# Optical Speedometer

Group 19

Zach Scheffer

Technical Report Submitted in Partial Fulfillment of the
Engineering Technology Senior Design Project Course
ETG 4950C

Dr. Al Ducharme

April 25, 2007

Table of Contents



Figure Index



Appendices

## I. Abstract

The goal of this project was to create a speedometer that would use an optical sensor from a standard consumer computer mouse, as the input method, that would be able to read speeds up to 20 miles per hour. The optical sensor used is an ADNS-3080. This system was designed to work on an electric skateboard. A small lens was also used to create a sharp clear image of the surface of the road on the optical sensor to insure accurate data input. The use of a peripheral interface controller, or PIC, would allow communication to and from the optical sensor, as well as a way to calculate the data from counts per inch into miles per hour. The PIC would also control a LCD screen for output results. The device would display a compiled running average of the speed and a maximum speed reached.

## II. Background

Optical mice were first developed in the 1980's. An optoelectronic sensor is used to take sequential image captures of the surface below the device and then compare the images to recognize movement. The sensor looks at any differences in the image, or features and then calculates how far it has traveled based on how far the features have moved since the last image. This also depends on the resolution of the images, the shutter speed, and the frame rate. The sensor can take image captures up to around 6000 frames per second [2].

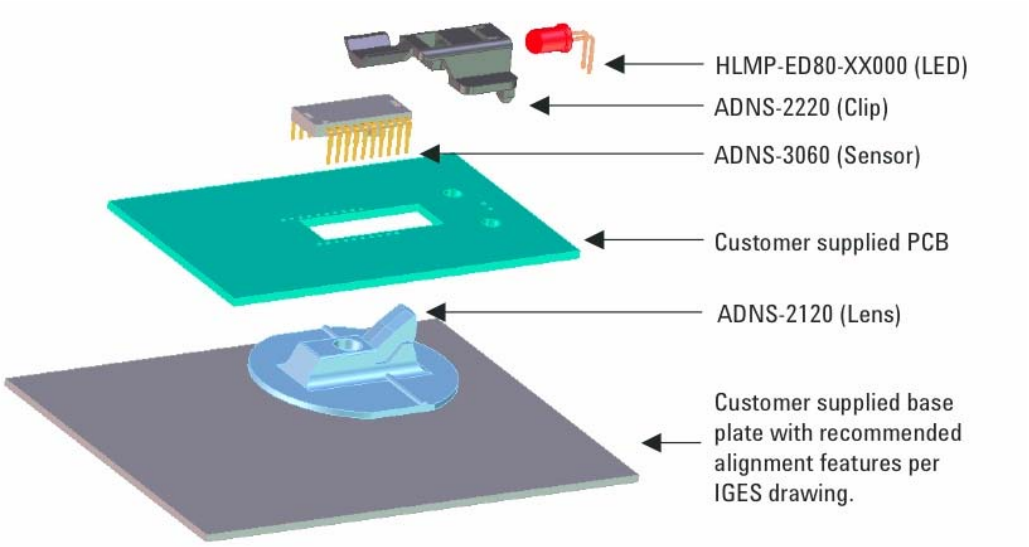



**Figure 1- [3] setup for optical mouse unit**

The sensor can detect lighter and darker areas and can determine how far the mouse has moved by how far the distinctive patterns of the sequential images have moved. Light emitting diodes (LEDs) are used to illuminate the surface to provide a good image. Small lasers are also being used and provide an even more accurate image than the LED. A small lens is used to create a focused image on the sensor. A lens is also used to spread the led light out over a uniform area. In the beginning of this project, it was

determined that a LED optical mouse could provide the quality needed for this project and at less of a cost. The optical sensor communicates with a USB interface chip and sends data describing the motion to the computer [1][3]. For my use, the optical sensor will instead send the data to a PIC and it will compile the speed traveled by the skateboard.

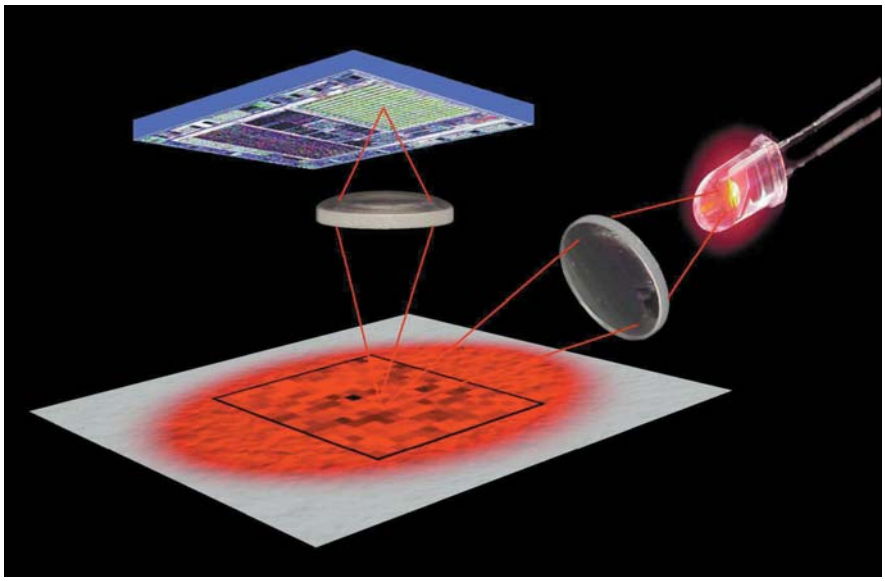


**Figure 2- [1] Diagram of optical mouse**

Although there are several documented projects interfacing with an optical mouse sensor, there is no evidence that anyone has ever tried to use a sensor to determine the speed of a transportation device, such as a skateboard. Most ventures into interfacing with an optical mouse have been done to use the sensor as an image capture device much like a camera, only concerned with the images [4]. The goal of this device is to compute the distance traveled per time to give an accurate value of how fast the skateboard is moving. Furthermore, currently there are no means to find out how fast you are traveling on a skateboard. With the electric skateboard, speeds of up to 20 mph are reached. The

goal is to be able to document and record these speeds. This is accomplished by comparing count values of the delta X register sequentially. The delta X register holds data describing how far the features of the image have moved from the last capture of data. There is also a Delta Y register, but I am only concerning with one direction of the Delta X register since it will only be computing speed when the skateboard is traveling forward. Taking this value and multiplying it by a conversion factor I determined, will give miles per hour. The ADNS-3080 is the chip that will be used in this device. According to Avago, it can read up to 40 inches per second. 20 miles per hour equates to roughly 352 inches per second. It was then determined that at least a 10:1 magnification would be need to recorded these speeds.

**Some examples of LED-based optical mouse sensors:**

| | | Corded | | | | |
|---|---|---|---|---|---|---|
| **Sensor** | | **ADNS-2610** | **ADNS-2620** | **ADNS-2051** | **ADNS-3060** | **ADNS-3080** |
| **General Features** | **Units** | **Entry Level** | **Entry Level** | **Mid Range** | **High Performance** | **High Sensitivity** |
| Power Consumption - Running | mA | 15 (typ) | 15 (typ) | 15 (typ) | 40(max) | 40(max) |
| Max Speed | ips | 12 (@1500fps) | 12 (@1500fps) | 14 (@1500fps) | 40 (@6400fps) | 40 (@6400fps) |
| Frame Rate | fps | 1500 | 500-2300 | 500-2300 | 500-6469 | 500-6469 |
| Resolution | cpi | 400 | 400 | 400/800 | 400/800 | 400/1600 |
| Acceleration form Sleep Mode | g | 0.25g @ 1500fps | 0.25g @ 1500fps | 0.15g @ 1500fps | 15g @ 6469fps | 15g @ 6469fps |

**Figure 3- [3] attributes for several optical sensors**

Optical mice are relatively inexpensive and if it is possible to use the sensors for alternate applications, they could provide a good starting point for many useful and innovative products.

Traditional speedometers are mechanical systems that use at least one rotating piece to record movement. A metal cable is attached to the transmission in a car or to the axel in other moving devices and turns as the car travels. This in turn rotates a gauge using magnetic fields. This allows it to move while accelerating, but once a steady speed is met, the gauge stays still. A major advantage of the optical speedometer is the absence of moveable parts. Moveable parts in a system tend to degrade over time and need to be fixed or replaced. With this system, as long as the lens is in good condition, it could last for a long time. The inclusive system is another advantage of the optical speedometer. The entire system is one unit that can be put on any moving device. This makes it applicable to a wide variety of uses. The only change needed for the system to work on a different moving device would be the focal length and the position of the lens relative to the ground surface. A different mounting piece for different size lenses could easily be fabricated. Using the Thin Lens Equation, $1/q = 1/f + 1/p$, would provide the right focal length and distance from the sensor in order to get accurate clear image results [8].

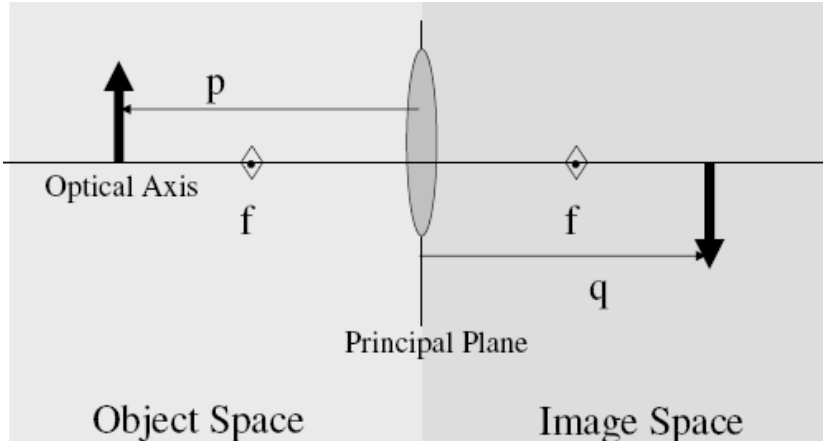



**Figure 4- thin lens equation [8]**

**III. Current Literature Review**

[3] Understanding Optical Mice, by Teo Chiang Mei is a white paper reference sheet for Avago Technologies. They are the leading developer of optical mouse sensors. This paper describes the basic functions of an optical mouse. It starts off with some background history on computer mice and then explains how an optical mouse differs from the standard mechanical mouse. It also gives information on the advantages of a laser based optical mouse compared to an LED based mouse. It also has a good table that provides some comparable attributes of various mouse sensors. This table was used as a starting point for myself in determining what quality of mouse sensor I needed to record speeds up to 20 mph.

[4] Optical Mouse Cam is a posted project on the website Sprites Mods. It describes how an optical mouse sensor, the ADNS-2610 in this case, can be used to create a hand scanner. The site provides some downloadable software to allow the user to communicate with the mouse and to view the actual images that the sensor is capturing. It also explains what connections you need to change in order for it to work. This document has been referenced in almost all other occurrences found online of optical mouse modifications.

[5] Humanoids 16-264- Vision is a webpage posted by Carnegie Mellon University students Tristan Trutna and Mark Schumacher. Their goal is to prove that by using a different lens with a greater focal length, the optical mouse sensor can be used to capture accurate images at great distances compared to the traditional use of an optical

mouse. They used an ADNS-3060 and a Parallax Basic Stamp PIC for their project. The page shows their GUI for the software they coded and some results from the experiment. This showed me that by changing the lens I could achieve similar successful results.

## IV Experimental Method

When building the schematic and circuit board layout for this project, I used a sample schematic for USB, PS/2 mouse application and a schematic from the class CET 4134c, taken at UCF in the summer of 2006. I referenced this schematic because I am using the same LCD screen and same PIC in this current project.

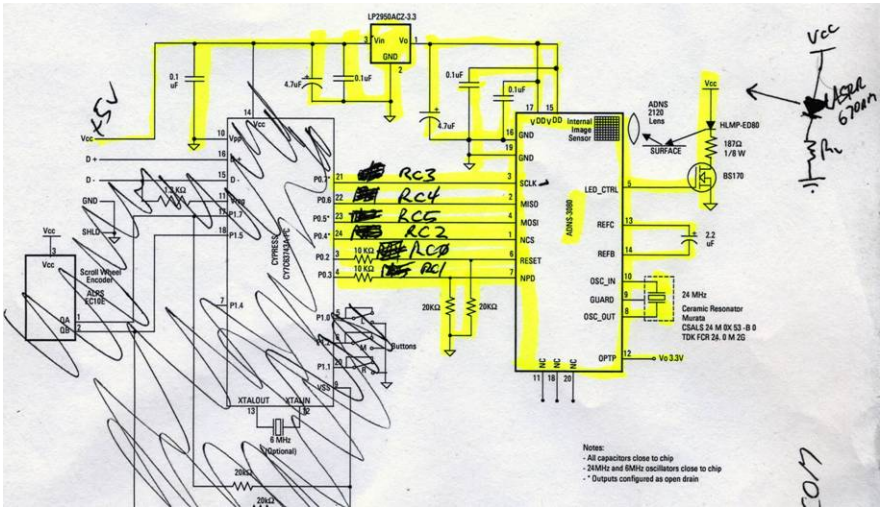



**Figure 5- suggested diagram of ADNS3080 communication [2]**

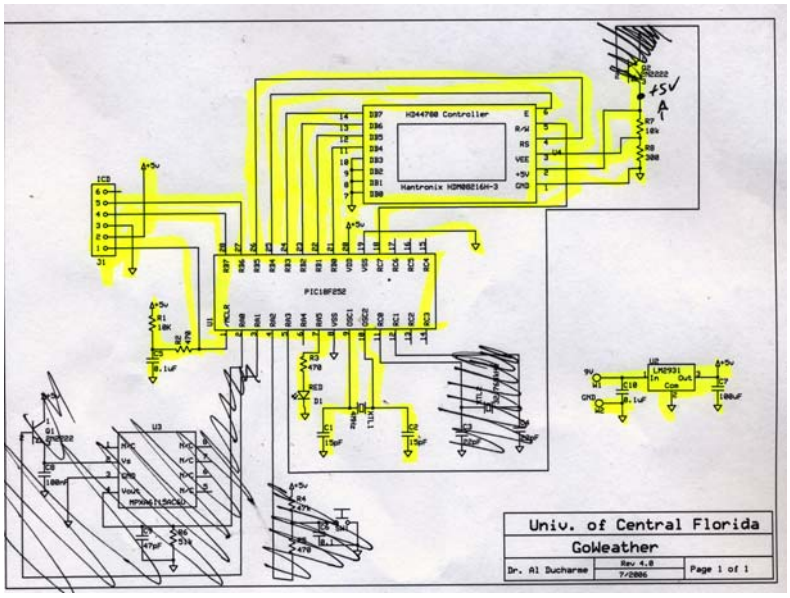



**Figure 6- Go weather diagram [10]**

I combined the two to create a workable schematic, referenced in appendix A. I used ExpressSCH to create the schematic and to layout the physical components, I used ExpressPCB. I had to create several custom components, including the ADNS-3080, Hantronix LCD, and the ICD. After assembling the components on the board and testing the device, I noticed that the voltage converter was not correctly converting the 9volt power source down to the required 5 volts. After swapping the voltage regulator and removing the 300 ohm resistor, the device was operating under safe conditions.

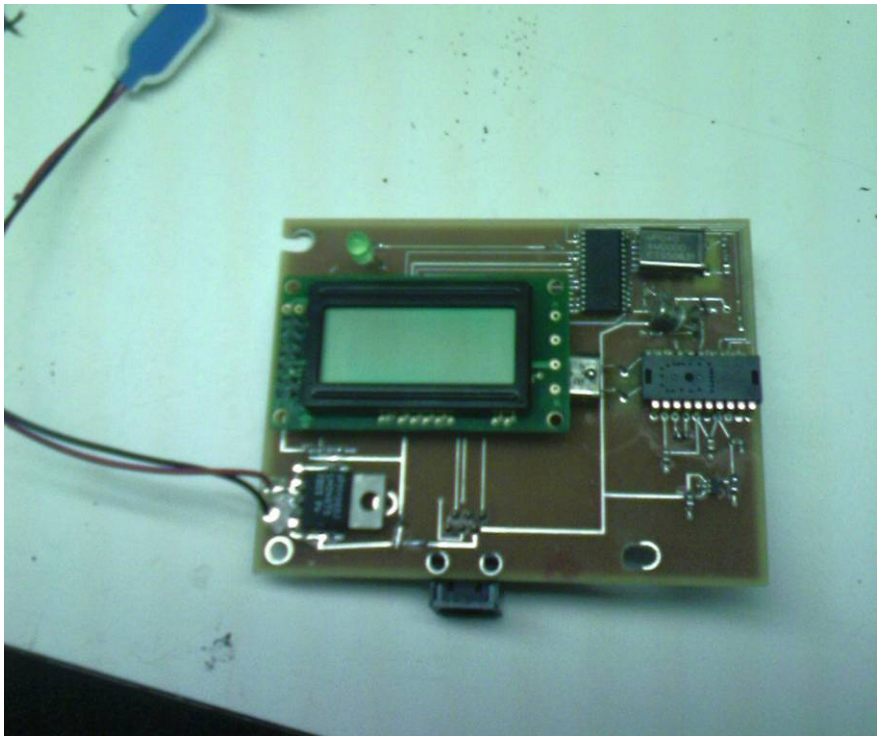


**Figure 7- Skate Speed circuit board**

The next task was to create the code to allow the ADNS-3080 to communicate with the PIC. Using the data sheets for both the ADNS-3080 and the PIC, I was able to establish communication. With the devices communicating, next I looked at the SQUAL and Motion registers of the ADNS-3080. SQUAL is a measure of surface quality. This is

used to determine if the sensor is focused at the right distance to achieve quality data. The device was designed to sit 6.1 inches from the readable surface (road).

After determining the right focal distance of the lens for the correct distance from the ground, the next task was to determine the relationship of counts in the DeltaX register to actual distance traveled by the device. Using the information from the data sheet, it was determined that the relationship of mouse counts to distance is equal to mouse counts divided by counts per inch. Therefore, the distance is also dependent on the dpi value set at the time of operation. The ADNS-3080 can operate at 1600 and 400 dpi. It was determined that the device would operate at 1600 dpi. The distance traveled is also dependent on how fast the shutter speed is operating and the frame rate. It was concluded that it takes 2ms to capture data from the sensor. All of these factors plus the conversion from inches per second to miles per hour would be used to find the conversion factor needed. Once the device was accurately recognizing features in the images and correctly calculating speed, the final tasks would be to calculate a running average, store the max value, and display these two results onto the lcd. Several different tests would be used also in order to verify that the device is working properly. I tested the device on various surfaces, indoor and outdoor. I also speed tested it by using a car with a GPS speed monitoring device in it. A testing sheet was also created to perform preliminary “in the workshop” testing. This pattern was developed by Dr. Ducharme prior to this project to test various optical devices.

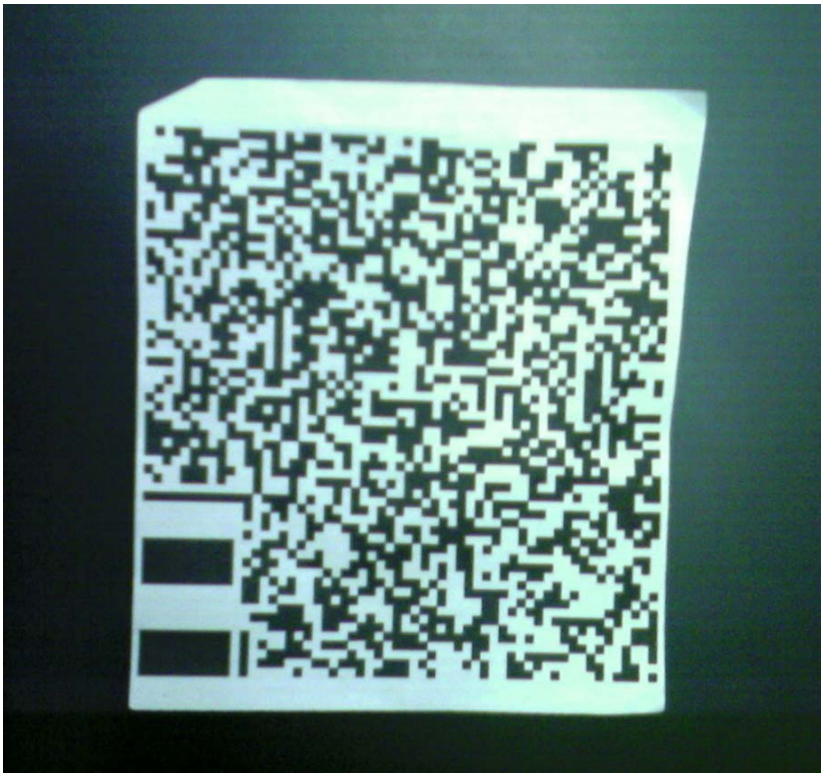


**Figure 8- testing grid**

**Optical Speedometer Flow Chart**

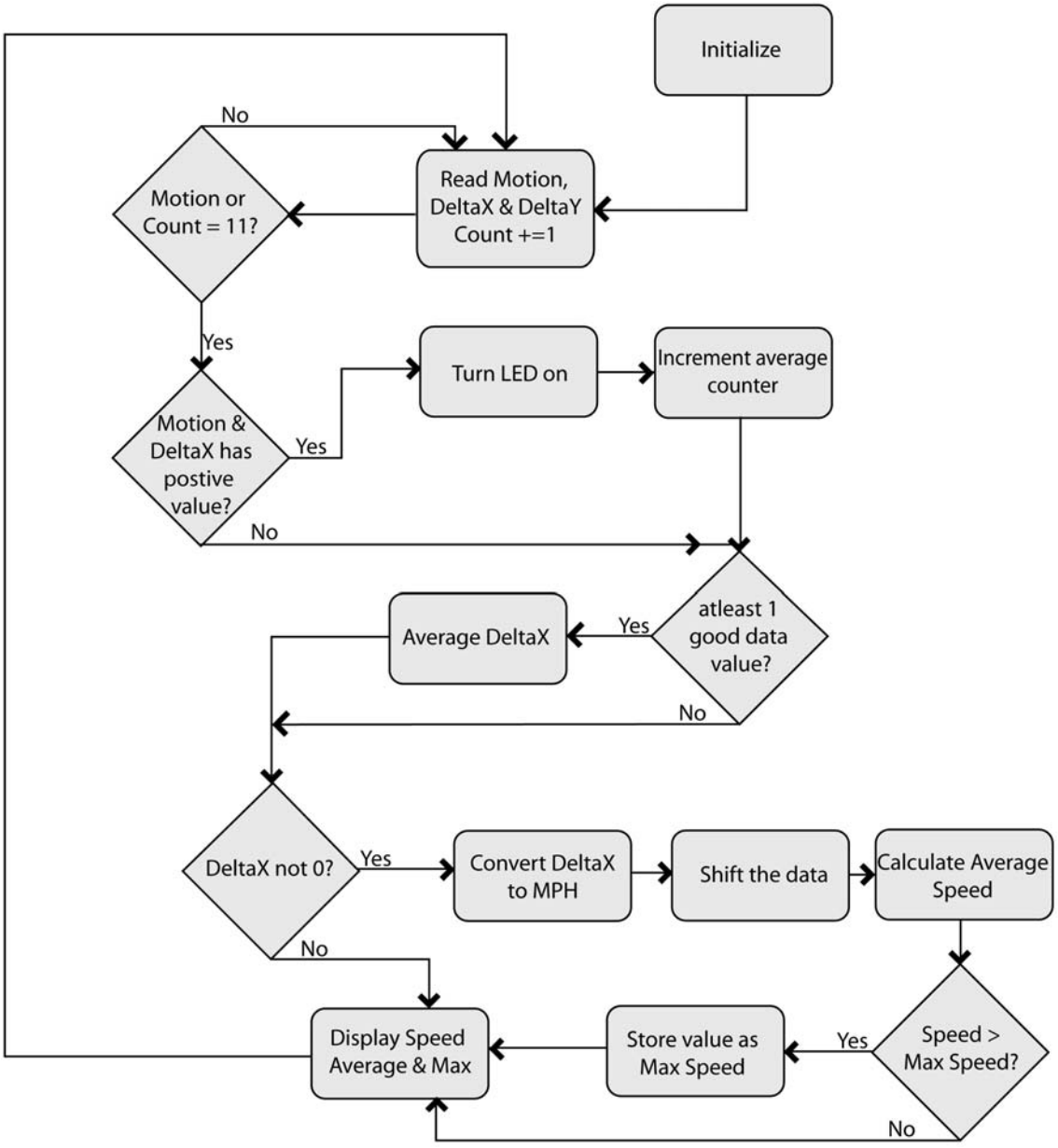



**Figure 9- Optical Speedometer Flow Chart**

(1) This represents all initialization that has to be done in order for this code to execute. This includes defining my two functions, Initialize_ADNS3080() and GetData_ADNS3080(), including the necessary header files to allow for lcd display,

delay counts, and reading and writing from the ports. This also includes defining all global variables to be used. For a complete list, refer to Appendix C. The LCD also has to be initialized and then displays the welcome message. The led is turned on and values are set to zero. Another important thing accomplished in this is to set the dpi to 1600.

(2) The program enters a while loop that runs until the power is cut to the device. The communication is turned on to the ADNS-3080 and looks for motion. It reads the motion register, deltaX register, deltaY register, and the overflow register.

(3) It looks for motion for 10 cycles. If no motion is read, the display updates and then it goes back to looking for Motion.

(4) This checks to see if the deltaX value is in the right direction. If it isn't, it changes the value to 0. If deltaX is a positive value and the motion flag is on then the led is turned on and the value is added to a running total. The number of values in total is also recorded.

(5) Led turns on to show the user that a good value has been read by the sensor.

(6) The value of deltaX is added to a running total. The number of values in total is also recorded.

(7) It checks to see if at least one value has been put into the running total, and if there has been then DeltaX is averaged. If not, it returns a zero value for deltaX, making the

program look for motion again. This allows zero values of motion to bring the average back down to 0, if you are decelerating, but prevents a zero to start the average calculating.

(8) The program takes the running total of values and divides by how many values have been added to the running total to get the average.

(9) Up until now, the program has been in the GetData_ADNS3080() function. Now the deltaX value is checked to see if it is a value other than 0. If it is it will then be converted to miles per hour and checked to see if it is a max value.

(10) The deltaX value is multiplied by 10 and then divided by 5. This converts counts to MPH. It is multiplied by ten now since I did not include the header files for divided decimals, but when the values are displayed on the LCD, a decimal point will be entered to reflect the actual value.

(11) This puts the speed value into an array of 10 values and then shifts the values over so the next value can be stored.

(12) The average of the ten data values is computed and stored.

(13)This value is compared to the current maximum speed value, and if it is higher, the max speed is replaced. If not the max speed remains the same.

(14) If the value is higher than the max speed, that value is stored in the max speed variable.

(15) The average speed and the max speed are sent to the LCD screen. The words, “MPH” and “Max” are also put on the screen.

(16) the program starts over by looking for motion again. This happens until the unit is turned off by the switch.

## V Experimental Results

The first prototype designed had a 1inch diameter lens attached to the box. This lens is a 1:1 lens, meaning the focal distance is equal to the diameter. Also, a string of four LEDs was attached and protruding out of the box.

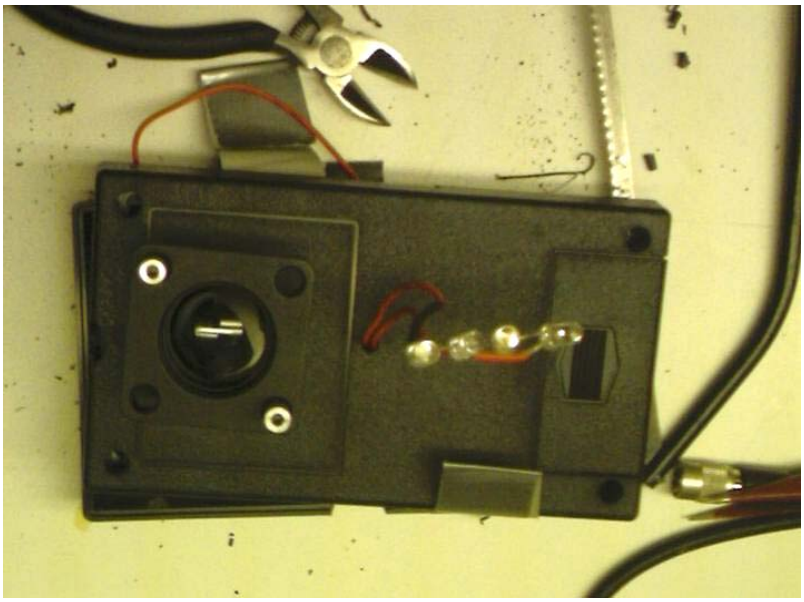


**Figure 10- 1st prototype**

The equation m= h'/h, where m is magnification, h is the height of an object in object space and h' is the height in image space, can be used to find the magnification value. The detector size is .1 in.

M= (-.1"/1.0") = -.1,

Knowing this and the focal length we can figure out how far the road needs to be from the lens to get focused images.

1/-.1p=1/1 + 1/p. p = -11inches.

In conclusion, the 1” focal length lens will not work for this project because the distance from the skateboard deck to the ground is roughly 6 inches. A different lens must be used.

Knowing that p had to be equal to -6” that would make q = .6” and the focal distance to be .5454”.

1/.6” = 1/f - 1/6”

1/f = 1.833

f = .545454”

A new box had to be designed to hold the smaller lens and also mount more leds.

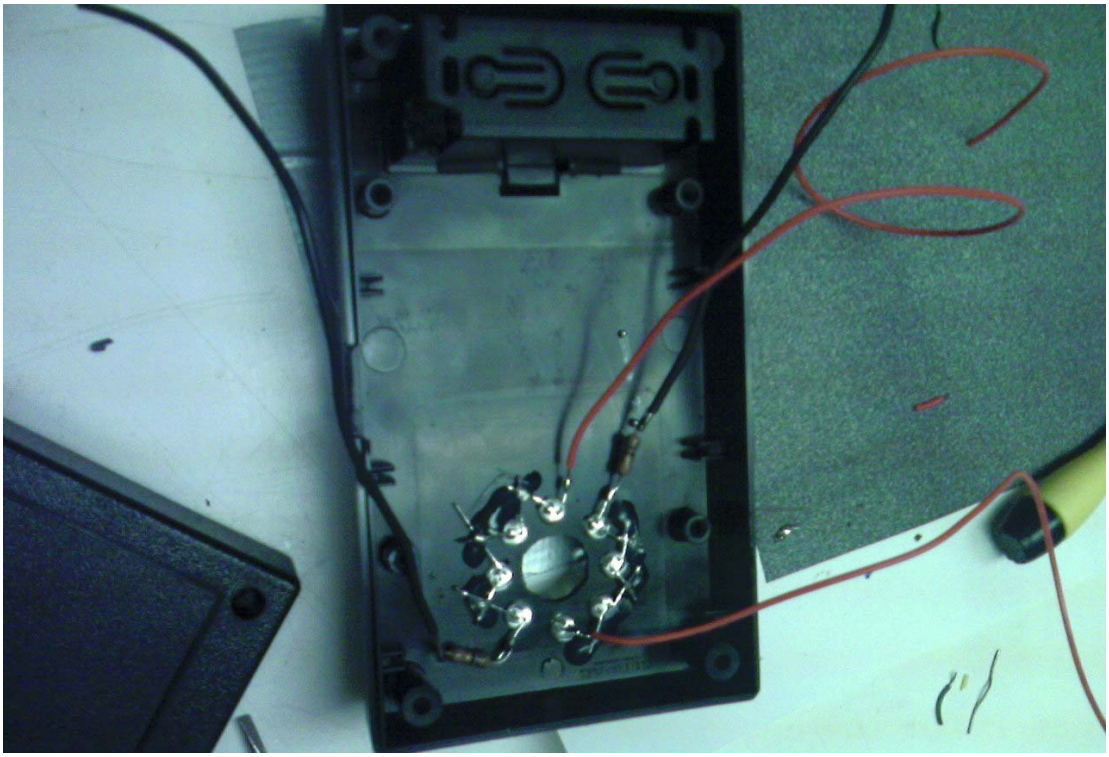


**Figure 11- Led configuration**

The leds were wired 4 in series in 2 sets. They were hot glued into the box in a circular pattern around the opening for the lens. A switch was also wired into the design. This was not at first part of the project, but a switch provides the ability to turn off the device to

save on battery life without having to unplug the battery. The lens was mounted using a tube from a plastic spindle that holds blank CDR media. The distance the lens had to be was calculated in the thin lens equation. The lens was secured to the tube and the tube to the board by epoxy.

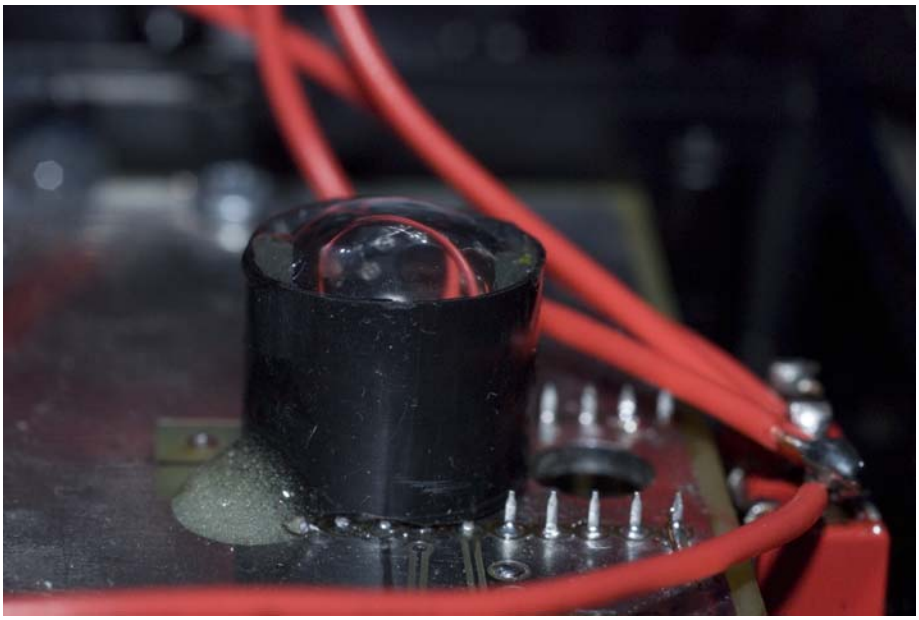


**Figure 12- lens mounting tube**

Another piece of equal diameter tubing will protrude out of the box and down onto the lens. This will prevent stray light from the leds from getting to the sensor.

When trying to calibrate the device we looked at the squal value. This is a number that represents the quality of the image the sensor is seeing. The Squal value * 4 = Number of features. Using Figure 9 as a reference we knew we had to get values at least 60 or higher. When testing on certain surfaces, during the brightest part of the day, the images tend to wash out to white because it is too bright. This causes very low Squal and features values and doesn't allow the device to read accurately. It was tested on carpet, red brick, asphalt, sidewalk, various shades of pavement, and grey brick. The lightest shades of pavement were the only surfaces to have troubles with it being too bright. The

ADNS-3080 has an automatic gain control, but is designed for indoor use, so it can not handle extremely bright areas.

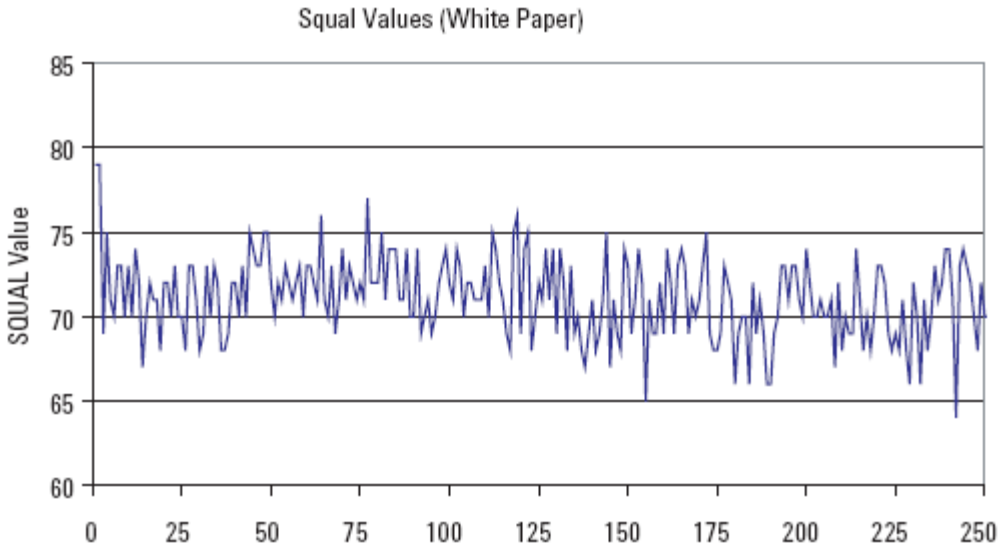



**Figure 13- Squal values on white paper [2]**

| Type of Surface | % of Time Good Values Were Acquired |
|---|---|
| Red brick | 85% |
| Grey brick | 85% |
| Asphalt | 85% |
| Darker concrete | 70% |
| White concrete | 40% |

**Figure 14- *estimated values from testing**

The deltaX values range from 0 to 127. These values need to be converted to miles per hour. A time of 2ms was determined to be the time it takes for the data to be captured. The resolution and magnification had to be taken into consideration. We knew that the

mouse could read at max 40 inches per second, prior to magnification. Using all of these values, a equation was derived.

DeltaX value/ dpi/read time * magnification* seconds in an hour * miles in an inch = conversion

If we took this value and divided the delta x value by it we would then get a conversion factor that we could multiply the incoming data by to get miles per hour in one step.

DeltaX/1600/.002 * 11 * 3600 * 1.5e-5 = conversion  delta x/conversion = **5.12**

5.12 is the value needed to convert the deltaX values into MPH.


**Figure 15- welcome screen**


**Figure 16- display**

**Figure 17- bottom of device**

To compensate for the extremely bright conditions during the middle of the day, I designed a second attachment tube that has a smaller opening. Much like the aperture of a

camera, the smaller opening lets less light reach the sensor, to prevent the washing out of features. In order to calibrate the speed, I tested the device along side a car with a GPS speed monitoring device. After several tests and fine tuning the integration value in the code, the device was working very accurate at slower speeds, but loosing accuracy as the speed increased. This is due to the error in calculations. The conversion factor of 5.12 was truncated to 5 and this difference was being magnified at higher speeds. The values in the chart below are average values.

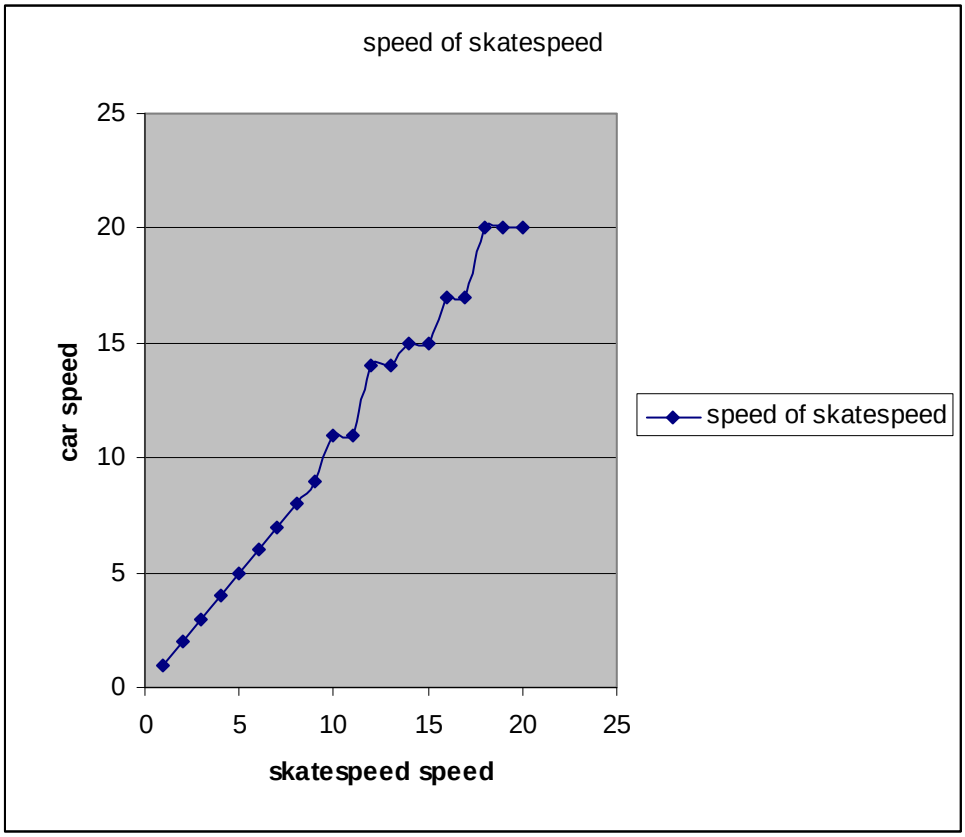



**Figure 18- speed test results**

## VI Conclusion

This report covers all aspects of my senior design project for the Spring 2007 semester at UCF. It was determined that it is possible to accurately measure the traveling speed of an electric skateboard using an optical mouse sensor. It was also concluded that this device could be used to measure the speed of any moving object up to 20 miles per hour as long as the correct focal length lens was used and it was the correct distance from the surface to be measured. The major limitation of this device is the focal area. With any skateboard, the distance from the side of the deck to the surface it is traveling on varies when maneuvering. This causes the sensor to sometimes not be able to acquire data values, therefore skewing the average speed. The green led on the top of the unit illuminates when data is being read to allow the user to know what angle the board needs to be to get data.

The total cost of this unit was $180.51 (refer to Appendix E). The overall cost could be much less if the sensor was bought directly from Avago, the company that produces them. For this project, the sensor was taken out of a consumer mouse, the Logitech MX518. This mouse cost roughly $50. Also if this unit was to be mass produced the cost per circuit board would be much less also. I ordered two boards for roughly $81. These two costs were by far the most expensive parts of the design.

A optical mouse sensor, like the ADNS-3080 used for this project, are highly sophisticated chips. They can perform many tasks very quickly. They capture images, compare the images, make assumptions on movement, automatically control gain, shutter speed, and frame rate. Paired with a PIC, this technology could be used for unlimited applications.

## VII. List of References


[1] Optical Mice and how they Work. Agilent Technologies.

http://literature.agilent.com/litweb/pdf/5988-4554EN.pdf

[2] ADNS-3080 Data Sheet. Avago Technologies.

http://www.avagotech.com/assets/downloadDocument.do?id=2307

[3] Understanding Optical Mice. Avago Technologies.

http://www.avagotech.com/assets/downloadDocument.do?id=2557

[4] Optical Mouse Cam. http://spritesmods.com/?art=mouseeye&page=1

[5] Humanoids 16-264- Vision.

http://www.contrib.andrew.cmu.edu/~ttrutna/16-264/Vision_Project/

[6] 32 bit microcontroller model cars. Carsten Grob.

http://www.siski.de/~carsten/diplomarbeit.html

[7] ADNS-3080 Optical Mouse Sensor. Avago Technologies.

http://www.avagotech.com/assets/downloadDocument.do?id=2310

[8] Basic Optics for Engineers. Dr. Al Ducharme.

[9] PIC18FXX2 Data Sheet. MicroChip

http://ww1.microchip.com/downloads/en/devicedoc/39564b.pdf

[10] CET4134C- Final Project-Go Weather. Dr. Al Ducharme.

## VIII. Bibliography


Horn, Berthold K.P. and Brian G. Schunck. Determining Optical Flow. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf

Horn, Berthold K.P. Determining Constant Optical Flow. 2003 http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf

Owens, Richard L. Optical Mouse Technology. 26-Apr-06 http://www.mstarmetro.net/~rlowens/OpticalMouse/

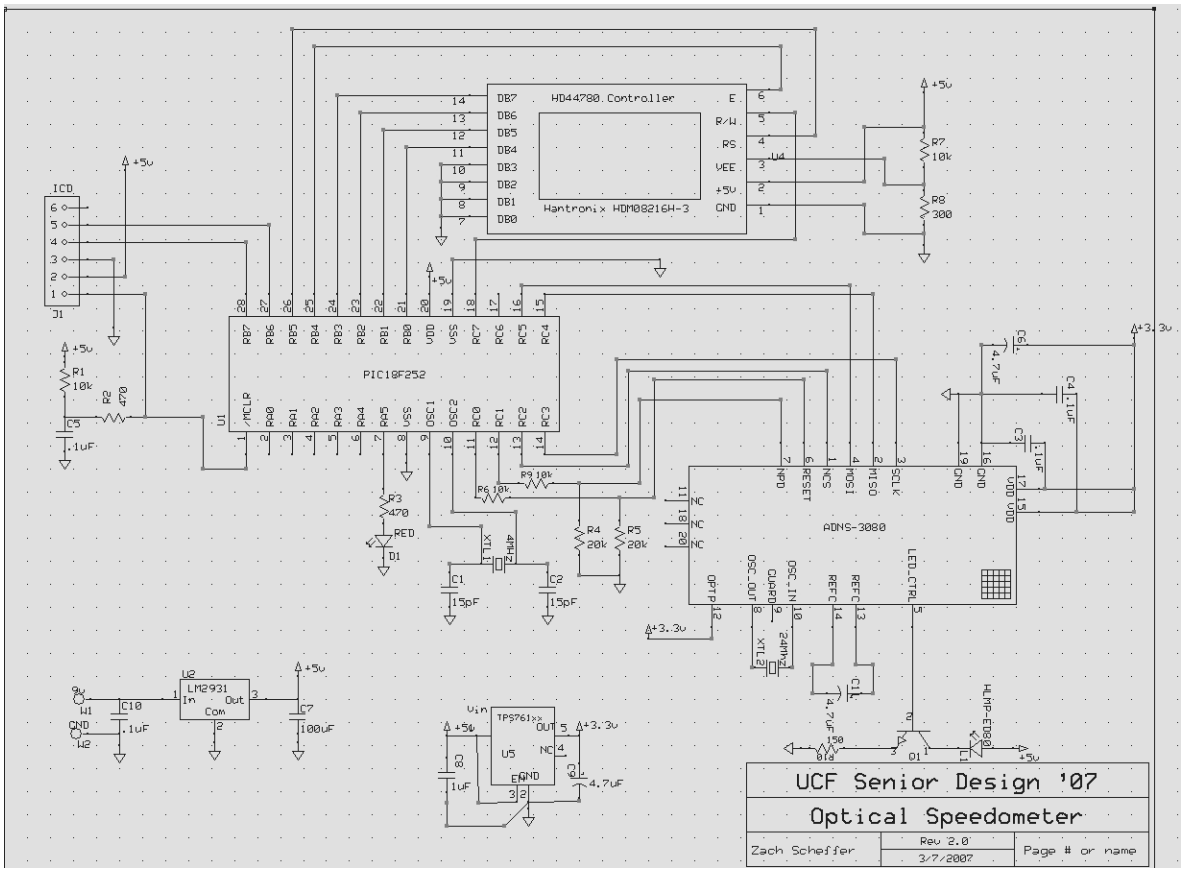
Riaziat, M. L. Introduction to high-speed electronics and optoelectronics. New York. Wiley, c1996

Stewart, James and Miao, Kai. The 8051 Microcontroller, second edition. Prentice Hall Inc. New Jersey, 1999

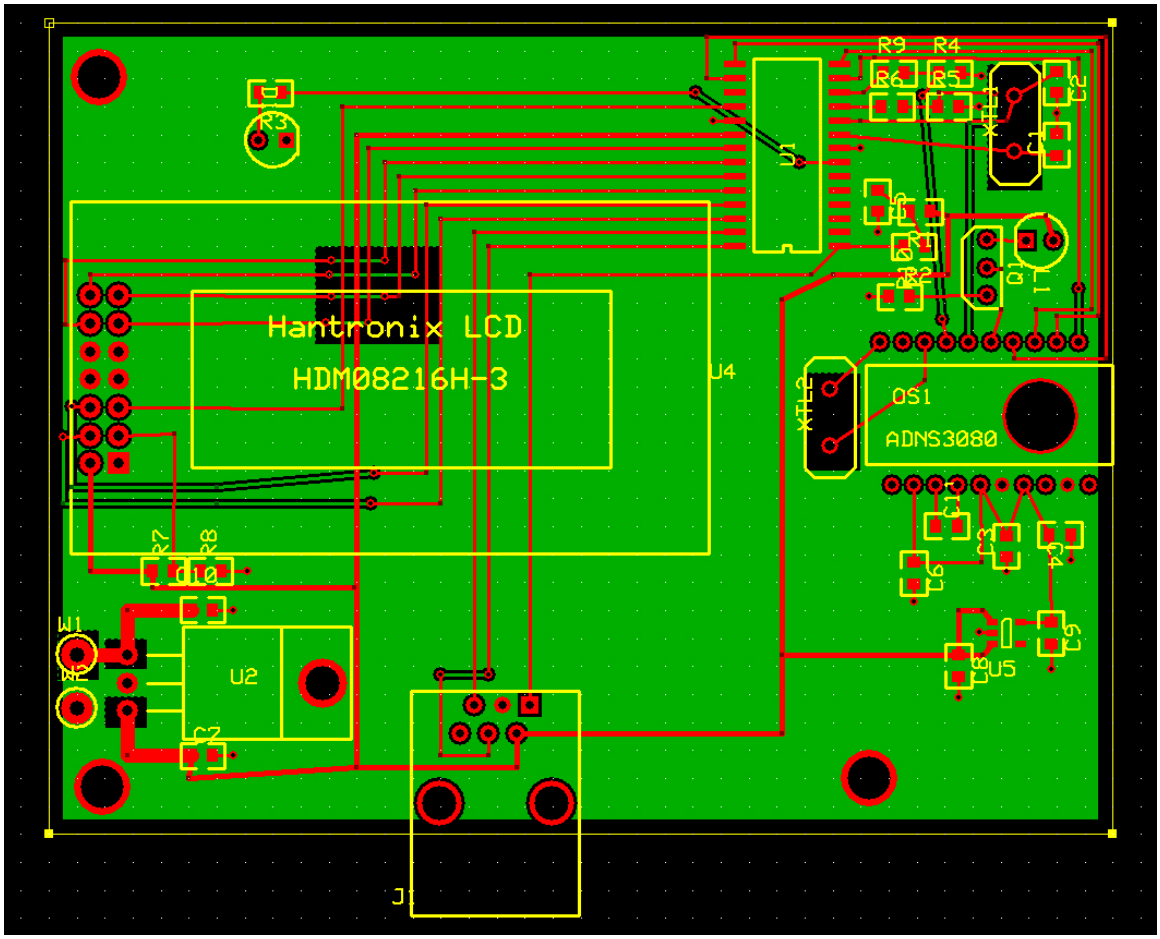
PIC19Fxx2 data sheet. http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

# Appendices

## A. Optical Speedometer schematic



## B. Optical Speedometer .PCB design



## C. Optical Speedometer device code

```
#include <p18f252.h>
#include <delays.h>
#include <portb.h>
#include <spi.h>
#include "xlcd.h"

#define LED   PORTAbits.RA5
#define RESET PORTCbits.RC0
#define NPD PORTCbits.RC1
#define NCS PORTCbits.RC2

void Initialize_ADNS3080(void);
void GetData_ADNS3080(void);


////////////////// Global Variables ////////////////
ram int dat[10];
unsigned char topmsg[]={"SkateSPD"};
unsigned char botmsg[]={"Scheffer"};
unsigned char disp_top[]=  {"00.0 MPH"};
unsigned char disp_bot[]=  {"Max 00.0"};
unsigned char DeltaX;
unsigned char DeltaY;
unsigned int Rate;
long Freq;
unsigned char MaxPix;
unsigned char Pixel_Ave;
unsigned char Motion;
unsigned int Speed;
long SpeedAve;
unsigned int SpeedPrev;
unsigned char Squal;
unsigned int Features;
unsigned char current;
unsigned char config;
unsigned char MaxSpeed;
unsigned int Speedval;
int t_dat;
unsigned char Motion_Flag,OVF_Flag;

unsigned char Integrity;
       long count;
       long DeltaXAve;
       long aves;
       int j;

////////////////////////////////////////////////
void main(void)
{

             int i;

             EnablePullups();                              // Enable pullup
resistors on PIC

             TRISA = 0b00000101;                           // Set PORTA
directions
             TRISC = 0b00110000;                           // Sete PORTC
directions
             OpenXLCD (FOUR_BIT & LINES_5X7);  // Open the LCD for display
```

```
            while(BusyXLCD());                              // wait while LCD
completes last operation
            WriteCmdXLCD(CLS);                              // clear the LCD
            while(BusyXLCD());                              // wait while LCD
completes last operation
            WriteCmdXLCD(HOME);;                            // return cursor to
home position
            putsXLCD(topmsg);                               // display turn on
message
            SetDDRamAddr(0x28);                             // Move cursor to
second line
            while(BusyXLCD());                              // wait while LCD
completes last operation
            putsXLCD(botmsg);                               // display second
line on LCD

            Delay10KTCYx(200);                              // Welcome message
delay

            Initialize_ADNS3080();                          // Initialize
ADNS3080 chip for data collection
            SpeedAve = 0;                                   // Reset Average
Speed value
            Speedval = 0;
            MaxSpeed = 0;                                   // Reset Maximum
Speed

            for(i=0;i<10;i++)
            {
                  dat[i] = 0;                                       // Clear
speed data array
            }



while(1)
{

            GetData_ADNS3080();

      //    if(Speed != 0)                                          // Data check
so that 0's are removed
      //          SpeedPrev = Speed;                        // set to previous
value
      //    if(Speed == 0)                                          // if Speed
is 0 then use previous value
      //          Speed = SpeedPrev;


            if(DeltaX != 255)
            {


      Speed = (unsigned int)DeltaX;     // reverse direction of data so
increasing DeltaX is increasing speed
            Speed = 10*Speed / 5;                           // Convert DeltaX
counts to MPH
            //if(Speed >= 0)
            //{
                        for(i=1;i<10;i++)               // This for loop
shifts all of the data by one index
                                                                        //
this way new data comes into the right most position (on the display)
```

```
                                                                        // the
last data point is lost
                        {
                              dat[i-1] = dat[i];        // shift the data
                        }
                        dat[9] = Speed;
                  SpeedAve = 0;
                  for(i=0;i<10;i++)
                  {
                        SpeedAve = SpeedAve + dat[i];    // Sum all speed
values for average calculation
                  }

                  SpeedAve = SpeedAve/10;                        // divide by
number of data points for average value
                  Speedval = (unsigned int)SpeedAve;             // Change
variable to unsigned int
                  if(Speedval > MaxSpeed)
                        MaxSpeed = Speedval;
            //}
            }
                  while(BusyXLCD());                             // wait while
LCD finishes last operation
                  WriteCmdXLCD(CLS);                             // Clear the
LCD
                  while(BusyXLCD());                             // wait while
LCD finishes last operation
                  WriteCmdXLCD(HOME);                            // put cursor
at home position
                  while(BusyXLCD());

                  disp_top[0] = Speedval/100 + '0';

                  disp_top[1] = (Speedval/10 - (10*(Speedval/100))) + '0';
                  disp_top[3] = Speedval%10 + '0';

                  disp_bot[4] = MaxSpeed/100 + '0';

                  disp_bot[5] = (MaxSpeed/10 - (10*(MaxSpeed/100))) + '0';
                  disp_bot[7] = MaxSpeed%10 + '0';

                  putsXLCD(disp_top);                            // Display
current speed on LCD
                  while(BusyXLCD());                             // wait while
LCD finishes last operation
                  SetDDRamAddr(0x28);                            // Move
cursor to second line
                  while(BusyXLCD());                             // wait while
LCD finishes last operation
                  putsXLCD(disp_bot);                            // Display
maximum speed on LCD

                  Delay10KTCYx(5);                               // Small
delay so user can view it

}

}

void Initialize_ADNS3080(void)
{
      LED = 1;                                                   // Turn green
LED on
```

```
	NCS = 1;                                                    // Set NCS
line to high
	NPD = 0;                                                    // set NPD
line to low
	Delay10KTCYx(1);
	NPD = 1;                                                    // set NPD
line to high
	Delay10KTCYx(60);
	RESET = 0;                                                  // Set Reset
to low
	Delay10KTCYx(1);
	RESET = 1;                                                  // Set Reset
to high to reset ADNS3080
	Delay10KTCYx(1);
	RESET = 0;                                                  // Set Reset
to low completing reset operation
	OpenSPI(SPI_FOSC_4, MODE_10, SMPEND);   // Open PIC SPI port for
communications to ADNS3080
	Delay10KTCYx(10);
	NCS = 1;                                                    // Set NCS to
high turning off Com to ADNS3080
	Delay10KTCYx(1);
	NCS = 0;                                                    // set NCS to
low turning ON Com to ADNS3080
	Delay10KTCYx(1);
	putcSPI(0x8A);                                              // write to
Config register 0x0A using 0x8A to set first bit to 1 for write operation
	Delay10TCYx(10);
	putcSPI(0x19);                                              // Set to
1600 dpi
	//putcSPI(0x00);                                    // Set to 400 dpi
	Delay10TCYx(10);
	putcSPI(0x0A);                                              // write to
Config register 0x0A using 0x8A to set first bit to 1 for write operation
	Delay10TCYx(10);
	config = ReadSPI();                                 //Check Config data
should be 00010000 or 0x10
	NCS = 1;                                                    // Set NCS to
high turning OFF Com to ADNS3080


}

void GetData_ADNS3080(void)
{

	//unsigned char Rate_Upper,Rate_Lower;
	//long PixSum;


	LED = 0;                                            // Turn green LED
off
	// Reset all variables
	//    Squal = 0;
	//    PixSum=0;
	//    Features = 0;
		DeltaX = 0;
		DeltaXAve = 0;
		DeltaY = 0;
	//    MaxPix = 0;
	//    Pixel_Ave = 0;
	//    Rate_Lower = 0;
	//    Rate_Upper = 0;
```

```
	//	Rate = 0;

	// Done resetting all variables

	NCS = 1;                                                // Turn ADNS3080
Com Off
	Delay10TCYx(1);
	NCS = 0;                                                // Turn ADNS3080
Com On
	Delay10TCYx(1);
	aves = 0;
	for (j=0;j<10;j++)
	{
	Motion_Flag = 0;                                // reset motion flag used
to determine if there is motion
	OVF_Flag = 0;                                   // reset OVF (overflow
flag)

	count = 0;                                              // reset count
variable for max time looking for data
	do
	{
		putcSPI(0x92);                                          // Clear the
DeltaX and DeltaY registers
		putcSPI(0x00);                                          // Write 0 to
clear DeltaX and DeltaY
		Delay100TCYx(100);                              // wait for 30ms?
		putcSPI(0x02);                                          // Write to
02 register to check motion
		Delay10TCYx(8);
		Motion = ReadSPI();                             //Get Motion
		Nop();
		putcSPI(0x03);
		Delay10TCYx(6);
		DeltaX = ReadSPI();                             //Get DeltaX
		Nop();
		putcSPI(0x04);
		Delay10TCYx(6);
		DeltaY = ReadSPI();                                     //Get DeltaY
		Motion_Flag = 0b10000000 & Motion;      // Mask to look at motion
bit
		Motion_Flag = Motion_Flag >> 7;
		OVF_Flag = 0b00010000 & Motion;         // Mask to look at OVF bit
and set OVF flag
		OVF_Flag = OVF_Flag >> 4;
		count = count + 1;
	}	while(Motion_Flag == 0 && count <= 10);                    //
check to see if there is motion

	Nop();
	//	putcSPI(0x05);
	//	Delay10TCYx(6);
	//	Squal = ReadSPI();
	//	Features = Squal;
	//	Features = Features << 2;
	//	Features = Features * 4;
	//	Delay10TCYx(10);
	//	putcSPI(0x07);
	//	Delay10TCYx(10);
	//	MaxPix = ReadSPI();                     //Get Maximum pixel value
	//	Delay10TCYx(10);
	//	putcSPI(0x06);
	//	Delay10TCYx(10);
```

```
	//	Pixel_Ave = ReadSPI();			//Get Pixel Average value
	//	PixSum=(Pixel_Ave*256)/900;
	//	Pixel_Ave = (unsigned char)PixSum;
	//	Delay10TCYx(10);
	//	putcSPI(0x11);
	//	Delay10TCYx(10);
	//	Rate_Upper = ReadSPI();			//Get lower 8 bits of
frame rate
	//	Delay10TCYx(10);
	//	Rate = Rate_Upper;
	//	Rate=Rate << 8;					//shift upper to
left 8 bits
	//	putcSPI(0x10);
	//	Delay10TCYx(10);
	//	Rate_Lower = ReadSPI();			//Get upper 8 bits of
frame rate
	//	Rate = Rate | Rate_Lower;	//combine upper and lower to get
16 bit frame rate


	if(DeltaX >=0 && DeltaX <=127)		// make sure only forward
direction data is used
		DeltaX = 255;					// make speed equal to 0
in the negative direction
	DeltaX = 255-DeltaX;
	if(Motion_Flag == 1 && DeltaX !=0)
	{
		LED = 1;
		DeltaXAve = DeltaXAve + DeltaX;
		aves = aves + 1;
	}
	}
	if(aves >=1)
	{
		DeltaXAve = DeltaXAve/aves;
		DeltaX = (unsigned char)DeltaXAve;
	}
	if(aves == 0)
		DeltaX = 0;

	NCS = 1;							// Turn ADNS3080
Com off


}
```

## D. Project Proposal

Optic speedometer
Zach Scheffer

Technical Report Submitted in Partial Fulfillment of the
Engineering Technology Project Course
ETG 4950C
Spring Semester, 2007

January 26, 2007

I am proposing to design and build a speedometer with a digital readout for use with an electric skateboard. Using an optical device, similar to one found on a computer optical mouse, the device will record images of the passing ground beneath the skateboard. The images will then be processed and x and y coordinates will be produced describing the movement of the ground relative to the optical device. A PIC will collect the data and calculate how fast the sensor, therefore the skateboard, is moving. I will be using the PIC18F252. The PIC will also display the speed to the user of the skateboard via a LCD screen mounted on the top of the nose of the skateboard (fig. 1-1). The PIC, sensor, and other necessary components will be assembled in a project box to be mounted on the under side of the skateboard, on top of the wireless remote sensor box (fig1-2). The device will work up to speeds of 22 mph. A LED will illuminate the ground beneath the optical sensor using a similar system as an optical mouse, illustrated in fig 1-4.

The ADNS-2051 Optical Mouse Sensor is used in most common computer mice. The output format is in X and Y directions, each direction having a two channel output. The data can be accessed by use of these four outputs, pins 2 through 5, or pin 16, the serial input/output pin on the chip as seen in fig 1-3. The output is separated into positive and negative x and y directions. If the y direction is to be used as the forward reference, the negative y value will not be needed since we are only measuring the speed traveling forward. Both the negative and positive values of x can be used to account for total distance traveled since traveling completely straight is unrealistic. A simple application of the distance formula, $\sqrt{((x_2-x_1)^2+(y_2-y_1)^2)}$, can be used to figure out the overall distance traveled. There are two major issues that will have to be solved for this system to work. The first being to develop the system in such a way that the optics can recognize speeds up to 22 mph, or roughly 9.8 meters per second. The second is the distance from the sensor to the ground. It will be roughly 4 inches above the surface.

fig 1-1


fig 1-2

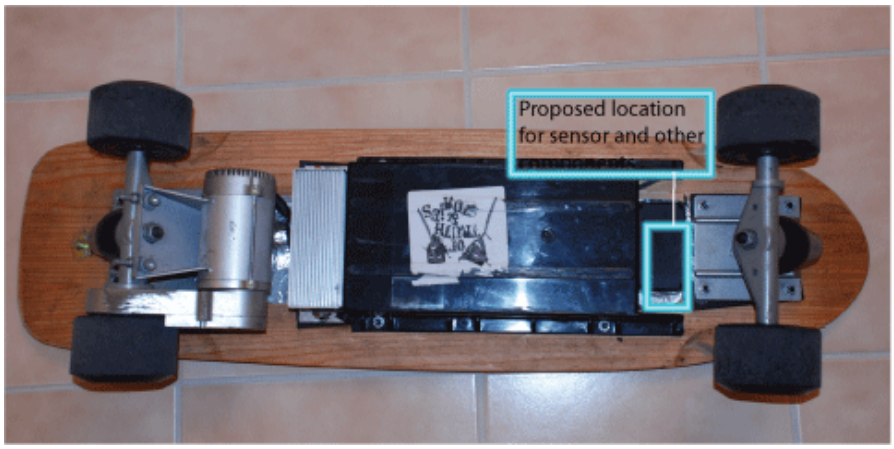



fig 1-3 (taken from the A2051- Data sheet)

**Outline Drawing of ADNS-2051 Optical Mouse Sensor**

**Pinout**

| Pin | Pin | Description |
|---|---|---|
| 1 | SCLK | Serial port clock (input) |
| 2 | XA | XA quadrature output |
| 3 | XB | XB quadrature output |
| 4 | YB | YB quadrature output |
| 5 | YA | YA quadrature output |
| 6 | XY_LED | LED control |
| 7 | REFA | Internal reference |
| 8 | REFB | Internal reference |
| 9 | OSC_IN | Oscillator input |
| 10 | GND | System ground |
| 11 | OSC_OUT | Oscillator output |
| 12 | GND | System ground |
| 13 | $V_{DD}$ | 5.0 volt power supply |
| 14 | R_BIN | LED current bin resistor |
| 15 | PD | Power down pin, active high |
| 16 | SDIO | Serial data (input and output) |

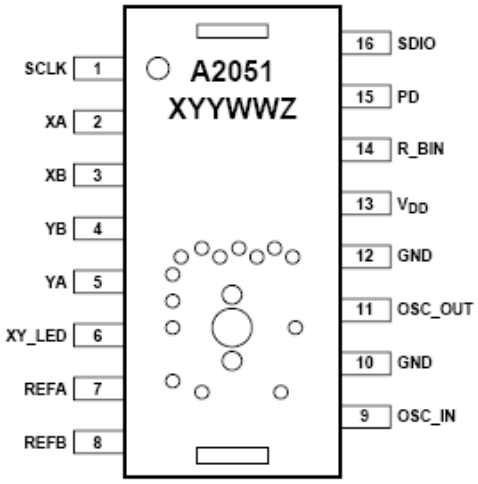



**Figure 1. Top view.**

fig 1-4 (taken from the A2051 Data sheet)

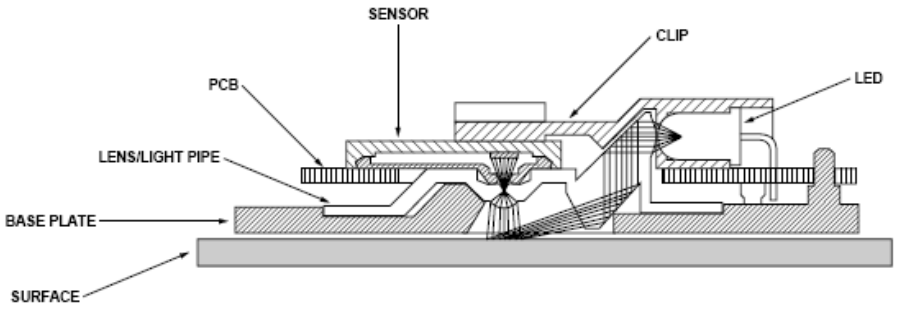

SENSOR
CLIP
PCB
LED
LENS/LIGHT PIPE
BASE PLATE
SURFACE

**Table I: Timeline**

| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Brainstorm ideas | * | * | * | | | | | | | | | | | | |
| Research topics | | * | * | * | * | | | | | | | | | | |
| Project Proposal | | * | * | | | | | | | | | | | | |
| Organize Parts- Optics, PIC, Skateboard, LCD screen, Etc. | | | * | * | * | | | | | | | | | | |
| Design how the PIC will Read data from the Optical Device | | | | | * | * | * | | | | | | | | |
| Code the process of the PIC compiling data and calculating the speed | | | | | * | * | * | * | * | | | | | | |
| Code the process of the PIC displaying the data on the lcd screen | | | | | | * | * | * | * | | | | | | |
| Design circuit board layout | | | | | | | | * | * | * | | | | | |
| Design project box | | | | | | | | * | * | * | | | | | |
| Order circuit board | | | | | | | | | * | * | * | | | | |
| Order project box | | | | | | | | | * | * | * | | | | |
| Assemble project box | | | | | | | | | | * | * | * | | | |
| Mount system on skateboard | | | | | | | | | | | * | * | * | | |
| test | | | | | | | | | | | | * | * | * | |
| | | | | | | | | | | | | | | | |

**Engineering Specification**

Speedometer with LCD readout using optics

This speedometer will be attached to an electric skateboard for use in a residential environment. It is expected to operate at speeds up to 22 mph and to last over 4 years without major repair. It should be accurate to the tenth of a mph. The system should be able to withstand moderate bumps.

| | |
|---|---|
| Max measurable speed | 22 mph |
| Input | Optics device |
| Output | LCD screen 3 digits and one decimal |
| Active devices | Optics, PIC |
| Maximum size | 2" x 3 ½ "x 2" plus the size of the lcd(not to exceed 2"x4" |
| Maximum weight | 2 pounds max |
| Operating ambient temp | 40 to 120 deg. F |
| Case | Black plastic |

Table II: Engineering Specifications

**References**

http://www.avagotech.com/assets/downloadDocument.do?id=1568 – Data sheet for the ADNS 2610 optical mouse sensor.

Data sheet for the Agilent ADNS-2051 Optical Mouse Sensor

Horn, Berthold K.P. and Brian G. Schunck. Determining Optical Flow. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf

Horn, Berthold K.P. Determining Constant Optical Flow. 2003 http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf

Owens, Richard L. Optical Mouse Technology. 26-Apr-06 http://www.mstarmetro.net/~rlowens/OpticalMouse/

Riaziat, M. L. Introduction to high-speed electronics and optoelectronics. New York. Wiley, c1996

Stewart, James and Miao, Kai. The 8051 Microcontroller, second edition. Prentice Hall Inc. New Jersey, 1999

PIC19Fxx2 data sheet. http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

## E. Bill of Materials

| Vendor: | UCF Contact | Optical Speedometer |
|---|---|---|
| Digi-Key Corporation<br>800-344-4539<br><br>Mouser<br>(800) 298-5076 | Dr. Al Ducharme<br>Engineering Technology<br>407-823-0770 | |

ETG4950h (Spring07)

All in Stock as of 3/15/07

# Bill of Materials

| | | | | 1 Boards | | | |
|---|---|---|---|---|---|---|---|
| **Item** | **Identifier** | **P/N** | **1 board** | **Qty to Order** | **Cost/unit** | **Total Cost** | **Vendor** |
| 3.3V Regulator | U5 | 296-2695-1-ND | 1 | 1 | $ 1.0000 | $ 1.00 | Digikey |
| PIC18F252 | U1 | PIC18F252-I/SP-ND | 1 | 1 | $ 9.2500 | $ 9.25 | Digikey |
| 24MHz | XTL2 | X095-ND | | 1 | $ 0.4000 | $ 0.40 | Digikey |
| LP2950ACZ-3.3-ND | Ip | LP2950ACZ-3.3-ND | 1 | 1 | $ 1.0500 | $ 1.05 | Digikey |
| 4MHz Crystal | XTL1 | CTX006-ND | 1 | 1 | $ 0.9400 | $ 0.94 | Digikey |
| ICD Connector ** | J1 | A9031-ND | 1 | 1 | $ 0.7200 | $ 0.72 | Digikey |
| 5V Regulator | U2 | 497-1565-5-ND | 1 | 1 | $ 1.1300 | $ 1.13 | Digikey |
| 4.7 uF Polarized Cap | C9, C6, C11 | | 3 | 10 | $ 0.4400 | $ 4.40 | Digikey |
| 0.1 uF Caps | C10, C5, C3, C4 | 1109PHCT-ND | 6 | 10 | $ 0.8600 | $ 0.86 | Digikey |
| 470 ohm | R3, R2 | 470EBK-ND | 1 | 5 | $ 0.0520 | $ 0.26 | Digikey |
| 9V Battery Connector (Snap) | W1,W2 | 2239K-ND | 1 | 1 | $ 0.8300 | $ 0.83 | Digikey |
| 10k ohm | R7, R1, R6, R9 | 10KEBK-ND | 4 | 5 | $ 0.0520 | $ 0.26 | Digikey |
| 300 ohm | R8 | 300EBK-ND | 1 | 5 | $ 0.0520 | $ 0.26 | Digikey |
| 20k ohm | R4, R5 | | 4 | 10 | $ 0.0800 | $ 0.80 | Digikey |
| 15pF Cap | C1, C2 | 1012PHCT-ND | 2 | 2 | $ 0.0845 | $ 0.17 | Digikey |
| 150 ohm | R10 | | 1 | 1 | $ 0.0520 | $ 0.05 | Digikey |
| 100 uF cap | C7 | | 1 | 1 | $ 4.2500 | $ 4.25 | Digikey |
| 28-Pin DIP Sockets | U1 | ED3328-ND | 1 | 1 | $ 2.0000 | $ 2.00 | Digikey |
| Box | BOX | SR251B-ND | 1 | 1 | $ 8.9500 | $ 8.95 | Digikey |
| 1uF cap | C8 | | 4 | 10 | $ 0.1100 | $ 1.10 | Digikey |
| LED | D1 | 160-1656-ND | 1 | 1 | $ 0.4100 | $ 0.41 | Digikey |
| 2N2222 | Q1 | 497-3106-5-ND | 1 | 1 | $ 1.0900 | $ 1.09 | Digikey |
| Board Connector | - | 929836-09-36-ND | 1 | 1 | $ 3.3900 | $ 3.39 | Digikey |
| PCB board | - | | 1 | 1 | $ 81.0300 | $ 81.03 | Digikey |
| Logitech MX518 mouse | | | 1 | 1 | $ 50.0000 | $ 50.00 | Best Buy |
| LCD | U4 | HDM08216H-3-S00S | 1 | 1 | $ 5.9100 | $ 5.91 | Mouser |
| | | | | | | $ 180.51 | |