841	6.250
9	23.446	7.032
10	26.051	7.813

Figure 2-7 **Portrays the information from** Table 2-2 **in a visual manner**.

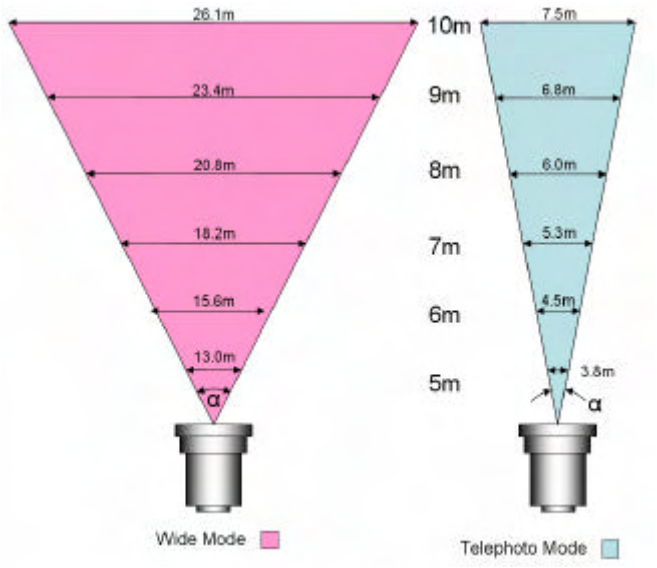


Figure 2-7. Tamron viewable diameter.

Since the image area is always constant, the image height of a given object in the object field is a factor of the object height and the viewable diameter at the distance of the object and the diameter of the image area.

$$h_i = \frac{h_o}{x \cdot \tan(a)} \cdot 16.933 \qquad \textbf{(2-6)}$$

The Galilean System

The Galilean system reduces the field of view of both the sagittal axis and the tangential axis. The factor of reduction provided by the Galilean system along the sagittal axis is approximately equal to the ratio of the focal lengths of the lenses that comprise the Galilean system. The first lens, the objective lens, is a 60mm high, 50mm wide cylindrical plano lens with a focal length of 250 mm. The second lens is a 25mm high, 12.7mm wide cylindrical plano lens with a focal length of -12.7 mm. Using these two lenses, the Galilean system provides a reduction factor of

$$RF = \frac{f1}{f2} = \frac{250}{12.7} = 19.685.$$

The main purpose of reducing the sagittal field of view of the optical system is to decrease the critical height at which the system can operate. The imposition of the critical height is due to the basic design of the LBDS described in [20]. The reduction of the tangential field of view is due to the design of the optical system. Figure 2-8 shows the tangential field of view reduction of an optical system as shown in Figure 2-6.

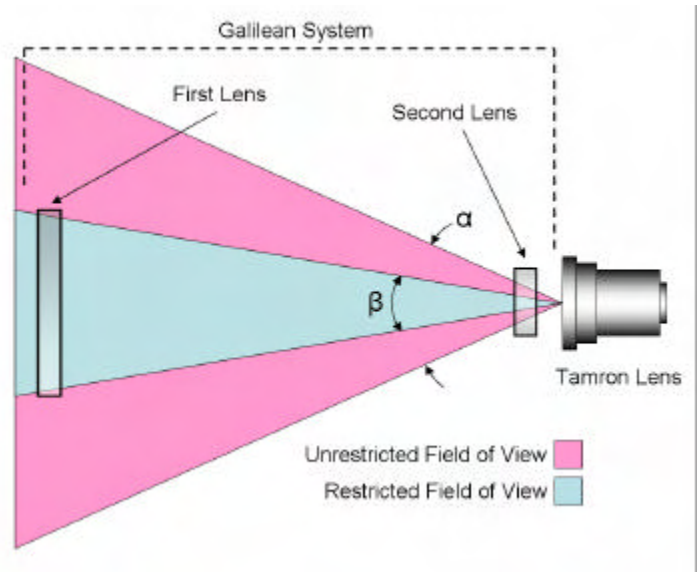


Figure 2-8. Reduced tangential field of view.

Figure 2-8 shows that the first lens of the Galilean system reduces the tangential field of view from a to b by reducing the size of the entrance pupil. The unrestricted field of view of the Tamron lens is 69 degrees in wide mode and 38 degrees in telephoto mode. The reduced field of view can be calculated from

$$b = \arctan\left(\frac{y_1}{x_1}\right) = \arctan\left(\frac{60mm}{250mm}\right) = 13.5^\circ$$

where y_1 is the height of the first lens and x_1 is the distance between the first lens and the Tamron lens. The modified tangential viewable object height is given by

$$D = x \cdot \tan(13.5^\circ) . \qquad \textbf{(2-7)}$$

Table 2-3 shows the modified viewable height as a function of distance from the LBDS.

Table 2-3

Object Distance (m)	Viewable Diameter (m)
5	1.200
6	1.440
7	1.681
8	1.921
9	2.161
10	2.401

Table 2-3 shows the current optical system is not capable of imaging a laser line that is 4m high at a distance of 8m and fails the first optical requirement. At most, the system can image a laser line that is 1.921m high at 8m. The restricted view can however be corrected by increasing the height of the first lens. Given that the Tamron lens has an unrestricted tangential field of view of 69 degrees, the

first lens would have to be $y_1 = 250mm \cdot \tan(69^\circ 1'') = 651.27mm$ high in order to encompass the full field of view of the Tamron lens in wide mode. However, this would give us a viewable height of 20.841m at 8m, which is unnecessary. Since only 4m is required, a first lens height of $y1 = 250mm \cdot 4m / 8m = 125mm$ would work giving a field of view of 26.6 degrees. The rest of this report will assume that the first lens of the Galilean system is large enough to view a 4m laser line at a distance of 8m in order to concentrate on other performance criteria.

Performance

Since the Galilean system only affects the magnification along the sagittal axis, the magnification along the tangential axis is strictly based on the Tamron imaging lens and only depends on the object distance from the Tamron lens. Although the Galilean system affects the tangential field of view of the Tamron system, the tangential magnification of the Tamron system would be the same as if the field of view restriction were not there. The tangential magnification is calculated with

$$m_t = \frac{h_{t,i}}{h_{t,o}} = \frac{16.933}{x \cdot \tan(a)} \quad \textbf{(2-8)}$$

The magnification along the sagittal axis is modified by the reduction factor introduced by the Galilean system and is calculated with

$$m_s = \frac{h_{s,i}}{h_{s,o}} = RF \cdot \frac{16.933}{x \cdot \tan(a)} = \frac{333.326}{x \cdot \tan(a)} \quad \textbf{(2-9)}$$

Table 2-4 gives the sagittal and tangential magnification at different object distances along with the desired magnifications at those distances.

Table 2-4

Object Distance (m)	Wide Mode		Telephoto Mode		Desired Magnification	
	Sagittal Mag.	Tan. Mag.	Sagittal Mag.	Tan. Mag.	Sagittal	Tangential
5	0.0256	0.0013	0.0853	0.0043	0.0955	0.0030
6	0.0213	0.0011	0.0711	0.0036	0.0796	0.0025
7	0.0183	0.0009	0.0609	0.0031	0.0682	0.0021
8	0.0160	0.0008	0.0533	0.0027	0.0597	0.0019
9	0.0142	0.0007	0.0474	0.0024	0.0531	0.0017
10	0.0128	0.0007	0.0427	0.0022	0.0477	0.0015

As can be seen from Table 2-4, in order to achieve the desired sagittal magnification, the Tamron system would have to zoom in more than is possible. The desired tangential magnification is achievable by zooming out a small amount from telephoto mode. This shows that the current optical system is not capable of completely satisfying the desired requirements. However, the narrowness of the image width is not a real concern since the APD array will still be able to detect a signal. The calculations of Table 2-4 are based on the ideal laser diode being present in the LBDS. This, however, is not so. The linear laser beam is produced by utilizing two overlapping laser diodes each with a full fan angle of 15 degrees in the tangential axis and .036 degrees in the sagittal axis. Each laser separately produces a Gaussian distributed linear laser beam with a height of 2.144m and width of 5mm at 8m. Since the linear laser beams are of a Gaussian distribution, the ends of the line contain less radiance power compared with the centers and therefore are not detectable. Two laser lines overlap approximately 2/3 of the laser length producing a single linear laser beam with a height of 2.858m which is less than the desired height and a width of 5mm. The actual linear laser beam height as a function of distance from the road surface is given by

$$h_o = 4/3 \cdot x \cdot \tan(15^\circ) \quad \textbf{(2-10)}$$

and the width is given by Equation (2-3) since the desired laser line width and actual laser line width are the same. Using Equation (2-1) and Equation (2-9), we find that the desired tangential magnification for the 15 degree fan laser diode is

$$m_t = \frac{h_i}{h_o} = \frac{7.5mm}{4/3 \cdot x \cdot \tan(15^\circ)} \quad \textbf{(2-11)}$$

and the sagittal magnification is given by Equation (2-5). The magnification calculated in Equation (2-11) is the magnification that would create an image height of 7.5mm with the current laser line setup. Table 2-5 gives the actual magnifications of the current optical system at different object distances with the desired magnifications that would allow the current laser setup to create the desired image.

Table 2-5

Object Distance (m)	Wide Mode		Telephoto Mode		Desired Magnification for Current Laser Setup	
	Sagittal Mag.	Tan. Mag.	Sagittal Mag.	Tan. Mag.	Sagittal	Tangential
5	0.0256	0.0013	0.0853	0.0043	0.0955	0.0042
6	0.0213	0.0011	0.0711	0.0036	0.0796	0.0035
7	0.0183	0.0009	0.0609	0.0031	0.0682	0.0030
8	0.0160	0.0008	0.0533	0.0027	0.0597	0.0026
9	0.0142	0.0007	0.0474	0.0024	0.0531	0.0023
10	0.0128	0.0007	0.0427	0.0022	0.0477	0.0021

Table 2-5 shows that the tangential magnification of the current optical system works for the current laser setup when the Tamron lens is placed in telephoto mode. The sagittal magnification will not work since it requires a magnification that the Tamron lens is unable to provide.

2.2.1.3 Proposed Optics

Some work has been done using the optics program OSLO to create a new optical system that will meet all of the given optical requirements of the LBDS at a distance of 8m from the road surface. To keep the same Galilean system configuration, the Tamron imaging lens is removed from the optical system and a new lens will be designed in its place. The first lens of the Galilean system will also be changed from a cylindrical plano lens to a spherical plano lens giving the Galilean system power in both the tangential and sagittal axis. From the given requirement parameters, we perform calculations to obtain ideal design parameters of the new lens. The given requirements are

$$s_i = 27mm,$$
$$s_o = 8000mm,$$
$$h_i = 7.5mm,$$
$$h_o = 4000mm,$$

where s_i is the image distance, and s_o is the object distance. Using

$$\frac{1}{f} = \frac{1}{s_o} + \frac{1}{s_i}, \quad \textbf{(2-12)}$$

where f is the focal length of the Galilean system, the image distance produced by the Galilean system can be found by rearranging Equation (2-12)

$$s_{i,1} = \frac{s_o f}{s_o - f} = \frac{8000 \cdot 96.75}{8000 - 96.75} = 97.93mm.$$

Since the third lens will be placed 35mm behind the second lens of the Galilean system, the object distance of the third lens is the distance between the image of the Galilean system and the new lens and is $s_{o,2} = 150 + 35 - s_{i,1} = 150 + 35 - 97.93 = 87.07mm$. Using the new object distance and the desired image distance, we find the focal length of the third lens by rearranging Equation (2-12) as

$$f = \frac{s_{o,2} \cdot s_i}{s_{o,2} + s_i} = \frac{87.07 \cdot 27}{87.07 + 27} = 20.609mm.$$

From the calculations performed above, we then choose lenses that match the parameters of the lenses that we need. Figure 2-9 shows the new proposed optical system.

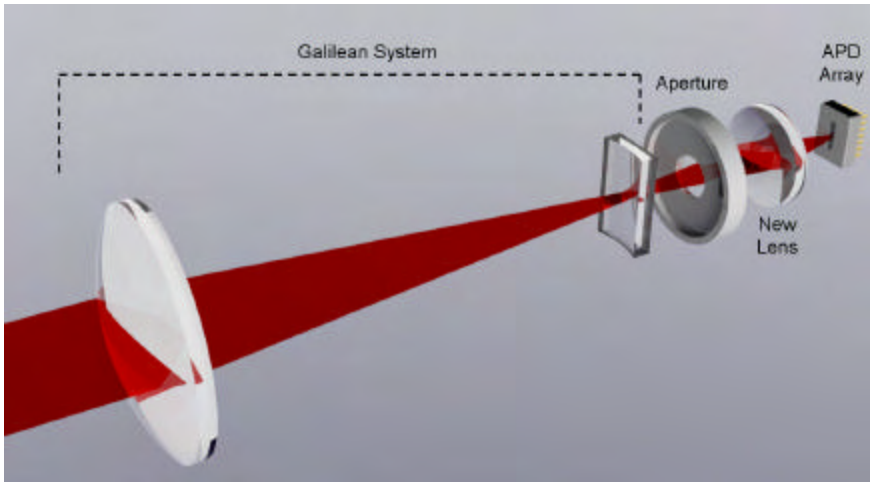


Figure 2-9. Proposed optical system.

OSLO Simulation

The lenses chosen have parameters displayed below in Figure 2-10. Lenses 2 and 3 (surface 3, 4 for lens 2, surface 5, 6 for lens 3) are catalog lenses that can be ordered from some optical companies. Although the object height in the table is displayed as 1 mm, it is actually 2000 mm (half of the actual length) as determined in another window not shown here.

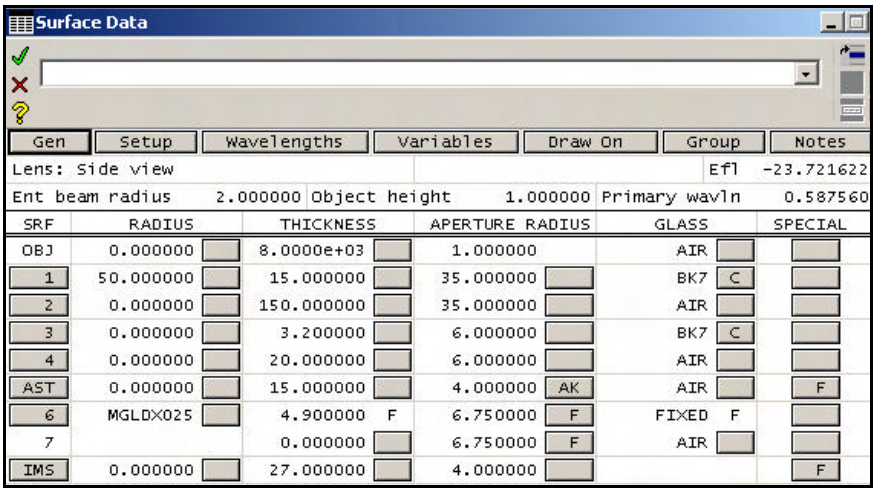


Figure 2-10. Optical system data of the tangential (side) view.

From the above Figure 2-10, we can see that the image size of the system actually meets the requirement of at least 24 elements of the APD array; we obtain an image size of about 8 mm. Having covered the required 24 elements of the APD array, the detector can have a sufficient amount of intensity to perform detection efficiently; thus, this newly designed optical system allows the LBDS to be more effective.

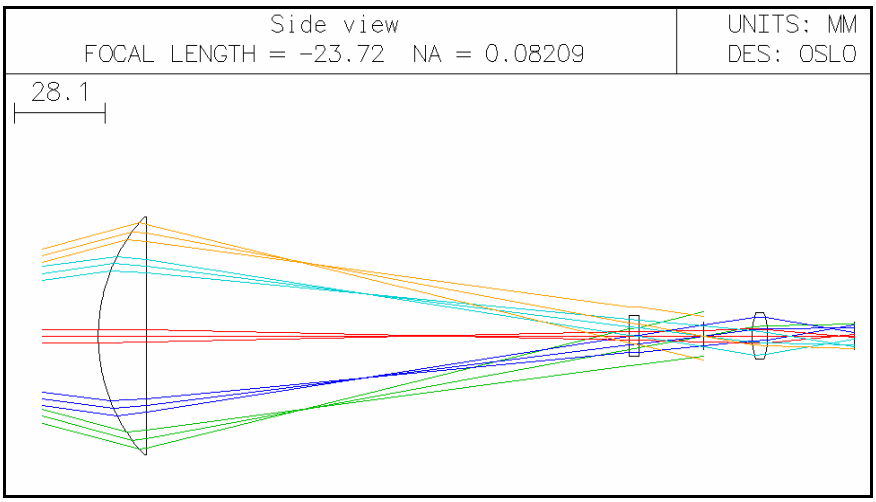


Figure 2-11. Sagittal view of the lens design layout.

From Figure 2-11 above, we can see that the light beams, which bounced off from an opposing wall, focused onto the image plane and forms an image of about 8 mm (notice the image plane is about 8 mm). The aperture stop is placed at the most focused spot of the system, minimizing the spherical aberrations that are caused by light beams that go through parts of the lens that have different curvatures.

Lens: Top view — Efl -11.099521

Ent beam radius 2.000000 | Object height 1.0000e-06 | Primary wavln 0.587560

SRF	RADIUS	THICKNESS	APERTURE RADIUS	GLASS	SPECIAL
OBJ	0.000000	8.0000e+03	1.0000e-06	AIR	
1	50.000000	15.000000	35.000000	BK7 C	
2	0.000000	150.000000	35.000000	AIR	
3	MGLCN001	3.200000 F	6.000000 FX	FIXED F	A
4		20.000000	6.000000 FX	AIR	A
AST	0.000000	15.000000	4.000000 A	AIR	F
6	MGLDX025	4.900000 F	6.750000 F	FIXED F	
7		0.000000	6.750000 F	AIR	
IMS	0.000000	27.000000	0.150000		F

Figure 2-12. Optical system data of the sagittal view.

Similar to the Tangential View, the parameters of the Sagittal View displayed above fulfills the given requirements of the optical system. The object height from the top view is actually 2.5 mm as determined in a separate window (not shown here). Given that all parameters are the same except for the object height and the image height, we can observe the difference from the OSLO simulation below.

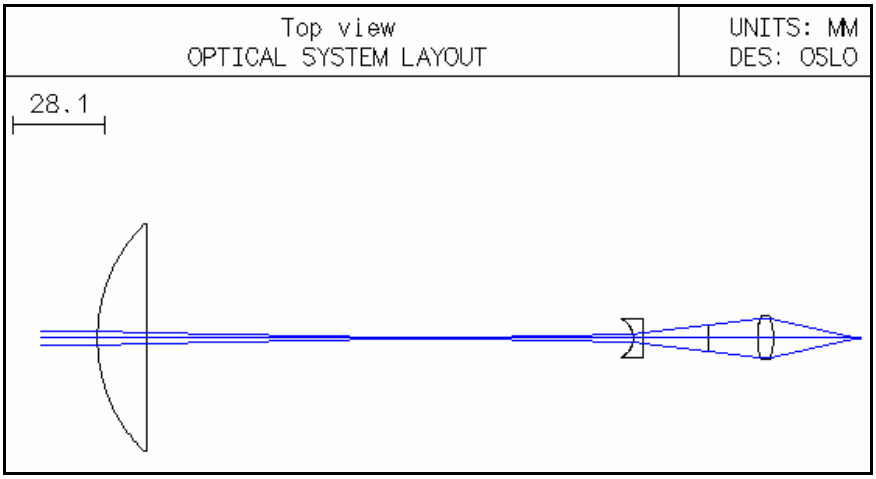


Figure 2-13. Sagittal (Top) view optical system layout.

As we can see from the optical system layout, the rays on the image plane are focused and form an ideal size of 0.15 mm, which means the full size of the theoretical image is 0.3 mm, as required. Since the system fulfills the requirement, the APD array receive sufficient amount of intensity to accurately detect the profile of a vehicle. Notice the aperture stop does not block the light beams from passing through the third lens, but it does eliminate unwanted light beams from passing through into the APD array. In the case where unwanted noises reflect onto the APD array, the system can acquire inaccurate data. To further ensure the validity of the design, we must further investigate other criteria of the system such as the ray intercept plots and the critical height of the system.

OSLO Analysis

The ray intercept plots show the various possible aberrations that can exist in a given optical system. Analyzing the system by checking the ray intercept plots allows us to determine the amount of aberrations in the system.

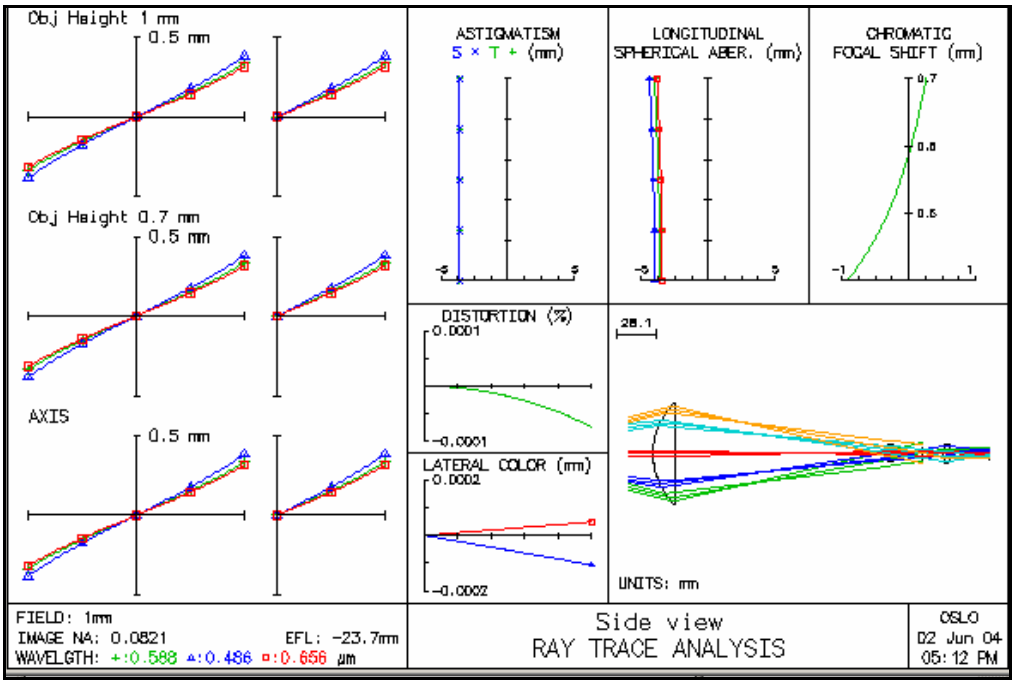


Figure 2-14. Tangential view ray intercept plot.

The tangential ray intercept plots on the very left side of Figure 2-14 show that there is some spherical aberration presents because the ray intercepts are not flat (parallel to the x-axis). The sagittal ray intercept plots next to the tangential plots also show that there are some amounts of spherical aberration (from 0.5 to -0.5 mm). The longitudinal spherical aberration plot on the top right of Figure 2-14 also shows that there are spherical aberrations, which may affect the detection of the system. The plot shows that spherical aberration is not minimized at the surface of the lens because none of the rays intersect the y-axis. Chromatic aberration, however, does cross the y-axis at about 2/3 lens distance from the center of the lens, which shows that chromatic aberration is decently adjusted in the system. Distortion, on the other hand, is also not much of a concern as it shows that the distortion is only 0.0001 to -0.0001 at the edge of the lens.

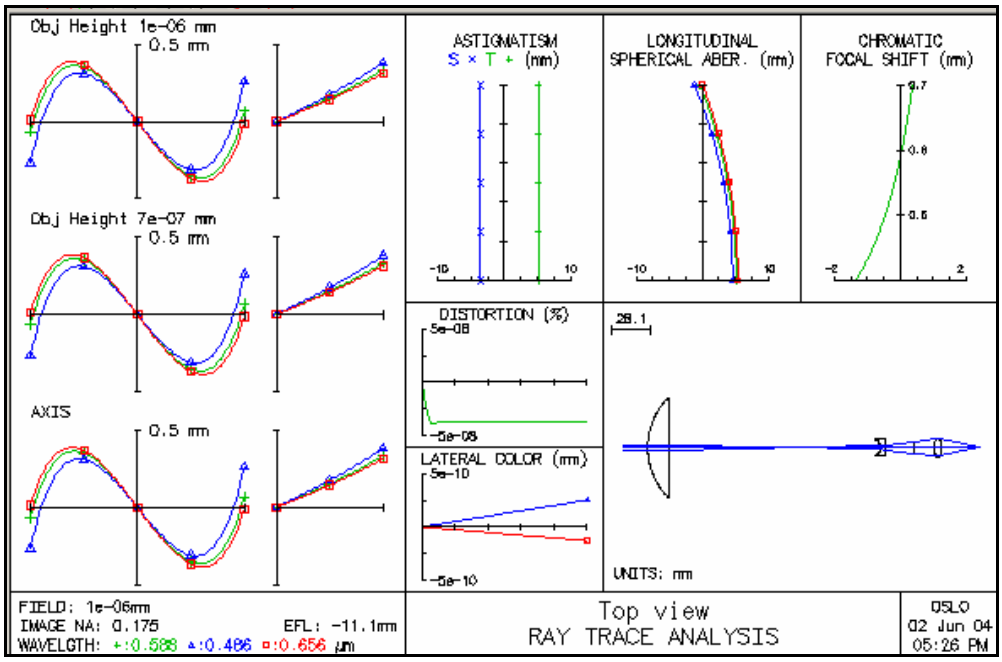


Figure 2-15. Sagittal view ray intercept plot.

To improve the system, we can adjust the system so that spherical aberration, chromatic aberration, and distortion are eliminated, giving the system a better focus, which in turn gives the APD array a better detection of the light source.

From Figure 2-15 above, we can see that spherical aberration from the top view is not much of a concern since the tangential ray intercept plots show that the rays intersect the x-axis at the outer edges of the lens. The longitudinal spherical aberration plot also shows that the rays intersect the y-axis at the edge of the lens, which means spherical aberration is fairly minimal in the system. Chromatic aberration, on the other hand, is also fairly adjusted as it crosses the y-axis at 2/3 of the lens. However, distortion in the top view shows that there is enough distortion to affect the outcome of the system. Therefore, to better adjust the top view of the system, we need to adjust the distortion.

Furthermore, to analyze the validity of the system, we perform calculations of the system's critical height as it was done in [24] giving the proposed system a critical height of 1.07m. The critical height of the designed system is not at the desired position. Since we cannot sacrifice the critical height to accommodate for other requirements such as image size and object size, we will need to adjust the system to obtain a small critical height even if we need to sacrifice the object height.

From the above tables and diagrams, we can see that the newly designed system has a better performance than the older system. However, in this system, having an ideal image height means that we will need to sacrifice the critical height, which is not an option. The system will need to maintain a critical height of 0.25 m, which means we need to have an X of about 6.3 mm, to have a valid system that does not have too large of a field angle. A large field angle will allow the detector to pick up too much noises and unwanted signals. Therefore, to improve the system, we will need to adjust the parameters of the lenses so that we can obtain a desired critical height of 0.25 m. After obtaining the ideal critical height, we can further improve the system by adjusting spherical aberration, chromatic aberration, and distortion of the system.

The proposed optical system is still in the design phase. Once the design is completed, the appropriate lenses and other required opto-mechanical fixtures will be purchased and the design will be tested to ensure that is fully satisfies the optical requirements. The biggest challenge after designing the optical system will be obtaining the designed lenses and aperture stop. The first lens in the designed system is currently not a catalog lens, which means customizing a lens will become very pricey. Therefore,

we will perhaps substitute a lens that has the same amount of optical power and can be purchased from catalogs. Also, the aperture stop posts a problem because we will need an aperture stop that can be adjusted according to the amount of light on a given day. If there is too much sun or a vehicle is giving too great of a reflection, the aperture stop will need to auto-adjust to give the correct amount of intensity to the APD array. Theses problems will be solved in the near future to ensure the quality of the optical system.

2.2.2 Laser Mounts

New laser mounts were created for the LBDS due to the instability caused in the previous design. Figure 2-16 shows the previous design of the laser mounts.

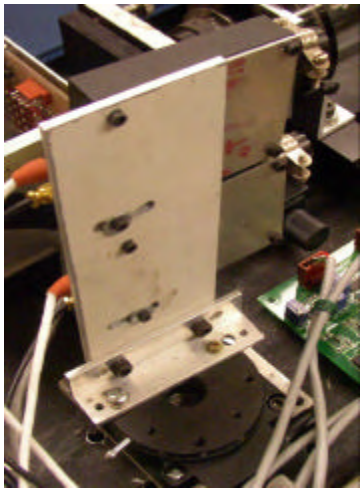


Figure 2-16. Previous version of the laser mounts.

As can be seen from Figure 2-16, each laser is held to the laser mount by two machine screws where the top of the two screws is the pivoting point and the bottom screw slides along a curved slot. Pivoting of the lasers is required in order to align the lasers into one laser line. The curved slot allows for 20 degrees of motion. Since the laser is only fixed to the laser mount by the two rear screws, the lasers are able to vibrate in the vertical direction. During highway testing, the LBDS experiences a lot of vibrations, which can cause the lasers to vibrate. The LBDS is very sensitive to the position of the lasers on the road surface and any minor variation will cause a signal loss. The pivoting motion of the lasers is controlled only by the bottom screw. Notice that the bottom washer has been deformed due to the slot being too wide. The deformation of the washer allows the laser to easily lose its pivoted position causing a loss in signal. A new laser mount was designed and fabricated to reduce the possibility of signal loss due to vibrations.

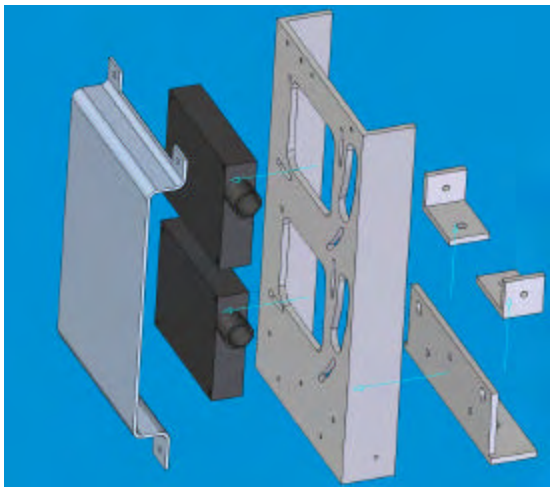


Figure 2-17. Exploded view of the new laser mount design.

Figure 2-17 shows the design of the new laser mounts. The slots used for laser pivoting have been designed to be narrower than the previous design. Washers will no longer be needed but can still be used if desired. Four screws are now used to mount the lasers instead of the previous 2. This keeps the laser snugly fit to the laser mount and reduces the chances of loss of signal due to lateral movement of the lasers or noise due to vibrations.

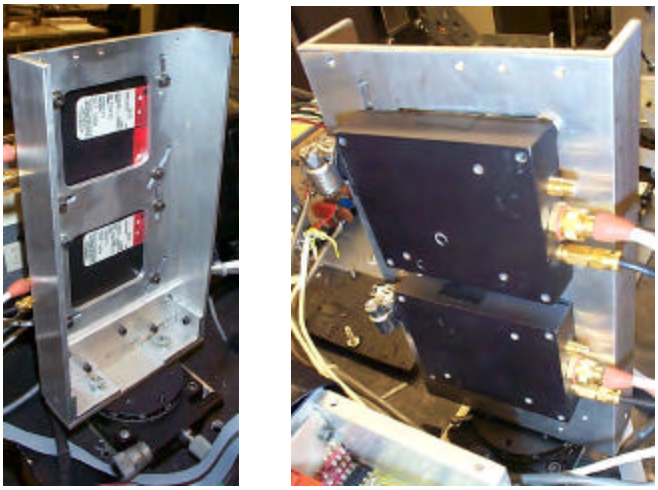


Figure 2-18. Rear and front views of the fabricated laser mount.

Figure 2-18 shows a picture of the fabricated laser mounts mounted to the LBDS. The laser mount back part was created from aluminum C-channel while the laser mount base and laser mount flanges were created from aluminum right angle channel. Most of the machining took place on a CNC mill. The CNC mill allowed for the translation of a DXF file containing the geometry of the part to be machined into milling operations. This is how the laser mount back piece was machined. The combination of the bottom assembly and the C-channel of the laser mount further stabilize the lasers by reducing the ability of the laser mount to sway laterally. The laser mount has not yet gone through a highway test. If signal noise is experienced due to vibrations, then vibration-damping material can be looked into and placed in-between the laser mount and the LBDS base.

2.3 Electronics (signal amplification and shaping)

According to the principle of detection, the system needs only to distinguish whether or not a vehicle is present under the LBDS. It is advantageous to simplify the system by providing a digital output signal from the electronic circuitry. Using a digital signal as output from the hardware will significantly simplify the signal processing in the software, and therefore reduce the computer system requirements. The implementation of this method is based on the high signal-to-noise ratio of the new circuitry. The hardware consists of five parts: the power supply, the laser diode and its trigger controller, the analog signal amplifiers and digital output circuitry, a microcontroller with an Ethernet interface, and a digital I/O board with a computer.

2.3.1 Signal Pre-Processing Circuitry

The high voltage pulse generator for the laser diode is built into the laser unit and is powered with 12 V DC. Because the pulse generator is isolated by a transformer and is well shielded, there is very low noise induced by the pulse. The power supply for the laser units is well isolated by filters as well. As a result, even though we used only one power supply for both laser and sensor electronics, there is no interference between the parts. This reduces the system cost. A 25-element avalanche photodiode (APD) array is used as the sensor in our detection system. This sensor converts the reflected laser light into a current signal.

The sensor circuit is the main part of the electronic hardware in the detection system. In this new version of the electronic hardware, some low-cost amplifier chips with suitable bandwidth, which can meet the high demand of our detection system, were chosen for signal amplification. The time response of the new circuit provides a good match to the pulse length of the laser. The high-frequency signal can be amplified effectively without oscillation. This new circuit increases the signal-to-noise ratio to a factor of 10 relative to the previous circuit. A new method of using a TTL logic circuit, instead of a sample-and-hold amplifier, has been used to handle the short signal pulse. Using this method, a TTL logic circuit is triggered by the amplified signal and the output is a digital signal. Usually the sample-and-hold amplifier is the bottleneck of the circuitry time response, so this method will improve the reliability of the system time response and allow us to increase the amount of signal channels (i.e., to 24 channels). Using a digital signal as output from the hardware will significantly simplify the signal processing in the software. The implementation of a digital output is based on the high signal-to-noise ratio of the new circuitry. The new circuitry is based on simple, cheap, commonly available electronic components. Thus the cost of new circuitry is lower than the previous version. This is vital for the commercialization of our system in the future.

The signal processing circuitry can be divided into three sections: signal pre-amplification, variable threshold comparison, and digital signal output, as shown in Figure 2-19. To describe the working principle clearly, the circuitry for one channel is given in Figure 2-19. The circuitry for the other channels is similar to the schematic in Figure 2-19.

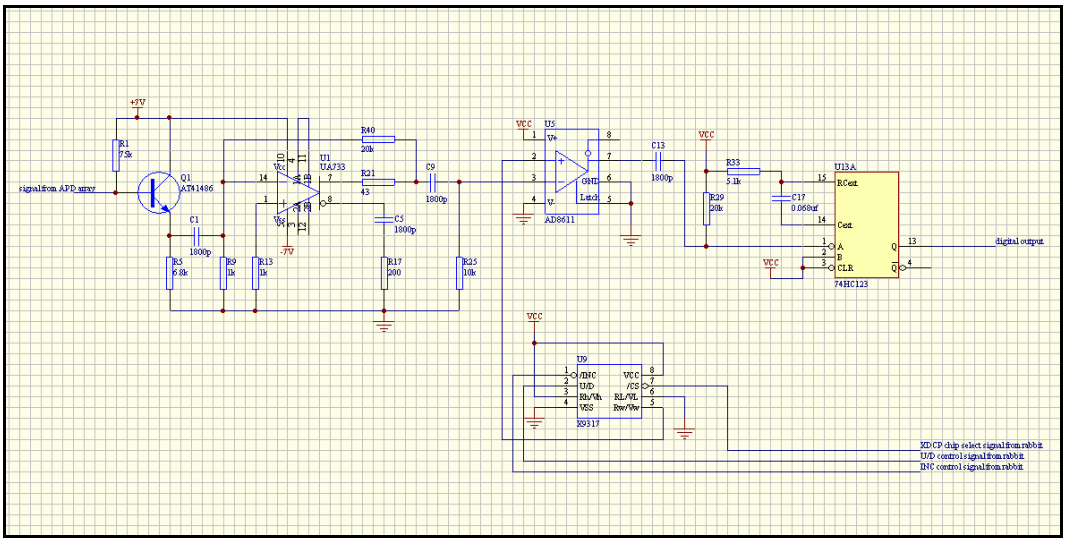


Figure 2-19. Circuitry of sensor electronics.

In the first section of the circuitry, a two-stage pre-amplifier is used to convert the very weak current signal output from the APD array to a voltage signal for further processing. The new circuitry adds a negative feedback from the output of the second amplifier to its input. The new circuitry increases the output of this section from about 2V (corresponding to the old version) to about 4V and lowers the noise to about 0.3V. This improvement makes the system more robust giving it the ability to function effectively in most environments.

The second section includes a comparator and a digital potentiometer. The negative input of the comparator is connected to the output of the pre-amplifier, while its positive input is connected to the wiper output of the digital potentiometer. The voltage on the wiper of the digital potentiometer can be changed under the control of the Rabbit microcontroller. When the pulse amplitude of the output of the pre-amplifier is higher than the wiper output of the digital potentiometer, the comparator will output a negative pulse that will activate the mono-stable trigger in the third section. However, if the pulse output amplitude of the pre-amplifier is lower than the wiper output of the digital potentiometer, no negative pulse will be output from the comparator. The algorithm is described in section 3.

The last section consists of a mono-stable trigger, which converts the pulse output of the comparator to a digital signal. This digital signal will be fed into the Rabbit microcontroller and then sent to the computer through a TCP/IP connection.

Since the laser diodes are triggered every 100μs, the APD will output a narrow pulse every 100μs if the laser beam is not blocked. In this case, the mono-stable trigger will remain in the unstable status (logical "1") because it is triggered continuously by the output signal of the comparator. When the laser beam is blocked by vehicles, the mono-stable trigger will output logical "0". As a result, the presence of the vehicles can be distinguished.

2.3.2 Microcontroller Based Control System

The control circuitry is shown in Figure 2-20. Different from the old version, which uses PIC microcontroller, the new system uses a Rabbit microcontroller as its control core. The circuitry can be divided into 5 sections: the laser diode power driver, digital signal acquisition, digital signal output, control circuitry for signal pre-processing, and push-buttons/indicators.

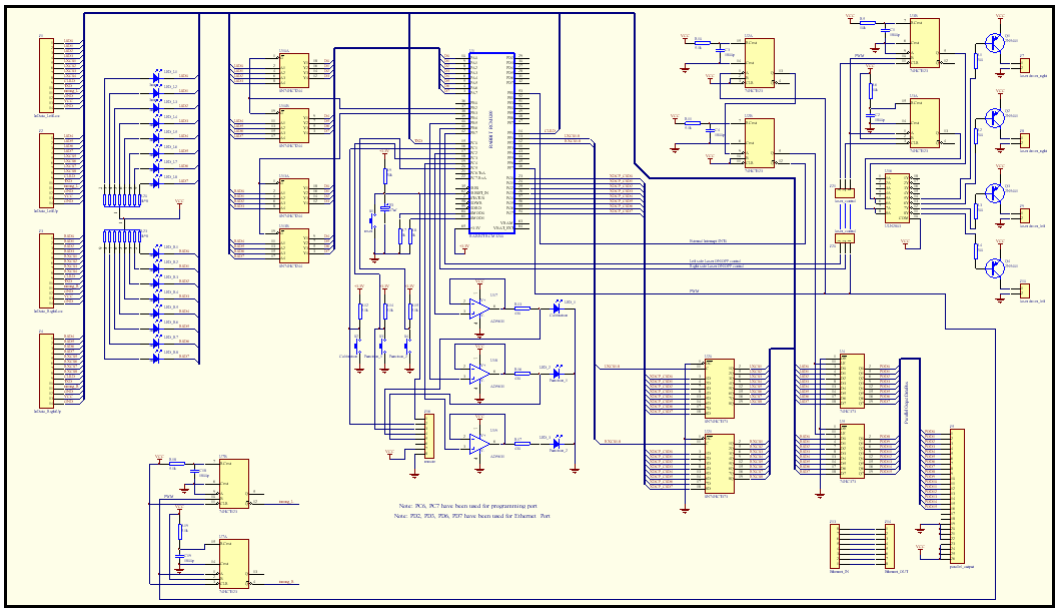


Figure 2-20. Main board of the control system.

Laser Diode Power Driver.

The power driver will generate trigger signals to trigger both (left side and right side) laser diodes. There is a PWM module in the Rabbit microcontroller. When initializing, the PWM module is set to generate a series of PWM pulses (at a frequency of 10 kHz with a duty cycle of 10%). This PWM signal serves as a clock for signal sampling. Two mono-stable triggers are used to generate pulses at both the rising and falling edges of the PWM signal. One rising edge is used to trigger the right side laser module and another is for the left side laser. According to this design, the laser diodes at both left and right sides will be triggered once in every 100 microseconds of the PWM cycle, and the trigger signals for both sides are separated by 10 microseconds.

Data Acquisition Circuitry.

The digital signals from the outputs of the signal pre-processing circuitry are input into this circuitry through the connectors on the most left side of Figure 2-20, and LEDs are connected to input signals (one for each digital signal) to indicate whether the signal level is high or low. An I/O port (port A of the Rabbit) is set to be the input port to read-in the digital signals. The 16 digital signals from the signal processing circuits are input into two buffers (74hct244), and then, the data in the buffers will be read by the Rabbit microcontroller. The data in both buffers are ready 500ns after the second laser beam has been emitted and maintains its value in the next 100µs. The Rabbit microcontroller reads these data about 10µs after the falling edge of the PWM signal to make sure of a correct data reading. Two mono-stable triggers are connected in series. The first one is used to realize the 10µs delay and the second one generates an interrupt-request signal and a latch-lock signal for parallel output. When the Rabbit responds to the interrupt, the digital signals in the buffers will be read in sequentially in the interrupt service routine. With this design, increasing the number of input channels is easily achieved.

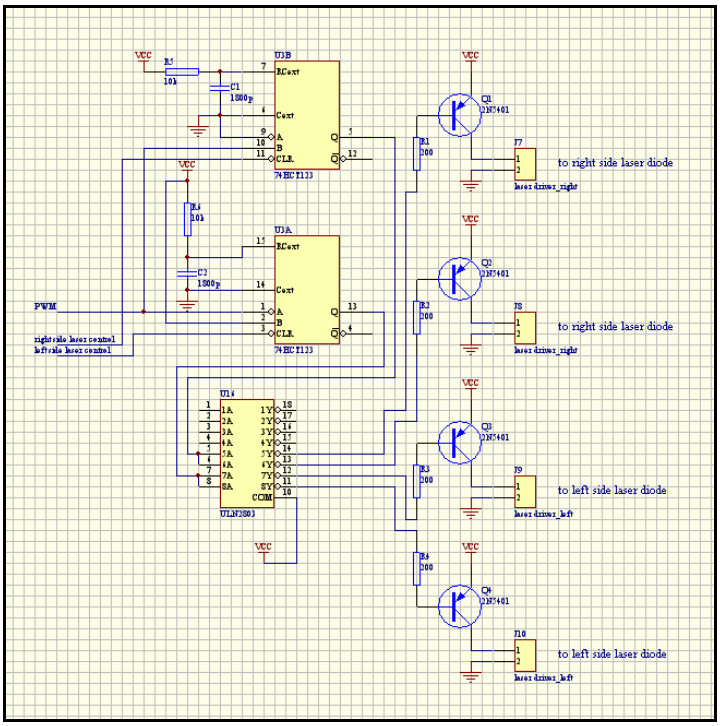


Figure 2-21. Schematic of the laser power trigger

Data Output Circuitry

Two ways of obtaining digital signal outputs have been designed in this circuitry and they can work simultaneously. Both of them are described as follows.

a. Parallel output. Two latches (74hct373) are used for parallel output of the 16 digital signals. The inputs of the latches are connected directly to the connectors on the most left side of Figure 2-20, and their outputs will drive the photocouplers, which are used to connect the LBDS with a DIO96 card in a PC computer. About 10μs after the falling edge of PWM signal, all 16 output digital signals from the signal pre-processing boards will be ready to be read, and the positive pulse generated by the mono-stable trigger at this time will lock these 16 digital signals at the inputs of the latches to their outputs and drive the photocouplers, so that the digital signals will be transmitted to the PC every 100μs. Because the signals are not processed by any microcontroller, the outputs of the latches will be exactly the same as their inputs.

b. Output through an Ethernet port. After the Rabbit core has responded to the interrupt and read in the data, it will add the data to a buffer. When the buffer is full, all data in it will be packed to a data package and then placed in the on-board Ethernet module of the Rabbit RCM3200 for transmission to the server. In the meantime, another data buffer will be started to store the data received. In this way, continuous storage of data can be ensured.

Control Circuitry for Signal Pre-Processing.

The digital outputs of the signal processing circuitry will change their logical outputs according to the output level of the digital potentiometer even though the output from the pre-amplifier remains the same. To retrieve the desired signal from a noisy environment, it is important to set the wiper position of the digital potentiometer correctly. In this circuitry, we have 16 potentiometers. In order to save the I/O port of the Rabbit, we use one I/O port (port G of the Rabbit) to output chip selection signals

for the potentiometers. Port G output chip selection signals twice, each for eight potentiometers, and the total sixteen chip selection signals are locked in two latches (74hct373). To move the wiper positions of the potentiometers, another two I/O lines are used as the signals for $\overline{INC}$ (increment) and $U/\overline{D}$ (up/down) of the potentiometers. When the wiper of a certain potentiometer is to be moved, the corresponding output line of latches, which is connected to the $\overline{CS}$ of the potentiometer, should be set to low level and the direction of increment $U/\overline{D}$ should be set correctly. After this, the line of $\overline{INC}$ outputs a set of negative pulse. The wiper position will be set correctly in this way. More detailed information is provided in section 3.

Command Input and Status Indication

Four push buttons and three LEDs have been designed on the board to act as the human-machine interface. Only one push button and one LED indicator are used for system calibration in the current implementation. The rest of them are reserved for future use.

2.3.3 Power Supply Circuit

The power supply circuit is very important to the LBDS. The reliability and correct operability of the LBDS is based on the stable and low-noise output of the power supply. Since the LBDS needs 5 kinds of DC voltages, +5V, -5V, +12V, -12V, and -300V, the power supply is designed as shown in Figure 2-22.

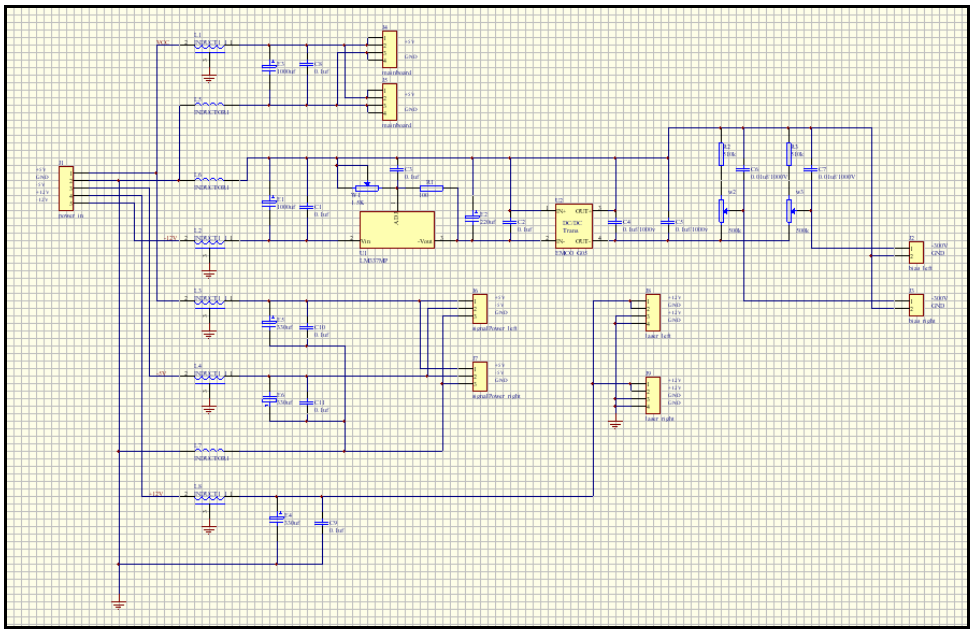


Figure 2-22. Power board.

A switching power supply (not included in Figure 2-22) is used in the system to transfer the AC power supply to DC power of +5V, -5V, +12V, -12V. The +5V and -5V are used for signal pre-processing circuits, and the +12V is used for both laser diodes. The -12V is used to generate the bias voltage for the APD array. This voltage is first fed to an adjustable voltage regulator and then to a DC/DC converter, which can transfer -8V input into a -400V output. Based on the fact that the optimal bias voltages for the APD arrays of both sides are generally different from each other, the bias voltages for both left and right sides are designed to be adjustable separately. To decrease the

power consumption on the potentiometer, the resistance of the potentiometer is selected to be up to 510 kΩ, which is in series with a resistor of 510 kΩ, giving a power consumption of only $(390/2)^2/510000 = 0.0745$ W.

To reduce the noise in the power, some capacitors and inductance filters are designed into the circuitry.

2.3.4 Control Software for the Rabbit Microcontroller

Based on the hardware described above, the control software on the Rabbit Microcontroller is designed to control the system. The program is written mainly in C language, but some parts of the program are written in assembly language when time constraints are important. The program flowcharts are shown in Figure 2-24.

The software consists of a main program, a calibration subroutine and an interrupt service subroutine. The main program is used to send real-time data to the server and test whether re-calibration is necessary. After the system is powered on, the main program initializes the system and then goes into an infinite loop.

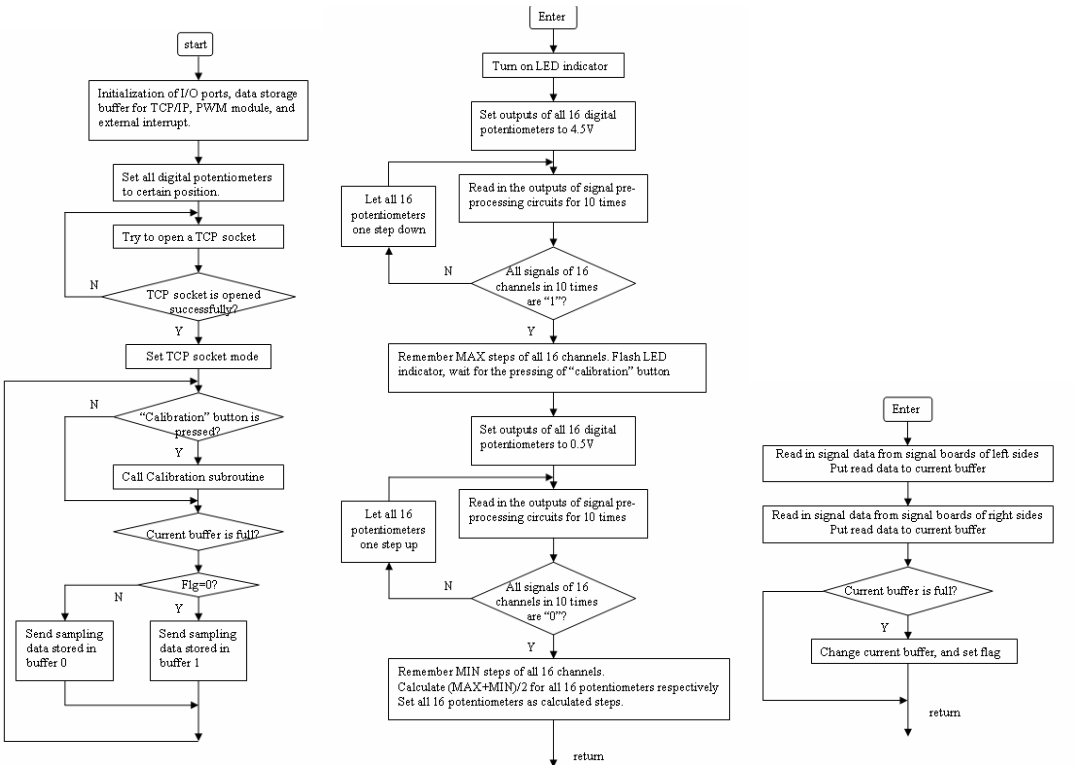


Figure 2-23. The flowchart of the main program.

The calibration subroutine is used to set all 16 digital potentiometers to their proper position. The digital potentiometers have 100 steps. The main idea of the subroutine is as follows: first, all digital potentiometers are set to the highest step (step 100). Then, we lower the wiper positions of potentiometers step by step. Meanwhile, digital signals of all 16 channels will be read in 10 times at every step. If all 10 read-in data of a certain channel are "1", the Rabbit will remember the current position of the potentiometer as "MAX STEP" for this channel. If all 16 channels arrive at their "MAX STEP", the program will change to search the "MIN STEP". At this time, the program will set

all of the 16 potentiometers to step 10, then raise the wiper position step by step; meanwhile, the digital signals of all 16 channels will be read in 10 times at every step. If all 10 read-in data of a certain channel are "0", the Rabbit microcontroller will remember the current position of the potentiometer as "MIN STEP" for this channel. When all 16 channels arrive at their "MIN STEP", the program will stop searching. After this, the program will set all 16 potentiometers to the step according to the calculation of (MAX STEP + MIN STEP)/2 respectively, and finish the calibration. The algorithm is described in section 3.

The interrupt service subroutine is used to read sampling data from signal pre-processing circuits. This subroutine will be carried out once every 100μs. All 16 bits of sampling data will be read-in at the beginning of the subroutine, and then the data will be put into a buffer for transmitting to a computer through an Ethernet cable. To ensure continuous data transfer, two buffers are used in turn to store and send data.

readData()
This is an interrupt service subroutine. It reads in the 16-bit sampling data from port A and stores the data into two array buffers in turn.

setXDCP ()
This function is used to initialize all 16 potentiometers when the system powered on.

setPwm()
This function sets the PWM module with required parameters.

setPortInit()
This function initializes all Rabbit ports that will be used in this system.

calibration()
This function is used to set all 16 potentiometers to their proper positions.

2.4 Data Acquisition and Communication

2.4.1 Rabbit Data Acquisition

The LBDS can utilize two methods for data acquisition. The first is to use a DIO96 PCI board, which is the same method as used in the prior version of the LBDS. This method is still kept in the new version for times when it may be needed. The second way is to use the Rabbit module to read in the digital signals through its parallel ports directly.

Unlike the DIO96 PCI board with sufficient parallel I/O pins, the Rabbit module can only use one 8-bit port to read 8 digital signals at a given time. The Rabbit module provides a total of 44 I/O pins for signal processing. Some of them are used as control signals, for communication, for PWM signal generation, etc. Therefore, one parallel I/O port is used six times to read in the 48 signals during one period of the PWM signal. When a pulse outputs from the Rabbit module to trigger the laser sources, an interrupt is fed to the Rabbit module to read in the digital signals from the parallel port into the buffer. The interrupt routine is written in assembly language to save time due to the stringent requirement that all 48 signals must be read in every 100 microseconds.

2.4.2 Standard TCP/CIP Communication

When the server gets a command "start" from the user program xvehicle, the server will create a TCP socket and wait for a connection from the Rabbit module. After a connection is created through a

server-client handshake, the server will read the data coming from the TCP session and send the data to the user program through FIFOs. The flow chart of the server is shown in Figure 2-24.

The server will listen for the connection from the Rabbit module on the TCP port “2002”. If it is successful, the server will be waiting to read a TCP package sent from the Rabbit module. If it gets the data, it will write the data into FIFOs, which will be read by the user program when data are available in the FIFOs. Meanwhile, when the Rabbit module sends out a package, a tcp_tick (&sock) should be called. Otherwise, the TCP package sending buffer will be full and the data package will not be sent out.

Unlike UDP data package, TCP connections only need to be created once. After a connection is created, the TCP socket can continuously be used to send data without any more connection creation. This reduces the transmission delay of a continuous data stream. It also makes the data communication more reliable and correct. In our system, a TCP package with a size of 600 bytes will be sent out every 10ms

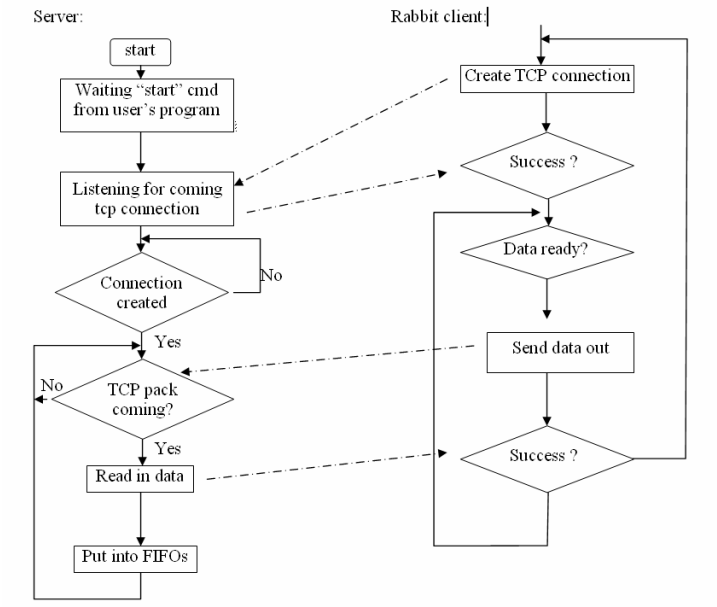


Figure 2-24 The flow chart of TCP/IP communication

2.5 Vehicles Parameters (calculation and display)

To maintain compatibility with the previous system, we still utilize the same programs for calculation of vehicle parameters and display of results. We added an interface API between the TCP data processing and existing parameters processing program. Here, we use a server to receive TCP packages from the rabbit module, restore raw data from the package, and send the received data into the FIFOs, which the previous version program used to transfer data. In this version we discharge the real-time DIO96 device driver.

3 Calibration of Signals

In section 2.3, we discussed the implementation of signal pre-processing in the LBDS. Here, it is necessary to discuss the algorithm of signal calibration to cancel various kinds of environment noise.

3.1 Basic Principle

A Timing Window (TW) [32] is just a narrow pulse used to extract a small regime from an input source as shown in Figure 3-1.. Signals outside the timing window are ignored. In Figure 3-1. a signal $i(t)$ is the original input pulse signal, signal $r(t)$ is a variable bottom line reference signal that is synchronized with *i(t)*, signal $u(t)$ is the digital output and signal $r_0(t)$ is the input reference signal. This simple principle can be used to reject most noise and extract the portion of the signal that is wanted.

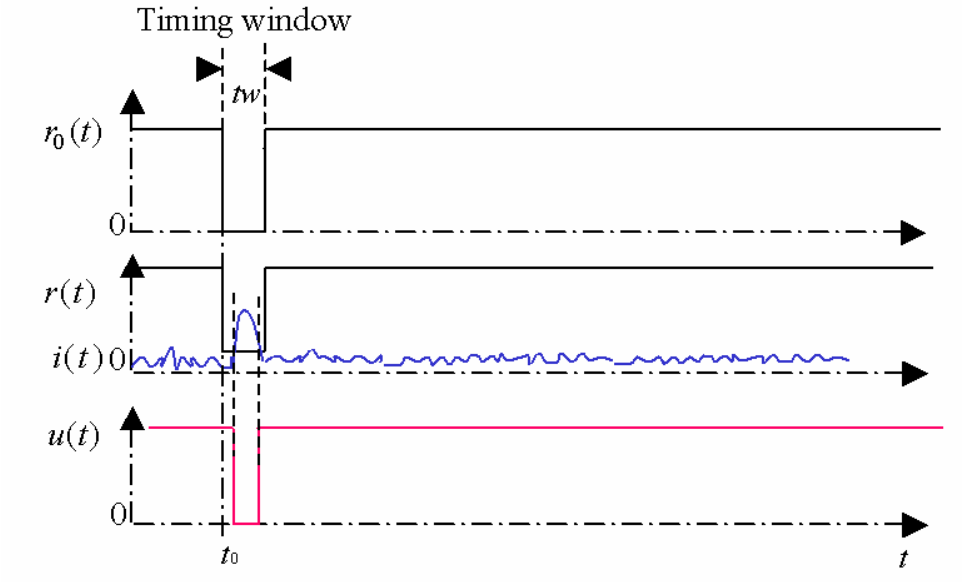


Figure 3-1. Basic timing window

This report assumes that all inputs and outputs follow the Transistor-to-Transistor Logic (TTL) standard in which the highest voltage is 5V and the lowest is 0V. If $i(t) > r(t)$, then the output $u(t)$ will be LOW, otherwise $u(t)$ will be HIGH. The variable reference pulse signal $r(t)$ should be synchronized to $i(t)$ at time t_0 companying the generation of the laser source. Assume that the period of the signal is T, the magnitude of signal is $M_i(t)$ and the magnitude of noise is $M_n(t)$.

Then:

$$i(t) = M_i(t) + M_n(t),$$

$$u(t) = (1 - sign(M_i(t) + M_n(t) - r(t)))/2.$$

When no signal is inputted, the output should be HIGH. $M_i(t) = 0, u(t) = 1$. This gives $r(t)$ as:

$$M_n(t) - r(t) < 0 \Rightarrow r(t) > M_n(t) \qquad (3\text{-}1)$$

In order for the output to be LOW inside of the timing window, $t \in [t_0 + nT, t_0 + tw + nT]$ and $u(t) = 0$, we find that $r(t)$ must be:

$$M_i(t) + M_n(t) - r(t) > 0 \Rightarrow r(t) < M_i(t) + M_n(t) \qquad (3\text{-}2)$$

From eqs(1) and (2), we can get

$$r(t) \in (M_n(t), M_i(t) + M_n(t))$$

Typically, we set

$$r(t) = \begin{cases} M_n(t) + \dfrac{M_i(t)}{2} = \dfrac{M_n(t)(2+S/N)}{2} & t \in [t_0 + nT, t_0 + tw + nT] \\ 5V & t \in [t_0 + tw + nT, t_0 + (n+1)T \end{cases} \quad \text{when} \quad (3\text{-}3)$$

Figure 3-2. shows the implementation of a simple timing window using a comparator in order to produce a digital output. A potentiometer is used to increase or decrease the height of the timing window pulse, making it variable by changing the reference signal. Having a static resister would make the timing window's height static. Knowing the magnitude of the input noise and signal, we can then set the appropriate height of the timing window ($r(t)$) by modifying the potentiometer in order to pick up the signal and filter out the noise. Since $r(t)$ is dependent on $M_i(t)$ and $M_n(t)$, the measurement of $M_i(t)$ and $M_n(t)$ is extremely important and will be discussed in section 3.2.

Figure 3-2. Circuit implementation of a timing window

3.2 Microprocessor Controlled Feedback Loop

As mentioned in the previous section, the variable $r(t)$ is adjusted by a potentiometer (R). A digitally controlled potentiometer (DCP) and a microprocessor to control the DCP are introduced in order to automatically set the proper R_1 which controls $r(t)$ as mentioned in eq(3).

3.2.1 Digitally Controlled Potentiometer

A DCP is controlled by two digital inputs in the form of pulses. One input determines whether to increment or decrement the DCP while the other determines how many steps to take. The resistance of a DCP is quantized into small steps allowing users to slowly increment or decrement the DCP but not as smoothly as a step-less potentiometer.

The amount of steps the DCP can take is DCP dependent. Let us assume n is the total amount of available steps, n_i is the current step number, R is the total resistance of the DCP and R_1 is the set resistive value. Looking at Figure 3-2., if $R_1 = 0$, then $r(t) = 0V$ and if $R_1 = R$, then $r(t) = 5V$. $r(t)$ can vary from $0V$ to $5v$. Therefore:

$$r(t) = \frac{R_1}{R} \times (5 - r_0(t)) = \frac{n_i(5 - r_0(t))}{n} \quad (3\text{-}4)$$

From eqs (3) and (4), we can get the required DCP position (n_i).

$$n_i = \begin{cases} \frac{n}{10} M_n(t)(2+S/N) & t \in [t_0 + nT, t_0 + nT + tw] \\ \forall & t \notin [t_0 + nT, t_0 + nT + tw] \end{cases}$$

where $\forall$ is any value from 0 to n because r_0 equals to 5V. Therefore,

$$n_i = \frac{n}{10} M_n(t)(2+S/N) \tag{3-5}$$

The value $r(t)$ depends on the current step of the DCP (n_i). From eq(5), we get n_i from M_n and the S/N. Therefore, how to get M_n and S/N is a key issue here. We will discuss this in the next section.

3.2.2 *F*eedback Loop

As mentioned in the previous section, $r(t)$ is controlled by a microprocessor through the use of a DCP. The microprocessor sets the proper value of $r(t)$ according to the feedback from the output. Figure 3-3. shows a diagram of the feedback control loop. Since the DCP's resistance is quantized, the system actually operates in discrete control.

Figure 3-3. Microprocessor feedback control loop

The microprocessor will increase with one step of DCP when $u_1(t)$ is HIGH(=1). So we get $r(t)$ as:

$$r(t) = r(t-1) + 5/n * u_1(t-1)$$
$$u_1(t) = (sign(i(t) - r(t)) + 1)/2$$

Here, the initial condition is $r(0) = 0$, and $5/n$ is the step value of the DCP. When the system is in steady state, $t = t_s$, $u_1(t_s) = 0$, the DCP is at a step number N_s. R_s is the steady value of referenced signal when $u_1(t)$ is 0. Therefore:

$$R_s = r(t_s) = r(0) + N_s * 5/n = N_s * 5/n$$

The feedback loop can be used to set the proper value of $r(t)$ during running time. The algorithm of setting the proper value of $r(t)$ is shown in Figure 3-4.. First, the microprocessor gets the value of signal noise N_{sn} through the feedback loop. Then, it gets the N_{si} through the same way. Lastly, the microprocessor sets the DCP at a certain value for the system.

Figure 3-4. Control algorithm

The interrupts of the microprocessor are used to measure N_{si} and N_{sn}. The output from the mono-stable is fed back into the interrupt pin of the microprocessor. The microprocessor will keep incrementing the DCP until an interrupt is detected. When an interrupt occurs, the microprocessor will stop incrementing the DCP. The step count reached is the value of N_{si} or N_{sn}.

The signal $i(t)$ consists of $M_i(t)$ and $M_n(t)$, i.e. $i(t) = M_n(t) + M_i(t)$. The variable $M_n(t)$ is determined from the environment state such as climate, season or time of day and electronic noise, which are uncontrollable. The variable $M_i(t)$ is the pure signal that comes from the signal source that can be controlled by the microprocessor. The microprocessor feedback control loop picks up N_s when $M_i(t)$ =0, indicated by N_{sn}. When the signal source is turned on, another N_s is measured, indicated by N_{si}. When the microprocessor gets the values of both N_{sn} and N_{si}, the proper value of the reference signal $r(t)$ can be obtained by setting $N_r = (N_{si} + N_{sn})/2$. The signal $r(t)$ is calculated as follows:

$$r(t) = \begin{cases} 5(N_{si} + N_{sn})/2n & t \in [t_0 + nT, t_0 + nT + tw] \\ 5 & t \notin [t_0 + nT, t_0 + nT + tw] \end{cases}$$

4 Comparison Study with the Previous System

The new system version provided a significant improvement. We used a Rabbit 3200 module instead of multi-processors, relieving the system of problems like synchronization, calibration, communication etc.

4.1 Synchronization

The previous LBDS had to use several microchips to adjust the DCP and sample signals. Since a microchip has limited functions and limited I/O pins, we needed to use multiple chips to process 48 channels, which is the final goal of the detector. Since each microchip can process 4 signals, 12 chips would be needed to process 48 signals. The multiple chips need to be synchronized with each other in order to sample the data. The new LBDS version utilizes a Rabbit 3200 core module and bus

technology is used to process signals, giving the LBDS the ability to sample more signals than previously possible.

4.2 Precision

The precision of the LBDS has been improved. In the previous version of the LBDS, data were acquired using a PCI DIO board pulsed at 10 kHz using the computer's system clock. The pulsing of the PCI DIO board was not synchronized with the laser pulse trigger of 10 kHz. The new version utilizes a PWM with a 10 kHz pulse, which is generated by the Rabbit module, as the interrupt for data acquisition. This means that the data is acquired at the exact same time as the laser pulse is sent out.

4.3 Communication

The previous version of the LBDS utilized a DIO96 cable to communicate with a computer. The cable was not standard and was difficult to make. The new version uses an Ethernet cable for the communication between the computer and the detector. It is easy to replace and to use. Also it is more scaleable. When several detectors communicate with each other, it is easy to construct a network with Ethernet cables and to transmit all of the data that the detectors acquire.

4.4 Size

The detector is more compact by using a Rabbit module and a laptop instead of a multi-microprocessor system and a desktop computer. Almost every thing, including calibration, PWM generation, signal acquisition, and data communication is done by a Rabbit module making the platform more portable.

5 Testing

The system has been tested in several different ways. Indoor lab testing allows for quick and easy testing of minor changes to the LBDS. Roof testing at Bainer Hall allows for a more accurate test utilizing a vehicle traveling at low speed. A mobile truss has been created to simulate highway tests. Highway tests have been done over highway I-80 in Sacramento. The procedure to align the laser lines and adjust the electronic system is as follows.

5.1 Lab Testing

The lab testing procedure is as follows:

1) Check whether the laser beam is in a proper position on the wall. If the laser line cannot be seen, inspect the laser module opening first. If the laser can be seen in the opening, check whether the laser line is being blocked by an obstacle. If the laser beam cannot be seen in the module opening, check the clock circuit and the 12V DC power supply.
2) Check whether the two laser beams are aligned in one line (both sides).
3) Increase the aperture of the image lens to half opening.
4) Attach the oscilloscope probe to the output of the pre-amplifier (pin 8) and move the laser line by adjusting the screw on the rear of the laser mount until the maximum signal is obtained.
5) Reduce the aperture until the width of the signal is about 100ns and not saturated. .
6) Push the reset button to communicate with a computer.

7) Push the calibration button to set DCP. The output of the digital signal displayed in the computer can be checked in states of blocked or unblocked.

5.2 Roof Testing

We can set up our detector easily and conveniently on the roof of Bainer hall located on the UC Davis campus shown in Figure 5-1. Roof testing allows for low speed vehicle testing. Roof testing is done as a pretest before the detector field test over a highway.

Figure 5-1. Roof testing

5.3 Mobile Truss Testing

In order to avoid testing on the highway often, we can test LBDS on a simulation site on old Hutchinson Road in Davis. The LBDS is mounted to a mobile truss system as shown in Figure 5-2. Since it is impossible for a person to stand on the top of the truss to adjust the LBDS for different height, the LBDS is pre-adjusted to work at a distance of 5 meters in the lab and the truss is set at a height of 5 meters for testing. A remote wire button for reset and calibration is used when the LBDS is tested on the mobile truss. Communication between a computer and the LBDS is accomplished via an Ethernet cable. A single CAT-5 Ethernet cable can simply and easily extend the distance between the computer and the detector.

Figure 5-2. Test on mobile truss

5.4 Outdoor Highway Testing

Field tests have been conducted near the junction of highway I-5 and I-80 in Sacramento. Figure 5-3. is a picture of the test site and the detection system mounted on the highway overpass.

Figure 5-3. Detection system mounted above the highway

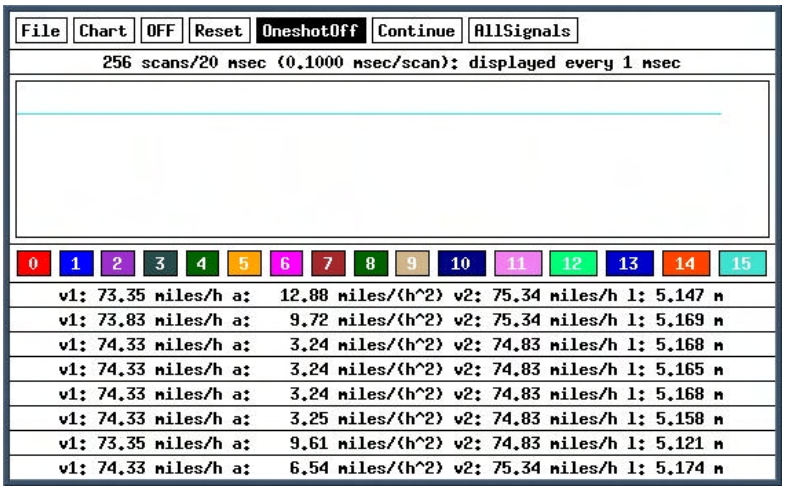


Figure 5-4. Field test result

The results in Figure 5-4. are from the highway test. A digital video camera was used to record the vehicle passing through the detection system. The picture was downloaded from the camera via an IEEE1394 fire wire port. The user interface window contains a menu bar, a status window, and a strip chart. The menu bar is used to control the system, i.e., to start or stop data acquisition and to set parameters of the system. The signals from the electronic sensors are displayed dynamically in the strip chart, which scrolls from right to left. The signal is at a high level when a vehicle is not present, and changes to a low level when a vehicle blocks the laser. This transition occurs over a time that is much shorter than the sampling interval. Since all the signals are displayed at the same position, only one signal line can be seen on the chart when there is a vehicle blocking the laser lines (or no vehicle). Signals from different sensor elements can be distinguished when they change from a low to high level (or inversely) at different times. The front speed (v1), rear speed (v2), acceleration (a), and length (l) of the vehicle are displayed on the lower part of the window. Each line corresponds to one pair of sensor elements. Currently we only display eight pairs of sensor elements.

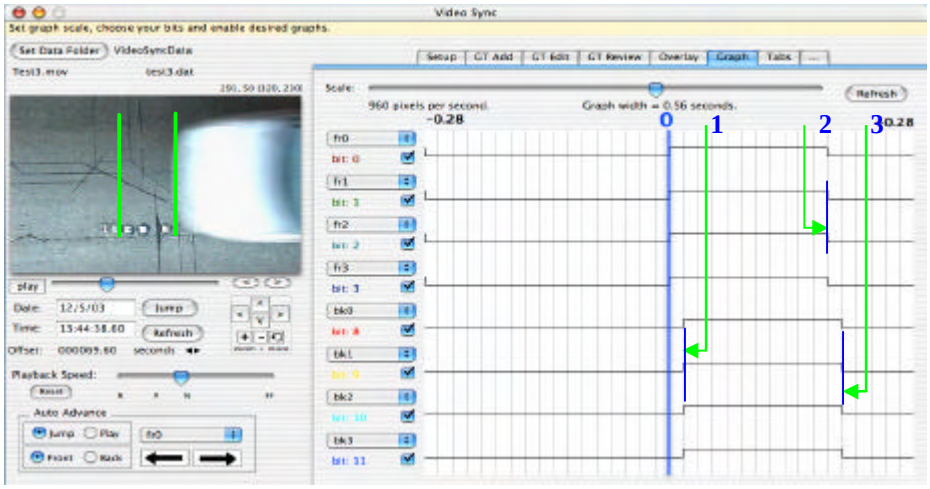


Figure 5-5. Snapshots of the Video Sync program

It is also noted that we have incorporated a video analysis system (Video Sync, provided by CALTrans) for analysis of the performance of the LBDS system. Figure 5-5. shows a snapshot of the Video Sync program. The Video Sync program synchronizes the video recorded during highway testing with the collected data. Since the Video Sync program takes in 60 Hz data encapsulated in the LOG_170 format, a quasi-standard format used in data communication in CaLTrans traffic management system, the 10kHz data output from the LBDS had to be reformatted. The left part of the figure shows the recorded video from highway testing. The right side shows a graph of the collected data synchronized with the video. When the video is played, the data graph will move from right to left. The green lines "A" and "B" are indicators for the laser lines projected onto the road surface. The blue 0 line in Figure 5-5. corresponds to the front edge of the vehicle as it passes through the Laser A beam. The blue 1 line corresponds to the front edge of the car as it passes through the Laser B beam. The blue 2 and 3 lines are the positions corresponding to the rear edge of the car as it passes through the Laser A and Laser B beams. Only 8 of the available 16 channels from the LBDS are visible in Figure 5-5. since the program only supports a maximum of 8 channels. The upper 4 channels in the graph are the front 0-3 channels while the bottom 4 channels are the back 0-3 channels. The Video Sync software gives a visual representation of what the system is actually capturing, giving an easier method to debug the system. It also helps in checking the real-time accuracy of the system's data output. In the results shown, the Video Sync analysis verified that the vehicle bumpers crossed the laser sheets at about the same time.

6 Future Work

6.1 Wireless Networking Communication

There are two possible network configurations for the laser system. The laser system can either be connected to an existing network through an Ethernet cable via the Rabbit processor or through the use of wireless technology. Utilizing wireless network technology is more favorable since the laser systems will be mounted above highways and hooking them up to a network through a hard-line would require routing miles of Ethernet cable for just one system. Figure 6-1. on the following page, shows a network configuration utilizing wireless technology from Cisco.

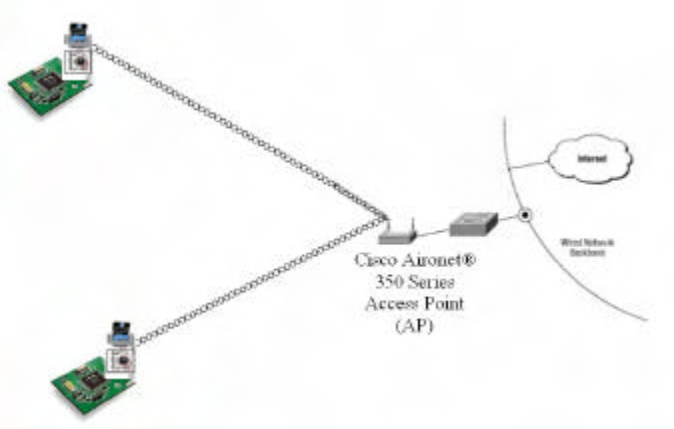


Figure 6-1. Possible Network Configuration

In the example network configuration, we have the Rabbit Processor, which provides the laser system with a 10/100Mbps wireless Ethernet adapter that wireless connects to a Cisco Aironet Access Point. The Access Point is connected to a local network through a switch or hub.

Listed in the table below in Table 6-1 are the required equipment needed to setup up the example network configuration specified in the previous section.

Product name	Qty.
Cisco Aironet 350 Access Point	2
Cisco 1538 Series Micro Hub 10/100	1

Table 6-1. Equipment Requirements for Network Configuration

6.2 Signal Expansion

The current LBDS uses only 8 output signals from the APD array even though there are 24 available output signals. The next step is to expand the system to utilize 24 output signals, which will improve the resolution of the system. Increasing the resolution of the system will allow us to obtain a more accurate profile of the vehicle that passes underneath the system.

6.3 Optical Design

The current optical system is inefficient. It takes too much time and steps to align the optical system. It has been found that any small deviation of the lenses reduces the overall performance of the LBDS substantially. A new optical design will be researched in order to minimize the amount of steps and time required to align the optical system. Furthermore, the image of new optical system will cover 24 sensor elements instead of 8 elements in the current optical system.

7 Conclusions

We have been working on the project of creating a real-time laser-based non-intrusive detection network for the measurement of true travel time on a highway. A powerful Rabbit 3200 was introduced into our system for signal processing, synchronization between signal collection and laser pulse generation, and data communication. The power suppliers of two laser modules were separated in order to further reduce the interference between two laser modules. New circuit boards were added with plug and play ability to give the system expandability and flexibility in updating the software on the rabbit and debugging any unexpected problems. The new system main board is more compact for deploying in the filed usage. The newly designed mechanical parts for aligning the laser beams are more efficient and easy to manipulate.

8 References

[1] Palen, J, The Need for Surveillance in Intelligent Transportation Systems, *Intellimotion*, Vol 6, No. 1, 1997, pp. 1--10.

[2] Palen, J, The Need for Surveillance in Intelligent Transportation Systems --- Part Two, *Intellimotion*, Vol 6, No. 2, 1997, pp. 1--17.

[3] Ostland, M., et al., Simple Travel Time Estimation from Single-Trap Loop Detectors, *Intellimotion*, Vol. 6, No. 1, 1997, pp. 4--11.

[4] MacCarley, C. A., Advanced Imaging Techniques for Traffic Surveillance and Hazard Detection, *Intellimotion*, Vol. 6, No. 2, 1997, pp. 6--15.

[5] Malik, J., and Russell, S., Measuring Traffic Parameters Using Video Image Processing, *Intellimotion*, Vol. 6, No. 1, 1997, pp. 6--13.

[6] Coifman, B. "Using Dual Loop Speed Traps to Identify Detector Errors", *Transportation Research Record no. 1683*, Transportation Research Board, 1999, pp 47-58.

[7] Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance", *Transportation Research: Part C*, vol. 6, no 4, 1998, pp. 271-288.

[8] Zhang, H. M. and Recker W. W. (1999) "On optimal freeway ramp control policies for traffic corridors", *Transportation Research*, B.33(6):471-436

[9] Zhang, H. M. (1999) "A mathematical theory of traffic flow hysteresis," *Transportation Research, B.*33(1):1-23

[10] Palen, J, ITS Roadway Detection Systems Development, personal communication, 1996.

[11] Coifman, B. A.,*Vehicle Reidentification and Travel Time Measurement Using Loop Detector Speed Traps*, Ph.D. Dissertation, University of California, Berkeley, 1998.

[12] Palen, J., Roadway Laser Detector Prototype Design Considerations, personal communication,1996.

[13] Tyburski, R.M., A Review of Road Sensor Technology for Monitoring Vehicle Traffic, *ITE Journal.* Vol. 59, no. 8 (Aug. 1989) Wangler, et al., *Intelligent Vehicle Highway System Sensor and Method*, U.S. Patent No. 5,546,188, 1996.

[14] Olson R. A.; Gustavson R L., Wangler R J., McConnell R E., Active Near-Field Object Sensor and Method Employing Object Classification Techniques, *U.S. Patent* No. 5,321,490, 1994.

[15] Wangler R. J., Gustavson R L., McConnell R E., Fowler K L., Intelligent Vehicle Highway System Sensor and Method, *U.S. Patent* No. 5,757,472, 1998.

[16] Wangler R. J., Gustavson R L., McConnell R E., Fowler K L., Intelligent Vehicle Highway System Multi-Lane Sensor and Method, *U.S. Patent* No. 5,793,491, 1998.

[17] Emerson, L., Mobil Video Surveillance and Ramp-Metering System, *Intellimotion*, Vol 6, No. 2, 1997, pp. 10--12.

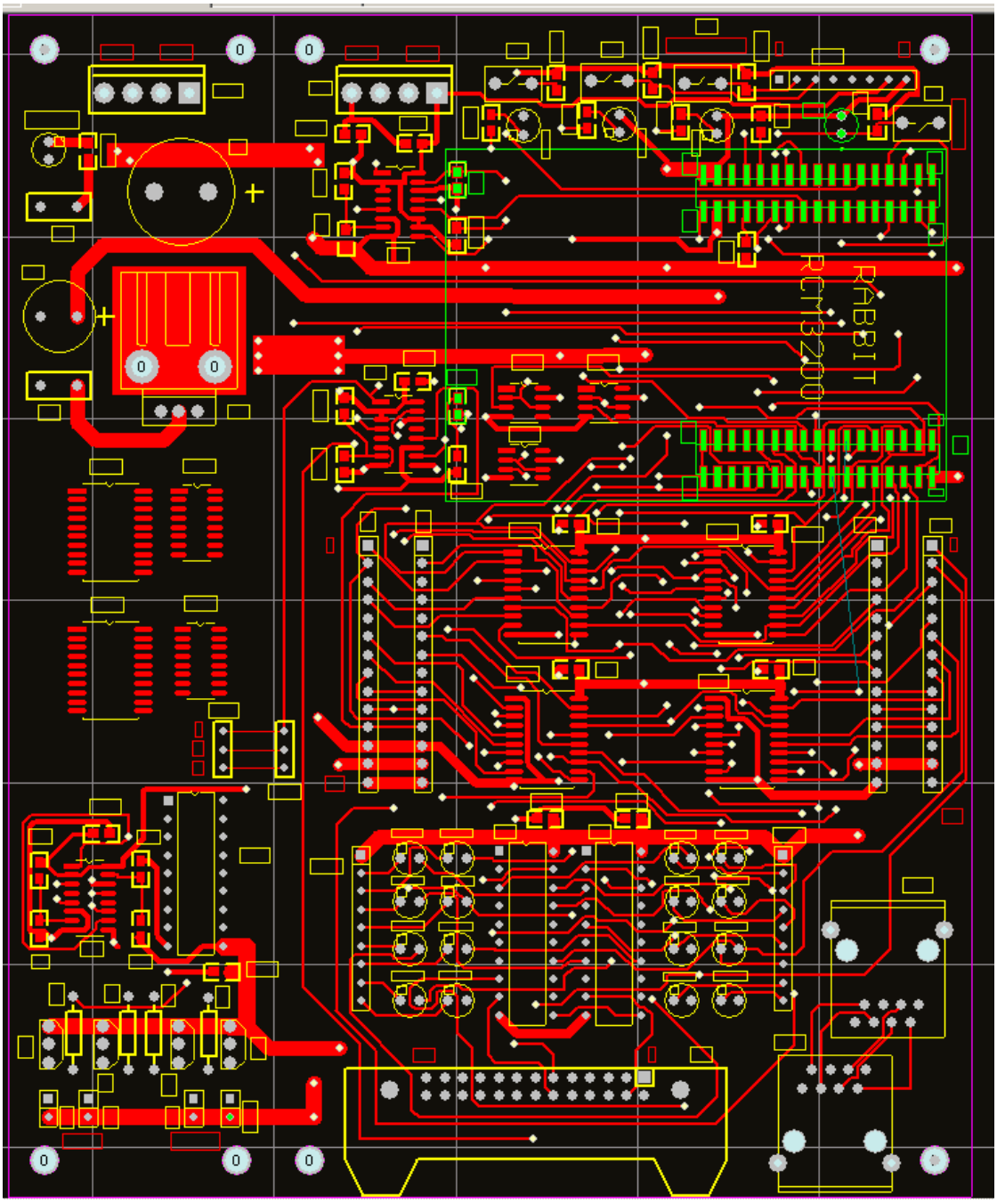
[18] Halvorson, G. A., *Automated Real-Time Dimension Measurement of Moving Vehicles Using Infrared Laser Rangefinders*, MS Thesis, University of Victoria, 1995.

[19] Cheng, H. H., Ben Shaw, Joe Palen, Jonathan E. Larson, Xudong, Hu, Kirk Van Katwyk, A Real-Time Laser-Based Detection System for Measurement of Delineations of Moving Vehicles, IEEE/ASME Trans. on Mechatronics, Vol. 6, No. 2, June 2001, pp. 170-187

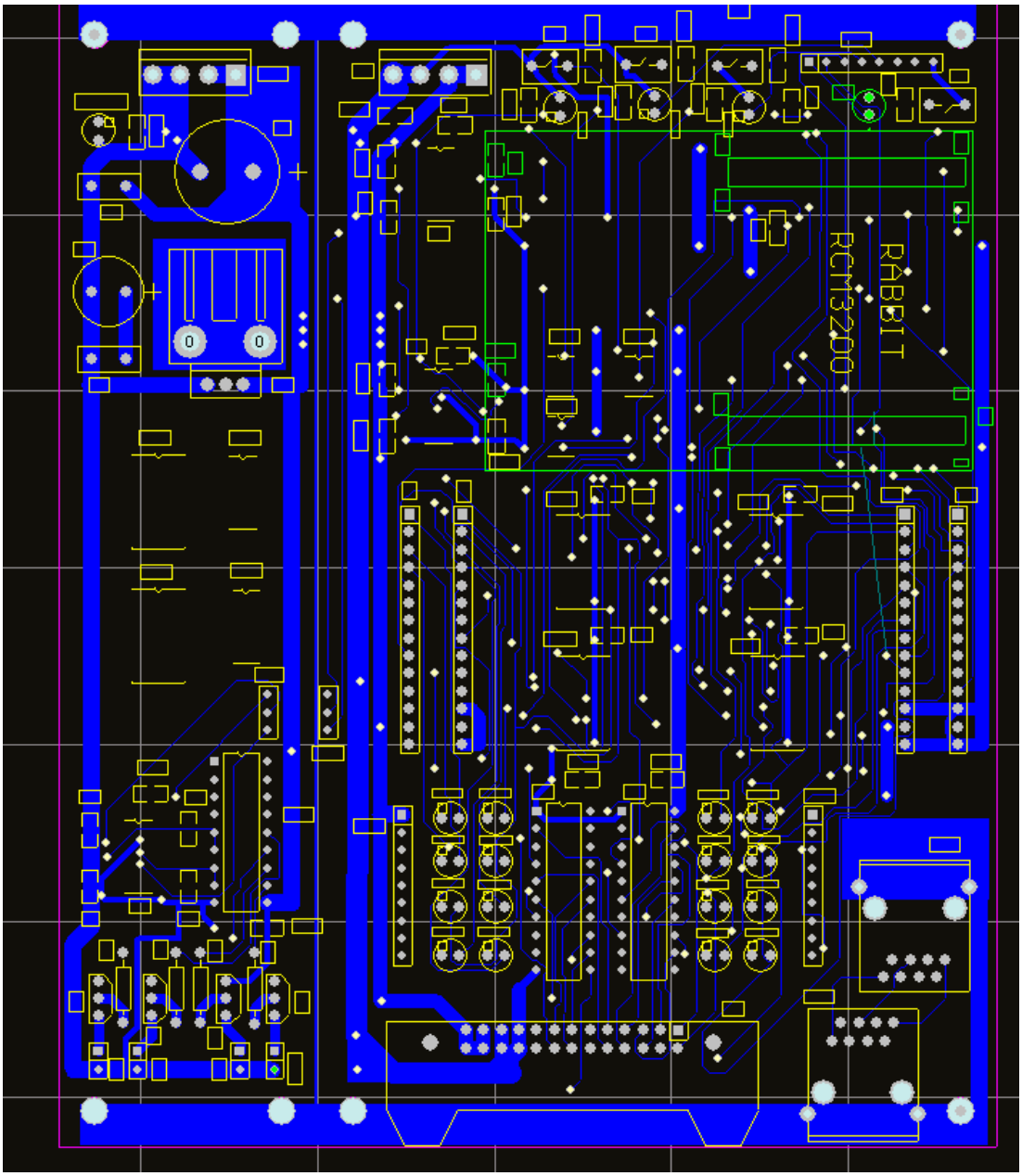
[20] Larson, J. E., Van Katwyk, K., Cheng, H. H., Shaw, B., and Palen, J., *A Real-Time Laser-Based Prototype Detection System for Measurement of Delineations of Moving Vehicles*, California PATH Working Paper, UCB-ITS-PWP-98-20. California PATH Program, Institute of Transportation Studies, University of California, Berkeley, September 1998.
[21] Bin Lin, Harry H. Cheng, Benjamin D. Shaw, Jeo Palen. *Optical and electronic design for a field prototype of a laser-based vehicle delineation detection system.* Optics and Lasers in Engineering 36(2001) 11-27.
[22] *American National Standard for the Safe Use of Lasers*, Laser Institute of America, Orlando, 1986.
[23] Rubini, Alessandro, *Linux Device Drivers*, First Edition, O'Reilly, 1998.
[24] Pedrotti, F. L. Pedrotti, L. S., *Introduction to Optics*, 2nd Ed., Prentice Hall, New Jersey, 1993.
[25] Smith, Warren J., *Modern Optical Engineering: The Design of Optical Systems*, 2nd Ed., McGraw-Hill, Inc., 1990.
[26] Ritchie, D. M. and Thompson, K. L., The Unix Time-Sharing System, *Commun. ACM*, vol.~17, No.~7, July 1974, pp.~365--375.
[27] Thompson, K., Unix Implementation, *The Bell System Technical Journal*, vol. 57, No. 6, July-August 1978, pp.~1931--1946.
[28] Kang, S. and S. Ritchie (1998). "Prediction of Short-Term Freeway Traffic Volume Using Recursive Least Squares and Lattice Filtering." *Proceedings, 5th International Conference on Application of Advanced Technologies in Transportation*, Newport Beach, USA. Edited by C. Hendrickson and S. Ritchie. American Society of Civil Engineers.
[29] Robert Siegel, John R. Howell. "*Thermal Radiation Heat Transfer*" 3rd Ed. 1992
[30] Zhaoqing wang, Harry H. Cheng, Ben D. Shaw, Jeo Palen. "Performance Analysis for Design of a High-Precision Electronic Opto-Mechanical System for Vehicle Delineation Detection on Highway" *Journal of Mechanical Design.* December 2003 -- Volume 125, Issue 4, pp. 802-808
[31] Zhaoqing Wang, Stephen S. Nestinger, Harry H. Cheng, Benjamin D. Shaw, Joe Palen. "REAL-TIME ARCHITECTURE FOR AN ELECTRO-MECH-OPTICAL SYSTEM FOR DETECTION OF VEHICLES ON HIGHWAY" *Proceedings of DETC'04:ASME 2004 International Design Engineering Technical Conferences And The Computers And Information In Engineering Conference*, September 28–October 2, 2004 Salt Lake City, Utah USA
[32] Zhaoqing wang, Harry H. Cheng, Ben D. Shaw, Jeo Palen. "Noise Rejection Using Microprocessor Feedback Control for Variable S/N Ratio Pulse Signals". *Submitted to IEEE Trans. on Instrumentation and Measurement*

Appendix A: Circuit and Board Diagrams

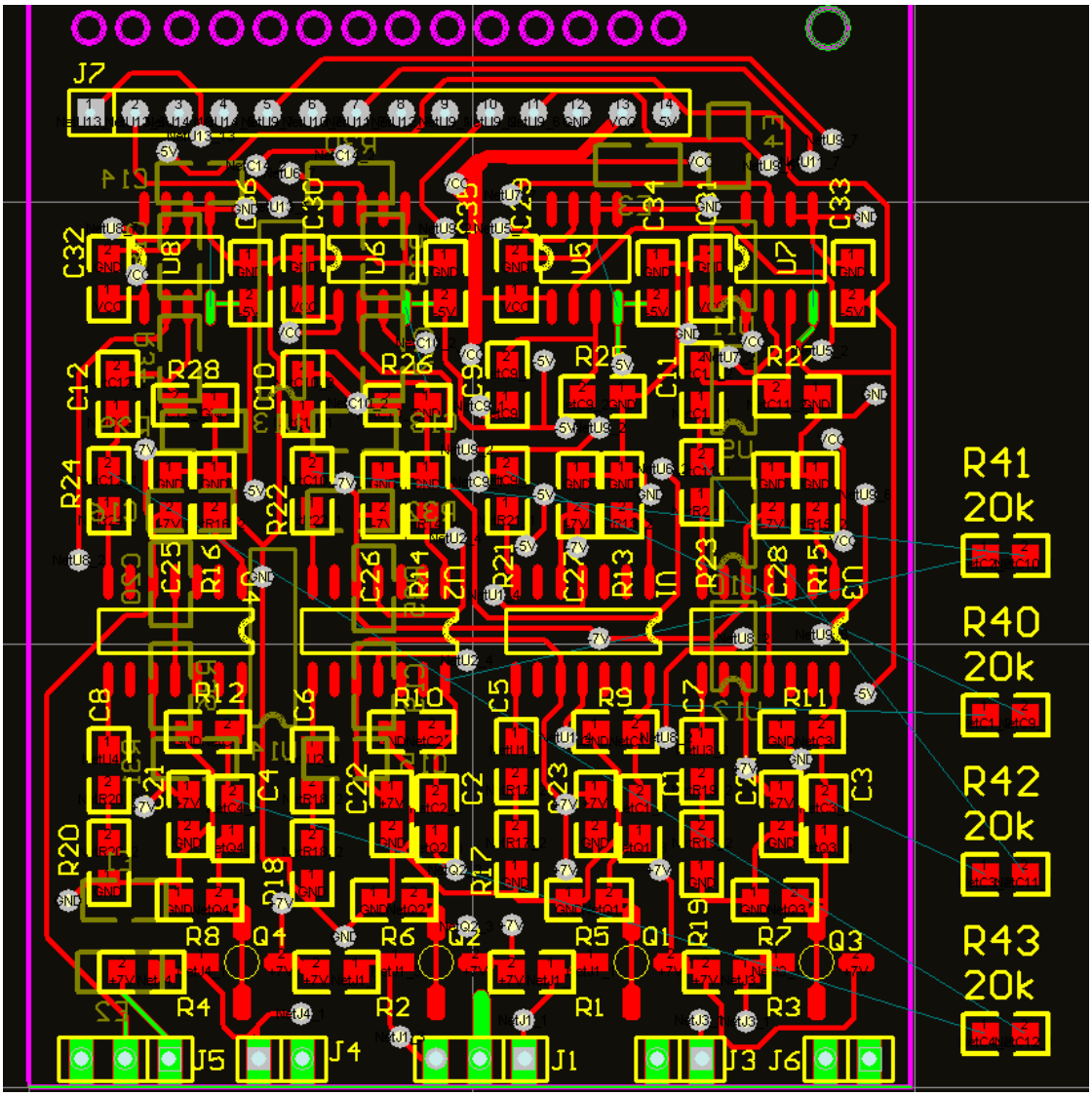
Main control board top layer



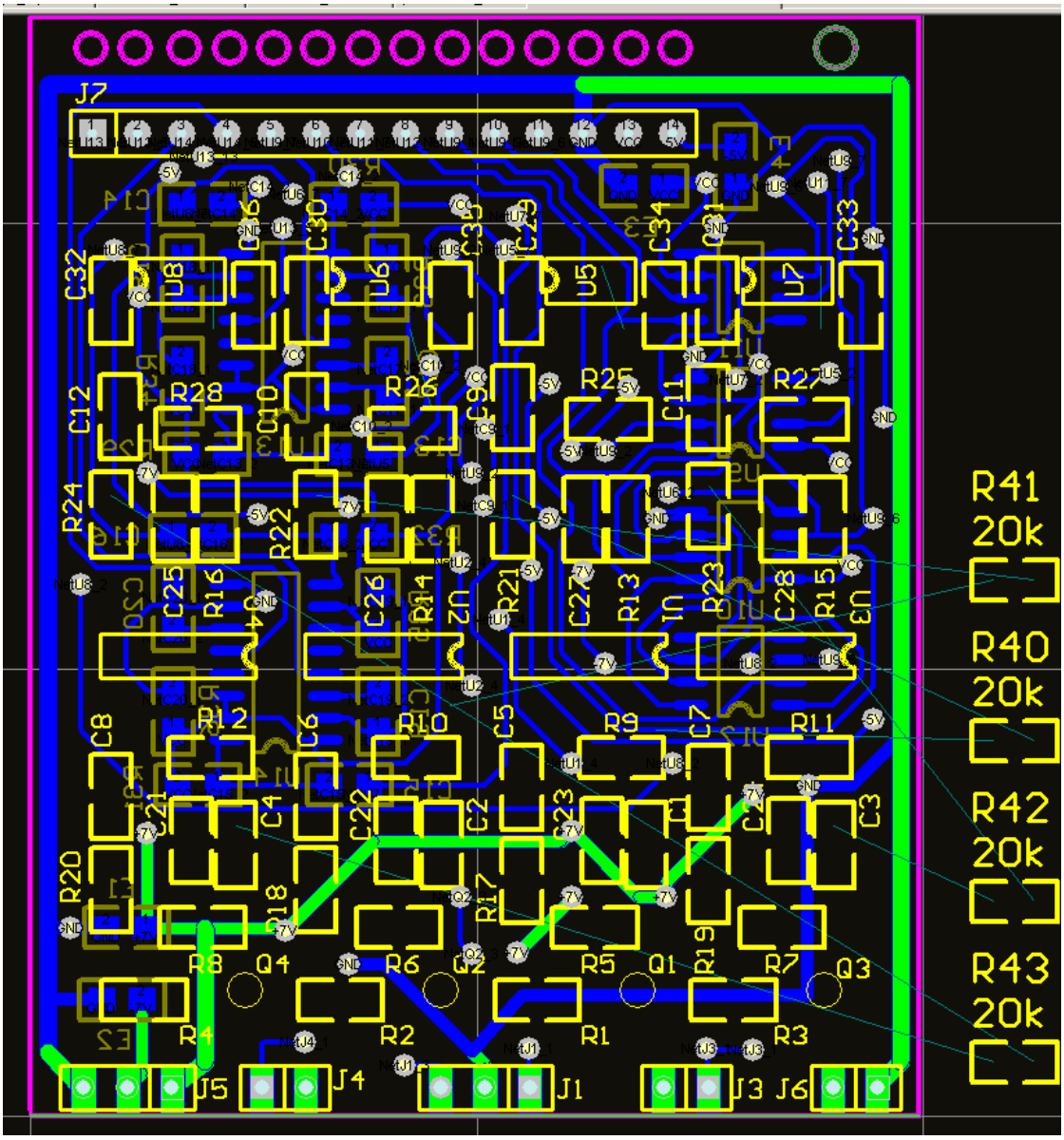
Main control board bottom layer



Signal preamplifier and shaping board top layer



Signal preamplifier and shaping board top layer



Appendix B: Source Code

Source Code in rabbit 3200 for laser pulse and timing window

```
/*************************************************************
 * File Name: LBDSController_3.C                             *
 * Author: Ping Feng                                         *
 * Latest update time: July.11, 2004                         *
 *                                                           *
 * This program is for controlling of LBDS system. It has    *
 * following functions:                                      *
 *  1. Reading all digital output signals of 48 channels from the*
 *      signal boards.                                       *
 *  2. Sending the collected data to PC using TCP.           *
 *    (The Rabbit controller generates 10kHz PWM to trig     *
 *     laser and it is interrupted every 100us to read       *
 *     the data of 6 bytes for 48 channels                   *
 *     from signal boards. After the Rabbit has read 100 data    *
 *     in a period of 100*100us=10ms, these 100 data will be     *
 *     bound to a package which will be sent to PC through       *
 *     Ethernet port. )                                      *
 *  3. Doing Calibration to make the system more reliable.   *
 *************************************************************/

#class auto
/*************************************************************
 * Pick the predefined TCP/IP configuration for this sample.  See
 * LIB\TCPIP\TCP_CONFIG.LIB for instructions on how to set the
 * configuration.
 * Config #100 = lbds_config.lib
 *************************************************************/

/* Defines the size of the buffer that is used to send TCP datagrams
*/
#define ETH_MTU          1000
#define  BUFF_SIZE       (ETH_MTU-40)*2  //must be smaller than
(ETH_MTU - (IP Datagram + TCP Segment))
#define  TCP_BUF_SIZE ((ETH_MTU-40)*8)   // sets up (ETH_MTU-40)*2
bytes for Tx & Rx buffers
#define TCPCONFIG 100

/* Defines the interval(in seconds) to send out TCP packet  */
#define  HEARTBEAT_RATE      5

/* Defines the local TCP port to send packets  */
#define LOCAL_PORT      2801

/* Where to send the heartbeats. If it is set to all '255's, it will
be a broadcast packet  */
#define REMOTE_IP       "169.237.108.228"

/* the destination port to send to */
#define REMOTE_PORT     2002

#memmap xmem
```

```
#use "dcrtcp.lib"

#define D_SIZE 300
#define D_CHANNELS 2
#define PWM_FREQUENCY 10000

/* define storage for the sampling data */
char data0[D_SIZE*D_CHANNELS+10];
char data1[D_SIZE*D_CHANNELS+10];
char *dataa;
char *dataa0;
char *dataa1;

/* Define storage for calibration  */
char XDCP_max[17], XDCP_min[17];      // 0-7 for left side, 8-15 for
right side
char testdata_left, testdata_right;   //storage for read data when
calibration
char *testdata_left_ptr, *testdata_right_ptr;    // pointers of the
variables
tcp_Socket tsock;
int  count;
char flg, intflag;                    // intflag to indicate Rabbit
has been interrupted
char send_num;

/*********************************************************
 *    Function Declaration
 *
 *********************************************************/
void readData();
void outData();
int  sendPacket(char *buf);

void setPortInit();
void setDataInit();
void setInterrupt();
void setPwm(int requency);
void setHDCP();
void printResults();
void dispData(char *buf);
int  openTcpSock();

void setXDCP();
void calibration();
void errorIndicate();

int main() {
   int cal_key;
   char oldflg;
   oldflg;
   setPortInit();
   setDataInit();
   setPwm(PWM_FREQUENCY);
   setInterrupt();
   //save the address in the pointer
```

```
  testdata_left_ptr = &testdata_left;
  testdata_right_ptr = &testdata_right;
  //save the address in the pointer

  setXDCP();

  oldflg=0;
  setHDCP();
  if(ifstatus(IF_ETH0))
    printf("ETH0 is up now.\n");
  printf("MTU: %d ", ETH_MTU);
  //printf("BUFF_SIZE:%d ",BUFF_SIZE);
  printf("TCP_BUF_SIZE:%d\n",TCP_BUF_SIZE);

  //open a TCP socket for communication
  while(openTcpSock()<=0);

  sock_mode(&tsock,TCP_MODE_ASCII);

  while ( 1 ) {
    if(BitRdPortI(PCDR, 1)==0) {
      costate {
        waitfor( DelayMs( 100L ) );
        if(BitRdPortI(PCDR, 1)==0) {
          while(1) {
            if(BitRdPortI(PCDR, 1)==1) {
              calibration();
              break;
            }
          }
        }
      }
  }

  printf("left=%x\tright=%x\n", testdata_left, testdata_right);

  if (oldflg!=flg) {
      switch (flg) {
        case 0:  //data1 is ready for send
          oldflg = flg;
          if(tcp_tick(&tsock))
            sendPacket(data1);
          break;
        case 1:  //data0 is ready to send
          oldflg = flg;
          if(tcp_tick(&tsock))
            sendPacket(data0);
          break;
        case 2:
          //adjust potentiometer
          //setXDCP();
          break;
        default:
          break;
      }                       // end of switch( flg )
    }                         // end of if(olflg!=flg)
  }                           // end of while()
```

```
}


/*********************************************************************
*********
 * setHDCP() For DHCP setup
 *
 * A very basic TCP/IP program, that will initilize the TCP/IP
interface,
 * if set TCPCONFIG as 3, DHCP (Dynamic Host Configuration Protocol)
or
 * BOOTP (Bootstrap Protocol) will be used to obtain IP addresses
and other
 * network configuration items.
 * Otherwise, using the static IP address for this rabbit.

*********************************************************************
********/
void setHDCP() {
  int status;
#if TCPCONFIG==3
  // Set runtime control for sock_init()...
  ifconfig(IF_DEFAULT,    // (DHCP only works on default interface)
    IFS_DHCP_TIMEOUT, 6,  // Specify timeout in seconds
    IFS_DHCP_FALLBACK, 1, // Allow use of fallbacks to static
configuration
    IFS_ICMP_CONFIG, 1,   // Also allow use of directed ping to
configure (only if DHCP times out).
    IFS_UP,
    IFS_END);
  printf("Starting network (max wait %d seconds)...\n",
_bootptimeout);
#endif

  status = sock_init();

  switch (status) {
    case 1:
      printf("Could not initialize packet driver.\n");
      exit(1);
    case 2:
      printf("Could not configure using DHCP.\n");
      break;  // continue with fallbacks
    case 3:
      printf("Could not configure using DHCP; fallbacks
disallowed.\n");
      exit(3);
    default:
      break;
  }
#if TCPCONFIG==3
  if (_dhcphost != ~0UL)
    printf("Lease obtained\n");
  else {
    printf("Lease not obtained.  DHCP server may be down.\n");
    printf("Using fallback parameters...\n");
```

```
  }
#endif
  printResults();
}

void outData(char flg) {
  BitWrPortI(PCDR, &PCDRShadow, flg, 4);
}

/************************************************
 * dispData(char *buf)
 * the buf is a data read by readData in interrupt
 * this function typically used as debug
 ************************************************/
void dispData(char *buf){
  static int i;
  printf("flg=%d\n",flg);
  for( i = 1; i <= D_CHANNELS*D_SIZE; i++){
    printf(" %d", (unsigned)buf[i-1]);
    if( i%D_CHANNELS == 0 )
      printf("\n");
  }
}


/************************************************
 *  setPortInit() set ports used as input or output
 *
 ************************************************/
void setPortInit() {
  WrPortI(SPCR, &SPCRShadow, 0x80); // set port A as all inputs,
p126
  WrPortI(PBDDR, &PBDDRShadow, 0xff); // set port B as all outputs,
p127
  WrPortI(PFDDR, &PFDDRShadow, 0xff);  // set port F as all outputs
  WrPortI(PGDDR, &PGDDRShadow, 0xff);  // set port G as all outputs


  BitWrPortI(PDDDR, &PDDDRShadow, 1, 4); // set PD4 as output
  BitWrPortI(PDDDR, &PDDDRShadow, 1, 5); // set PD5 as output
  BitWrPortI(PDDCR, &PDDCRShadow, 0, 4); // set PD4 as standard
output, not open drain
  BitWrPortI(PDDCR, &PDDCRShadow, 0, 5); // set PD5 as standard
output, not open drain

  BitWrPortI(PFDCR, &PFDCRShadow, 1, 0); // set PF0 as open drain,
to drive the x9317

  WrPortI(PBDR, &PBDRShadow, 0xdf); // set PB0-PB4=1, PB5=0, PB6-
PB7=1
  WrPortI(PGDR, &PGDRShadow, 0xff); // set all output of port G as
"1", PG0-PG7=1
  WrPortI(PFDR, &PFDRShadow, 0xff); // set all output of port F as
"1"
  //WrPortI(PCDR, &PCDRShadow, 0xea); // set PC0=0, PC2=0, PC4=0,
others="1"
```

```
  WrPortI(PCDR, &PCDRShadow, 0xeb);   // set PC0=1, PC2=0, PC4=0,
others="1", for TXD use ULN2803

       BitWrPortI(PFDR, &PFDRShadow, 1, 0); //set INC(CLKD) of XDCP
to 1
       BitWrPortI(PCDR, &PCDRShadow, 0, 0); //set PC0=0, U/D(TXD) of
XDCP = "1", count up

       BitWrPortI(PFDR, &PFDRShadow, 0, 1); //set PF1=0
       BitWrPortI(PFDR, &PFDRShadow, 0, 2); //set PF2=0,  set 16 CS
of all XDCPs to "1"
}


/*****************************************
 *   initializing the interrupter
 *   set the external INT0
 *   and interrupter handler as readData()
 *****************************************/
void setInterrupt() {
   //enable external INT0 on PE0, set the interrupt routin to
readData
   SetVectExtern3000(0, readData);
   //refer to p_95 00001001
   WrPortI(I0CR, &I0CRShadow, 0x05); // enable external INT0 on PE0,
                                     // falling edge,  priority 1
   WrPortI(I1CR, &I1CRShadow, 0x00); // disable external INT1 on
PE1,
                                     //
}

void setPwm(int frequency){
    unsigned long   freq;
    int pwm_options;

    // request frequency PWM cycle (will select closest possible
value)
    freq = pwm_init(frequency);
    printf("Actual PWM frequency = %lu Hz\n", freq);

    pwm_options = 0;

    pwm_set(3, 0.1 * 1024, pwm_options);   //set PF7 as the PWM
output port with 10% duration
}

void setDataInit() {
   int i;
   memset(data0, 0x55, D_SIZE*D_CHANNELS+1);
   memset(data1, 0xff, D_SIZE*D_CHANNELS+1);
   dataa = NULL;
   dataa0 = data0;
   dataa1 = data1;

   send_num =0x31;
   count = 0;
   flg = 0;
```

```
}

/****** interrupt routines
*****************************************
 *
 *  readData(),
 *
 *  use PE0 as INT. the data will be read from the input port
 *
 *  Data read from PORT A, restore in the data0[D_CHANNELS*D_SIZE]
when flg=1
 *                                        data1[D_CHANNELS*D_SIZE]
when flg=0
 *  Each interrupt reads the PORT A for D_CHANNELS times to get 48
channels from
 *  input data buffer. the PORT B is used to select which 8 channels
 *  will be read:
 *
 *  for 0-7   channels, set PB2 as 1
 *      8-15  channels, set PB3 as 1
 *      16-23 channels, set PB4 as 1
 *
 *      24-31 channels, set PB5 as 1
 *      32-39 channels, set PB6 as 1
 *      40-47 channels, set PB7 as 1
 *

*********************************************************************
**/
nodebug root interrupt void readData() {

   intflag=1;

   #asm
    ioi ld a, (PBDR)
    or 0x0c       ; set PB2,PB3 "1", not select all hc244
    or  0x01   ; set PB0=1, to indicate the start of interrupt
    ioi ld (PBDR), a
   #endasm

   //read data from PORT A into data0 array
   //outData(0);
   switch (flg) {
      case 0:
        #asm
         ;select left side 1-8 channels
          ioi ld a, (PBDR)
          and 0xfb ;set PB2 to 0, select hct244 for left side
channel 1-8
          ioi ld (PBDR), a

          ld hl, (dataa0)
          ld (dataa),hl

          ; dataa =  A7 A6 A5 A4 A3 A2 A1 A0
          ; channel: 8  7  6  5  4  3  2  1
          ioi ld a, (PADR)  ; get the data from the port A
```

```
          ld hl, (dataa)    ; get the address of dataa
          ld (hl), a        ; let data = a value

          ld de, (testdata_left_ptr); get the address of
testdata_left


          ld (de), a     ; save the data to testdata_left

          ioi ld a, (PBDR)
          or 0x0c
          ioi ld (PBDR), a ; set PB2,PB3=1, not select all hc244

          ;select right side 1-8 channels
          ioi ld a, (PBDR)
          and 0xf7 ; set PB3 to 0, select hct244 for right sied
channel 1-8
          ioi ld (PBDR), a

          inc hl
          ld (dataa),hl

          ioi ld a, (PADR) ; get the data from the port A
          ld hl, (dataa)   ; get the address of dataa
          ld (hl), a       ; let data = a value

          ld de, (testdata_right_ptr) ;get the address of
testdata_right
          ld (de), a  ;save the data to testdata_right

          inc hl
          ld (dataa),hl
          ld (dataa0),hl   ;for 16 channels
                           ;for 48 channels refer to the bottom of
                           ;this section{#asm ...#endasm}

          ioi ld a, (PBDR)
          or 0x0c
          ioi ld (PBDR), a  ; set PB2,PB3=1, not select all hc244
        #endasm
        break;              //end of case 0

      case 1:
      //read data from PORT A into data1[] array
        #asm
          ;select left side 1-8 channels
          ioi ld a, (PBDR)
          and 0xfb  ; set PB2 to 0, select hct244 for left side
channel 1-8
          ioi ld (PBDR), a

          ld hl, (dataa1)
          ld (dataa),hl

          ; dataa =  A7 A6 A5 A4 A3 A2 A1 A0
          ; channel: 8  7  6  5  4  3  2  1
          ioi ld a, (PADR) ; get the data from the port A
```

```
          ld hl, (dataa)   ; get the address of dataa
          ld (hl), a       ; let data = a value

          ld de, (testdata_left_ptr) ; get the address of
testdata_left
          ld (de), a   ; save the data to testdata_left

          ioi ld a, (PBDR)
          or 0x0c
          ioi ld (PBDR), a ; set PB2,PB3=1, not select all hc244

          ;select right side 1-8 channels
          ioi ld a, (PBDR)
          and 0xf7  ; set PB3 to 0, select hct244 for right sied
channel 1-8
          ioi ld (PBDR), a

          inc hl
          ld (dataa),hl

          ioi ld a, (PADR) ; get the data from the port A
          ld hl, (dataa)   ; get the address of dataa
          ld (hl), a       ; let data = a value
          ld de, (testdata_right_ptr) ; get the address of
testdata_right
          ld (de), a  ; save the data to testdata_right

          inc hl
          ld (dataa),hl
          ld (dataa1),hl  ; for 16 channels data
                          ; for 48 channels refer to the bottom of
                          ; this section{#asm ...#endasm}

          ioi ld a, (PBDR)
          or 0x0c
          ioi ld (PBDR), a  ; set PB2,PB3=1, not select all hc244
        #endasm
        break;                //end of case 1

      case 2:
        // adjust the potentiometer
        //SetXDCP();
        break;

      default:

  }                                 //end of switch(flg)
  #asm
    ioi ld a, (PBDR)
    or 0x0c    ; set PB2,PB3 "1", not select all hc244
    and 0xfe   ; set PB0=0, to indicate the end of interrupt
    ioi ld (PBDR), a
  #endasm

  if( flg <=1 ) {
    count += D_CHANNELS;
    if(count >= D_CHANNELS*D_SIZE){
```

```
      count = 0;
      if(flg){
        flg = 0;
        dataa1 = data1;
      }
      else{
        flg = 1;
        dataa0 = data0;
      }
    }
  }
  //outData(1);
}

/************************************************
 * Print some of the DHCP parameters received.
 ************************************************/
static void printResults(void) {
    auto long tz;
    auto word i;

    printf("Network Parameters:\n");
    printf("  My IP Address = %08lX\n", my_ip_addr);
    printf("  Netmask = %08lX\n", sin_mask);
#if TCPCONFIG==3
    if (_dhcphost != ~0UL) {
        if (_dhcpstate == DHCP_ST_PERMANENT) {
            printf("  Permanent lease\n");
        } else {
            printf("  Remaining lease = %ld (sec)\n", _dhcplife -
SEC_TIMER);
            printf("  Renew lease in %ld (sec)\n", _dhcpt1 -
SEC_TIMER);
        }
        printf("  DHCP server = %08lX\n", _dhcphost);
    }
#endif
    if (gethostname(NULL,0))
        printf("  Host name = %s\n", gethostname(NULL,0));
    if (getdomainname(NULL,0))
        printf("  Domain name = %s\n", getdomainname(NULL,0));
#if TCPCONFIG==3
    if (dhcp_get_timezone(&tz))
        printf("  Timezone (fallback only) = %ldh\n", tz / 3600);
    else
        printf("  Timezone (DHCP server) = %ldh\n", tz / 3600);
#endif
    for (i = 0; i < *_last_nameserver; i++)
        printf("  DNS server #%u = %08lX\n", i+1,
def_nameservers[i]);
    ip_print_ifs();
    router_printall();
}

/************************************************************
 * int sendPacket(char *buf)
 *
```

```
 * Send data to the remote Data. the zise of package is 601.
 * It should consider the situation:
 *    that the data can't send out because the send-buffer is full.
 *
 *
*****************************************************************/
int sendPacket(char *buf) {
    static long sequence;
    auto int    length, retval;
    auto int    ltlen;
#GLOBAL_INIT
    {
        sequence = 0;
    }

    /* fill the packet with interesting data (a sequence number) */
    //memset(buf, 'C', sizeof(buf));   //for test
    sequence++;
    length = D_SIZE*D_CHANNELS;

    /* there is enough room in the buffer
     * before adding data with a blocking function.*/
    if( sock_tbleft(&tsock) < length) {
       sock_close(&tsock);
       while( openTcpSock()<=0 );
    }
    retval = sock_awrite(&tsock, buf, length);
    //printf("sendout:%d\n",retval);

    switch (retval) {
      case 0:
       printf("Insufficient free space in transmit buffer!\n");
       break;
      case 1:
       printf("Len is greater than the total socket receive buffer
size!\n");
       break;
      case 2:
       printf("The socket has been closed for further
transmission!\n");
       while( openTcpSock()<=0 );
       break;
      case 3:
       printf("len<0 or parameter wa invalid!\n");
       break;
    }
    sock_flush(&tsock);
    return 1;
}

/***********************************
 * int openTcpSock()
 *
 * open a tcp socket
 * return 1 successfully
 * return -1 unable to open TCP session
 ***********************************/
```

```
int openTcpSock(){
     sock_close(&tsock);
     printf("try to open socket ... \n");
     if ( !tcp_open( &tsock, 0, resolve(REMOTE_IP), REMOTE_PORT ,
NULL )) {
       puts("Unable to open TCP session!\n");
       return -1;
     }
     else {
       while(!sock_established(&tsock)) {
         if (!tcp_tick(&tsock)) {
           printf("Failed to establish\n");
           break;
         }
       }
     }
     if (sock_established(&tsock)) {
       printf("Established OK!\n");
       // do whatever needs to be done...
     }
     return 1;
}

void setXDCP() {
    int i;
    WrPortI(PGDR, &PGDRShadow, 0x00);        //set PG0-PG7=0
    BitWrPortI(PFDR, &PFDRShadow, 1, 1);     //set PF1=1, set all CS
of left side XDCP to "0"
    BitWrPortI(PFDR, &PFDRShadow, 1, 2);     //set PF2=1, set all CS
of right side XDCP to "0"
    BitWrPortI(PFDR, &PFDRShadow, 0, 1);     //set PF1=0
    BitWrPortI(PFDR, &PFDRShadow, 0, 2);     //set PF2=0
    BitWrPortI(PCDR, &PCDRShadow, 0, 0);     //set PC0=0, U/D(TXD)
of XDCP to count up
    for(i=0;i<101;i++) {

        BitWrPortI(PFDR, &PFDRShadow, 0, 0);     //set INC(CLKD) of
XDCP to 0
        BitWrPortI(PFDR, &PFDRShadow, 1, 0);     //set INC(CLKD) of
XDCP to 1
    }                                              //set XDCP to top
    BitWrPortI(PCDR, &PCDRShadow, 1, 0);     //set PC0=1, U/D(CLKD)
of XDCP to count down
    for(i=0;i<50;i++) {
        BitWrPortI(PFDR, &PFDRShadow, 0, 0);     //set INC(CLKD) of
XDCP to 0
        BitWrPortI(PFDR, &PFDRShadow, 1, 0);     //set INC(CLKD) of
XDCP to 1
    }                                            //set XDCP to midway,
Vw=2.5V
    BitWrPortI(PCDR, &PCDRShadow, 0, 0);     //set PC0=0, U/D(TXD)
of XDCP to count up
    WrPortI(PGDR, &PGDRShadow, 0xff);        //set PG0-PG7=1
    BitWrPortI(PFDR, &PFDRShadow, 1, 1);     //set PF1=1, set all CS
of left side XDCP to "1"
    BitWrPortI(PFDR, &PFDRShadow, 1, 2);     //set PF2=1, set all CS
of right side XDCP to "1"
```

```
    BitWrPortI(PFDR, &PFDRShadow, 0, 1);     //set PF1=0
    BitWrPortI(PFDR, &PFDRShadow, 0, 2);     //set PF2=0
}

/***********************************************
 *     void calibration()
 *
 *     calibrate all the XDCP to optimal position
 *
 ***********************************************/

 void calibration() {
     char Step_counter;                       // step counter for
XDCP
     char leftXDCP_max[8], leftXDCP_min[8];   // store maximum and
minimum step count for left XDCP
     char rightXDCP_max[8], rightXDCP_min[8]; // store maximum and
minimum step count for right XDCP
     char leftXDCPset[8], rightXDCPset[8];    // save the setting of
XDCP
     char samplingData_left[10], samplingData_right[10];    // store
the sample data of 10 times
     char leftSetDone, rightSetDone;          // store the symbol of
the channel which has been set
     char temp, maxLow_left1, maxLow_left2, maxLow_right1,
maxLow_right2;
     char       minHigh_left1, minHigh_left2, minHigh_right1,
minHigh_right2;
     int i, j, symbol;

     memset(samplingData_left, 0x00, 10);
     memset(samplingData_right, 0x00, 10);

     BitWrPortI(PCDR, &PCDRShadow, 1, 2);     // set PC2=1, turn on
the LED indicator

     WrPortI(PGDR, &PGDRShadow, 0x00);        //set PG0-PG7=0
     BitWrPortI(PFDR, &PFDRShadow, 1, 1);     //set PF1=1, set 1-8
CS of left side XDCP to "0"
     BitWrPortI(PFDR, &PFDRShadow, 1, 2);     //set PF2=1, set 1-8
CS of right side XDCP to "0"
     BitWrPortI(PFDR, &PFDRShadow, 0, 1);     //set PF1=0
     BitWrPortI(PFDR, &PFDRShadow, 0, 2);     //set PF2=0

     BitWrPortI(PFDR, &PFDRShadow, 1, 0);     //set INC(CLKD) of
XDCP to 1
     BitWrPortI(PCDR, &PCDRShadow, 0, 0);     //set PC0=0, U/D(TXD)
of XDCP to "1",  count up
     for(i=0;i<101;i++) {

         BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of
XDCP to 0
         BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC of XDCP to
"1", genarate a negtive pulse on INC line
     }                                         //set XDCP to top
     BitWrPortI(PCDR, &PCDRShadow, 1, 0);     //set PC0=1, U/D(TXD)
of XDCP to "0",  count down
```

```
     for(i=0;i<10;i++) {
        BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC of XDCP to
0
        BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC of XDCP to
1, genarate a negtive pulse on INC line
     }                                        //set XDCP to step 90

     Step_counter=90;

     leftSetDone=0;
     rightSetDone=0;
     intflag=0;

     while(Step_counter>10) {
       for(i=0; i<10; i++) {
         for( ; ; ) {
          if(intflag==1) {
              intflag=0;
              break;
          }
         }
          samplingData_left[i]=testdata_left;     // store the
sampling data of left side
          samplingData_right[i]=testdata_right;   // store the
sampling data of right side
       }
       for(j=0; j<8; j++) {
        if(bit(&leftSetDone,j))
           continue;
        symbol=0;
        for(i=0; i<10; i++) {
          if(bit(&samplingData_left[i],j)==0) {
            symbol=1;
            break;
         }
       }
       if(symbol==0) {
        leftXDCP_max[j]=Step_counter;
        temp=0x01;
        temp=temp<<j;
        leftSetDone=leftSetDone|temp;
       }
     }                                          // set 8 maximum
steps of left side
     for(j=0; j<8; j++) {
       if(bit(&rightSetDone,j))
        continue;
        symbol=0;
        for(i=0; i<10; i++) {
           if(bit(&samplingData_right[i],j)==0) {
             symbol=1;
             break;
           }
        }
        if(symbol==0) {
           rightXDCP_max[j]=Step_counter;
```

```
           temp=0x01;
           temp=temp<<j;
           rightSetDone=rightSetDone|temp;
         }
        }              // set 8 maximum steps of right side

        if((leftSetDone==0xff)&&(rightSetDone==0xff))
          break;
        else {
         BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of
XDCP to 0
         BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC(CLKD) of
XDCP to 1, genarate a negtive pulse on INC line
         Step_counter--;
         intflag=0;
        }
      }


      // BitWrPortI(PBDR, &PBDRShadow, 0, 6);     // set PB6=0, turn
off left laser
      // BitWrPortI(PBDR, &PBDRShadow, 0, 7);     // set PB7=0, turn
off right laser

    symbol=0;

    while(1) {
      if((BitRdPortI(PCDR, 1)==0)&&(symbol==0))
         symbol=1;
      else if((BitRdPortI(PCDR, 1)==1)&&(symbol==1))
         break;

      costate{
         BitWrPortI(PCDR, &PCDRShadow, 1, 2);     // set PC2=1, turn
on the LED indicator
        waitfor( DelayMs(500L) );                // Turn on LED for
500ms
        BitWrPortI(PCDR, &PCDRShadow, 0, 2);     // set PC2=0, turn
off the LED indicator
        waitfor( DelayMs(500L) );                // Turn off LED for
500ms
      }
    }

    BitWrPortI(PFDR, &PFDRShadow, 1, 0);     //set INC(CLKD) of XDCP
to 1
    BitWrPortI(PCDR, &PCDRShadow, 1, 0);     //set PC0=1, U/D(TXD)
of XDCP to "0",  count down
    for(i=0;i<100;i++) {
      BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of XDCP
to 0
      BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC(CLKD) of XDCP
to 1, genarate a negtive pulse on INC line
    }                                       //set XDCP to bottom
    BitWrPortI(PCDR, &PCDRShadow, 0, 0);     //set PC0=0, U/D(TXD)
of XDCP to "1", count up
```

```
    for(i=0;i<10;i++) {
      BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of XDCP
to 0
      BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC(CLKD) of XDCP
to 1, genarate a negtive pulse on INC line
    }                                       //set XDCP to 0.5V

    leftSetDone=0;
    rightSetDone=0;
    memset(samplingData_left, 0xff, 10);
    memset(samplingData_right, 0xff, 10);
    intflag=0;
    Step_counter=10;

    while(Step_counter<90) {
        for(i=0;i<10;i++) {
            for( ; ; ) {
               if(intflag==1) {
                   intflag=0;
                   break;
               }
            }
            samplingData_left[i]=testdata_left;     // store the
sampling data of left side
            samplingData_right[i]=testdata_right;   // store the
sampling data of right side
        }
       for(j=0; j<8;j++) {
         if(bit(&leftSetDone,j))
           continue;
         symbol=0;

         for(i=0; i<10; i++) {
           if(bit(&samplingData_left[i],j)==1) {
             symbol=1;
             break;
           }
         }
         if(symbol==0) {
            leftXDCP_min[j]=Step_counter;
            temp=0x01;
            temp=temp<<j;
            leftSetDone=leftSetDone|temp;
         }
       }          // set 8 minimum steps of left side
       for(j=0; j<8;j++) {
        if(bit(&rightSetDone,j))
            continue;
        symbol=0;
        for(i=0; i<10; i++) {
          if(bit(&samplingData_right[i],j)==1) {
            symbol=1;
            break;
          }

        }
        if(symbol==0) {
```

```
         rightXDCP_min[j]=Step_counter;
         temp=0x01;
         temp=temp<<j;
         rightSetDone=rightSetDone|temp;
       }
      }      // set 8 minimum steps of right side

      if((leftSetDone==0xff)&&(rightSetDone==0xff))
        break;
      else {
        BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of
XDCP to 0
        BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC(CLKD) of
XDCP to 1, genarate a negtive pulse on INC line
        Step_counter++;
      }
    }

    for(i=0; i<8; i++) {
       if((leftXDCP_max[i]<=leftXDCP_min[i])||
                  (rightXDCP_max[i]<=rightXDCP_min[i]))
         errorIndicate();    // detect if there is any error
    }


    BitWrPortI(PFDR, &PFDRShadow, 1, 0);     //set INC(CLKD) of XDCP
to 1
    BitWrPortI(PCDR, &PCDRShadow, 1, 0);     //set PC0=1, U/D(TXD)
of XDCP to "0", count down
    for(i=0;i<100;i++) {
       BitWrPortI(PFDR, &PFDRShadow, 0, 0);   //set INC(CLKD) of
XDCP to 0
       BitWrPortI(PFDR, &PFDRShadow, 1, 0);   //set INC(CLKD) of
XDCP to 1, genarate a negtive pulse on INC line
    }                                       //set XDCP to bottom
    BitWrPortI(PCDR, &PCDRShadow, 0, 0);     //set PC0=0, U/D(TXD)
of XDCP to "1", count up

    WrPortI(PGDR, &PGDRShadow, 0xff);        //set PG0-PG7=1
    BitWrPortI(PFDR, &PFDRShadow, 1, 1);     //set PF1=1, set 1-8 CS
of left side XDCP to "1"
    BitWrPortI(PFDR, &PFDRShadow, 1, 2);     //set PF2=1, set 1-8 CS
of right side XDCP to "1"
    BitWrPortI(PFDR, &PFDRShadow, 0, 1);     //set PF1=0
    BitWrPortI(PFDR, &PFDRShadow, 0, 2);     //set PF2=0, set 16 CS
of XDCP to "1"


    temp=0x01;
    for(i=0; i<8; i++) {
       WrPortI(PGDR, &PGDRShadow, ~temp);              //set PGi=0,
i=0-7, select left XDCPi
       BitWrPortI(PFDR, &PFDRShadow, 1, 1);            //set PF1=1,
set CS of left side XDCPi to "0", other CS to "1"
       BitWrPortI(PFDR, &PFDRShadow, 0, 1);
       for(j=0; j<(( leftXDCP_max[i]+leftXDCP_min[i] )/2); j++) {
```

```
         BitWrPortI(PFDR, &PFDRShadow, 0, 0);        //set INC(CLKD)
of XDCP to 0
         BitWrPortI(PFDR, &PFDRShadow, 1, 0);        //set INC(CLKD)
of XDCP to 1, genarate a negtive pulse on INC line
       }                                              //set left
XDCPi to the position

       WrPortI(PGDR, &PGDRShadow, 0xff);               //set PG0-
PG7=1
       BitWrPortI(PFDR, &PFDRShadow, 1, 1);            //set PF1=1
       BitWrPortI(PFDR, &PFDRShadow, 0, 1);            //set PF1=0,
set 1-8 CS of left side XDCP to "1"

       temp<<=1;
    }

    temp=0x01;
    for(i=0; i<8; i++) {
       WrPortI(PGDR, &PGDRShadow, ~temp);  //set PGi=0, i=0-7,
select right XDCPi
       BitWrPortI(PFDR, &PFDRShadow, 1, 2);            //set PF2=1,
set CS of right side XDCPi to "0", other CS to "1"
       BitWrPortI(PFDR, &PFDRShadow, 0, 2);
       for(j=0; j<(( rightXDCP_max[i]+rightXDCP_min[i] )/2); j++) {
          BitWrPortI(PFDR, &PFDRShadow, 0, 0);        //set
INC(CLKD) of XDCP to 0
          BitWrPortI(PFDR, &PFDRShadow, 1, 0);        //set
INC(CLKD) of XDCP to 1, genarate a negtive pulse on INC line
       }                                              //set left
XDCPi to the position

       WrPortI(PGDR, &PGDRShadow, 0xff);    //set PG0-PG7=1
       BitWrPortI(PFDR, &PFDRShadow, 1, 2); //set PF2=1
       BitWrPortI(PFDR, &PFDRShadow, 0, 2); //set PF2=0, set 1-8
 CS of right side XDCP to "1"


       temp<<=1;
    }


    WrPortI(PGDR, &PGDRShadow, 0xff);        //set PG0-PG7=1
    BitWrPortI(PFDR, &PFDRShadow, 1, 1);     //set PF1=1, set 1-8 CS
of left side XDCP to "1"
    BitWrPortI(PFDR, &PFDRShadow, 1, 2);     //set PF2=1, set 1-8 CS
of right side XDCP to "1"
    BitWrPortI(PFDR, &PFDRShadow, 0, 1);     //set PF1=0
    BitWrPortI(PFDR, &PFDRShadow, 0, 2);     //set PF2=0

    BitWrPortI(PBDR, &PBDRShadow, 1, 6);     // set PB6=1, turn on
left laser
    BitWrPortI(PBDR, &PBDRShadow, 1, 7);     // set PB7=1, turn on
right laser

    BitWrPortI(PCDR, &PCDRShadow, 0, 2);     // set PC2=0, turn off
the LED indicator
```

```
}


void errorIndicate() {
    int i, syb;
    syb=0;
    for(i=0; i<100; i++) {
      if((BitRdPortI(PCDR, 1)==0)&&(syb==0))
        syb=1;
      else if((BitRdPortI(PCDR, 1)==1)&&(syb==1))
        break;

      costate{
        BitWrPortI(PCDR, &PCDRShadow, 1, 2);     // set PC2=1, turn
on the LED indicator
        waitfor( DelayMs(200L) );                // Turn on LED for
200ms
        BitWrPortI(PCDR, &PCDRShadow, 0, 2);     // set PC2=0, turn
off the LED indicator
        waitfor( DelayMs(200L) );                // Turn off LED for
200ms
      }
    }

}
```

Source Code of TCP/IP communication in Linux

File name: server.c

```
#include <sys/socket.h>       /*  socket definitions        */
#include <sys/types.h>        /*  socket types              */
#include <arpa/inet.h>        /*  inet (3) funtions         */
#include <sys/stat.h>

#include <unistd.h>           /*  misc. UNIX functions      */
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <time.h>
#include <signal.h>
#include <fcntl.h>

#include "control.h"

/*  Global constants  */
#define ECHO_PORT          (2002)
#define MAX_LINE           (1000)
#define LISTENQ            (1024)   /*  Backlog for listen()   */
#define N_POINTS           (300)

struct msg_struct msg;

void timeout( int sig )
{
   printf("\nTimeout\n");
```

```
   exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
    int       list_s;           /*  listening socket          */
    int       conn_s;           /*  connection socket         */
    short int port;             /*  port number               */
    struct    sockaddr_in servaddr; /*  socket address structure  */
    unsigned char     buffer[N_POINTS*2];
    char     *endptr;           /*  for strtol()              */
    time_t   time1;
    struct timeval tv_start, tv_end;
    int cc;
    char buf;
    struct sigaction sa;
    int fd0, fd1, fd2, fd3;
    unsigned char ret = 0;
    //short int ret = 0;

    /*  Get port number from the command line, and
        set to default port if no arguments were supplied  */
    if ( argc == 2 ) {
       port = strtol(argv[1], &endptr, 0);
       if ( *endptr ) {
          fprintf(stderr, "ECHOSERV: Invalid port number.\n");
          exit(EXIT_FAILURE);
       }
    }
    else if ( argc < 2 ) {
       port = ECHO_PORT;
    }
    else {
       fprintf(stderr, "ECHOSERV: Invalid arguments.\n");
       exit(EXIT_FAILURE);
    }


    /*  Create the listening socket  */
    if ( (list_s = socket(AF_INET, SOCK_STREAM, 0)) < 0 ) {
       fprintf(stderr, "ECHOSERV: Error creating listening
socket.\n");
       exit(EXIT_FAILURE);
    }


    /*  Set all bytes in socket address structure to
        zero, and fill in the relevant data members   */
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family      = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port        = htons(port);


    /*  Bind our socket addresss to the
    listening socket, and call listen()  */
    if ( bind(list_s, (struct sockaddr *) &servaddr,
sizeof(servaddr)) < 0 ) {
```

```
   fprintf(stderr, "ECHOSERV: Error calling bind()\n");
   exit(EXIT_FAILURE);
}

/*
sa.sa_handler = timeout;
sa.sa_flags = 0;
sigemptyset(&sa.sa_mask);
sigaction(SIGALRM, &sa, NULL);
alarm(300);  // Two minutes
*/

if ( listen(list_s, LISTENQ) < 0 ) {
  fprintf(stderr, "ECHOSERV: Error calling listen()\n");
  exit(EXIT_FAILURE);
}

/* Make a fifo for inter-process communications */
mkfifo("rtf0", 0777);
fd0 = open("rtf0", O_WRONLY, 0);
printf("done with rtf0\n");

mkfifo("rtf1", 0777);
fd1 = open("rtf1", O_WRONLY, 0);
printf("done with rtf1\n");

mkfifo("rtf2", 0777);
fd2 = open("rtf2", O_RDONLY, 0);
printf("done with rtf2\n");

mkfifo("rtf3", 0777);
fd3 = open("rtf3", O_WRONLY, 0);
printf("done with rtf3\n");

/*  Enter an infinite loop to respond
    to client requests and echo input  */
while ( 1 )
{
   printf("\n\n adafsedfasdfasdf\n\n ");
   read(fd2, &msg, sizeof(msg));
   if(msg.command == START_TASK)
   {
       printf("read START_TASK\n");
       break;
   }
}

int n = 0;
while ( 1 ) {


  /*  Wait for a connection, then accept() it  */
  if ( (conn_s = accept(list_s, NULL, NULL) ) < 0 ) {
      fprintf(stderr, "ECHOSERV: Error calling accept()\n");
      exit(EXIT_FAILURE);
  }
  printf("\nSocket opened\n");
```

```
      /*  Retrieve an input line from the connected socket
          then simply write it back to the same socket.     */
      int i = 0;
      gettimeofday(&tv_start, NULL);
      while( cc = read(conn_s, buffer, N_POINTS*2)){
          printf("read :%d\n", cc);
          //printf("%s\n", buffer);
          if( cc < 0 )
             break;
          gettimeofday(&tv_end, NULL);
          //printf("%s, %d\n", buffer, i);
          if(ret == 0)
          {
             if( (n = write(fd0, buffer, N_POINTS*2)) < 0)
             {
               fprintf(stderr, "WRITE: unable to write to fifo
0\n");
             }
             printf("r=%d\n", ret);
          }
          if(ret == 1)
          {
             if( (n = write(fd1, buffer, sizeof(buffer))) < 0)
             {
                fprintf(stderr, "WRITE: unable to write to fifo
1\n");
             }
             printf("r=%d\n", ret);
          }
          printf("write fifo=%d\n", n);

          if( (write(fd3, &ret, 1)) < 0)
          {
             fprintf(stderr, "WRITE: unable to write to fifo 3\n");
          }
          /*
          printf("   input strlen: %d", strlen(buffer));
          printf("   time: %d\n", abs(-
tv_start.tv_usec+tv_end.tv_usec));
              */
          gettimeofday(&tv_start, NULL);
          memset(buffer,0,sizeof(buffer));
          i++;
          if(ret == 0)
             ret = 1;
          else if(ret == 1)
             ret = 0;
      }


      /*  Close the connected socket  */

      if ( close(conn_s) < 0 ) {
         fprintf(stderr, "ECHOSERV: Error calling close()\n");
         exit(EXIT_FAILURE);
      }
```

```
      printf("\nSocket closed\n");

    }

    unlink("rtf0");
    unlink("rtf1");
    unlink("rtf2");
    unlink("rtf3");
}
```