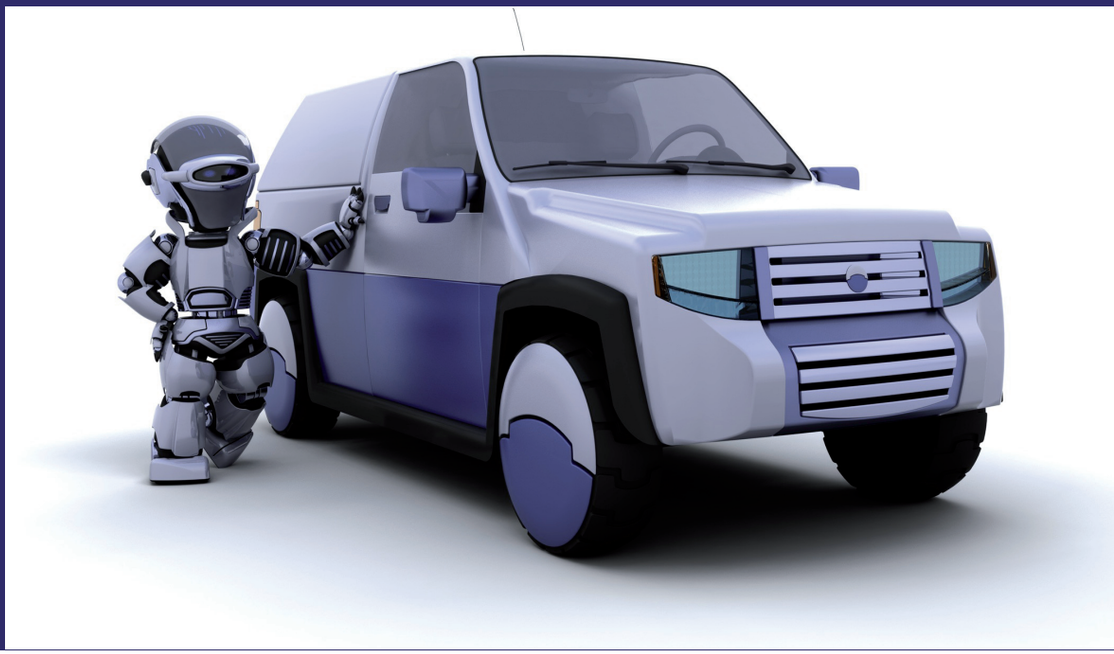
In this thesis, we built an autonomous navigational system that can support any roaming device exactly plot its position from a starting point. The system is based on ADNS-2610 optical chip and Hitachi HM55B Compass Module which together with basic stamp 2 module gathers information about its position and plots it in simulation software in real time. The simulation software consisted of some fields to show values such velocity, acceleration, and displacement, heading etc of the device (robot) in real time and plot the movement on a 2D map. The 2D map is in fact a continuous line on the grid that shows the path that is traveled by the robot. This helped us to visually detect any flaws in our simulation software. We also have implemented a line following algorithm and built a corresponding required hardware from scratch. This helped us to cause the robot to follow a line and along with the simulation software plot in. The last part of the thesis was to detect the end of the line when reached, the robot stops as it does so.



Supta Richard Philip

Supta Richard Philip was born at July 1983 in Bangladesh.Now He's doing masters in Computer Science at University of Trento,Italy.He completed his B.Sc. from American International University-Bangladesh(AIUB) in Computer Science.

978-3-659-10855-6

Supta Richard Philip
Amlan Chowdhury
Tohedul Islam

Autonomous Robot Navigation System And Line Following Robot

Autonomous Robot Navigation System Using Optical Mouse-based Odometry, Line Following and End of Line Detection Robot

LAP LAMBERT Academic Publishing

Supta Richard Philip
Amlan Chowdhury
Tohedul Islam

Autonomous Robot Navigation System And Line Following Robot

Supta Richard Philip
Amlan Chowdhury
Tohedul Islam

Autonomous Robot Navigation System And Line Following Robot

Autonomous Robot Navigation System Using Optical Mouse-based Odometry, Line Following and End of Line Detection Robot

LAP LAMBERT Academic Publishing

Impressum/Imprint (nur für Deutschland/only for Germany)
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Coverbild: www.ingimage.com

Verlag: LAP LAMBERT Academic Publishing GmbH & Co. KG
Heinrich-Böcking-Str. 6-8, 66121 Saarbrücken, Deutschland
Telefon +49 681 3720-310, Telefax +49 681 3720-3109
Email: info@lap-publishing.com

Approved by: Department of Computer Science, Faculty of Science, American International University - Bangladesh (AIUB), Dissertation,January 2008

Herstellung in Deutschland (siehe letzte Seite)
ISBN: 978-3-659-10855-6

Imprint (only for USA, GB)
Bibliographic information published by the Deutsche Nationalbibliothek: The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Cover image: www.ingimage.com

Publisher: LAP LAMBERT Academic Publishing GmbH & Co. KG
Heinrich-Böcking-Str. 6-8, 66121 Saarbrücken, Germany
Phone +49 681 3720-310, Fax +49 681 3720-3109
Email: info@lap-publishing.com

Printed in the U.S.A.
Printed in the U.K. by (see last page)
ISBN: 978-3-659-10855-6

Acknowledgement

We would like to thank our advisor Mr. Shakil Mahbubul Huq, for his reviews, invaluable advice without which this thesis would have been nothing but a scratch and also for his help writing thesis paper. We would like to express our deepest gratitude for his invaluable guidance and support. Then we would like to thank people from our other thesis groups like Farnush Anwar, Md. Ibne - Sina who really helped us by providing valuable information and resources during our research. We would also like to show appreciation to other thesis group members for their help in many ways. We would also like to show our gratitude to all of our teachers, parents for being so understanding. Last but not least we would like to express gratitude to our Department Coordinator Mr. Mashiour Rahman and Head of the Department Dr. Kazi A. Kalpoma for giving us more time to accomplish the work in the best possible manner.

Abstract

In this thesis we built an autonomous navigational system that can support any roaming device exactly plot its position from a starting point. The system is based on ADNS-2610 optical chip and Hitachi HM55B Compass Module which together with basic stamp 2 module gathers information about its position and plots it in simulation software in real time. The simulation software consisted of some fields to show values such velocity, acceleration, and displacement, heading etc of the device (robot) in real time and plot the movement on a 2D map. The 2D map is in fact a continuous line on the grid that shows the path that is traveled by the robot. This helped us to visually detect any flaws in our simulation software.

We also have implemented a line following algorithm and built a corresponding required hardware from scratch. This helped us to cause the robot to follow a line and along with the simulation software plot in. The last part of the thesis was to detect the end of the line when reached, the robot stops as it does so.

Table of Contents

1.1 Background

Industrial and technical applications of mobile robots are continuously gaining importance due to increased reliability, accessibility and cost effectiveness. In particular, for example, uninterrupted and reliable execution of monotonous tasks such as surveillance, inspection of sites that are inaccessible to humans, e.g. tight spaces, hazardous environments or remote sites and transportation systems based on autonomous mobile robots can be cheaper than standard track-bound systems. Moreover, mobile robots are widely used in surveillance, inspection and transportation purpose. To use the technology more effectively, there exist some challenging issues which need to be addressed. The most importantly, how a robot can manage to navigate to reach its destination optimally? What the robot should do if it finds an object on its way? In such case how does it respond to find an alternative path? How does the robot distinguish objects dimensions and color? What if the sensor misses an obstacle? What does the robot do if it doesn't find the destination? All these issues are very important to take into account to develop an effective navigational system.

While deciding our thesis we stumbled upon a Robot Contest "Robocon" which is going to be held within our university on January, 2008. This inspired us to develop the navigational system for such a robot which would be able to roam freely, grab objects, and avoid obstacles at the shortest period of time.

1.2 Rational for the Thesis

Localization and accurate position mapping has always been a hot research topic in robotics arena. Many researchers debate numerous methods as an attempt to solve the problem. A careful scrutiny of latest publication reveals different approaches to handle this problem efficiently.

1.3 Statement of the Problem

In this thesis, we develop simulation software which visually helped us to understand and plot each and every movement made by the robot to a very high degree of accuracy. In localization we have used ADNS-2610 optical navigation chip and an electrical compass, the optical chip fed us with the displacement of the robot and the compass fed us with the direction i.e. the angle the robot at corresponding to north. With proper logic these data was converted to meaningful co-ordinates. The whole arrangement helped us to understand the robots movement and the simulation software correspondingly plotted it.

The line following part used a combination of very cheap hardware and intelligent logic. The basic hardware used in the line following part is two infrared receivers and only one emitter, and the logic used to solve the problem is very simple, when both the receivers are over the line the robot is asked to move forward , when only one receiver is over the line the robot is asked turn in the opposite direction.

All the sensors are controlled by a co-processor called *Basic Stamp 2* (BS2) which is used to collect the raw data about the surroundings and to send it back to the simulation high level processing.

1.4 Objectives

The objectives for creating the Path finding robot are:

- Line following algorithm and required hardware.
- Simulation software.
- Positioning and optimization of sensors.
- 2D map for robot localization and path plotting.
- Integrating the vision sensing and processing developed by another thesis group on to the robot.

In the beginning we wanted to build a system where the robot had no defined source or destination, i.e. create a random navigational system but we couldn't determine if we had arrived to the goal using the sensors we had and the our image processing only worked in ideal conditions so we had to backtrack from our previous goal and alter our plans to manually feed the robot its goal and destination.

1.5 Structure of the Thesis

The remaining of this thesis is composed of the following sections. In Chapter 2, we discuss the various ideas and technologies used in this thesis. Chapter 3, Simulation software is the main chapter of our thesis, which discusses about the main simulation software of our robot and in Chapter 4 Sensors & Line Following is discussed. The next two chapters Chapter 5, Shaft Encoder and Chapter 6, Locomotion give a brief overview of the construction of our Robot and how the sensors and movement works. The Last Chapter the conclusion gives an overview of what we achieved, what we couldn't and of course the future work to be implemented.

2.1 Intelligent Agents in Artificial Intelligence

In artificial intelligence, an *intelligent agent* is used for intelligent actors, which observe and act upon an environment, to distinguish them from intelligent thinkers isolated from the world. An agent in this sense of the word is an entity that is capable of perception and action. Such an agent might be a robot or an embedded real time software system - and is intelligent if it interacts with its environment in a manner that would normally be regarded as intelligent if that interaction were carried out by a human being [1].

It is possible to group agents into five classes based on their degree of perceived intelligence and capability:

1. Simple reflex agents,
2. Model-based reflex agents,
3. Goal-based agents,
4. Utility-based agents.
5. Learning agents.

Our robot was a simple goal based agent whose goal at any instant was to follow a line.

2.1.1 Goal Based Agents/Robots

Goal-based agents are model-based agents, which store information regarding situations that are desirable. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state

Our implementation was a goal based agent which used its sensors to acquire knowledge about its environment and after considering its goal the agent acted upon its surrounding through its actuators.

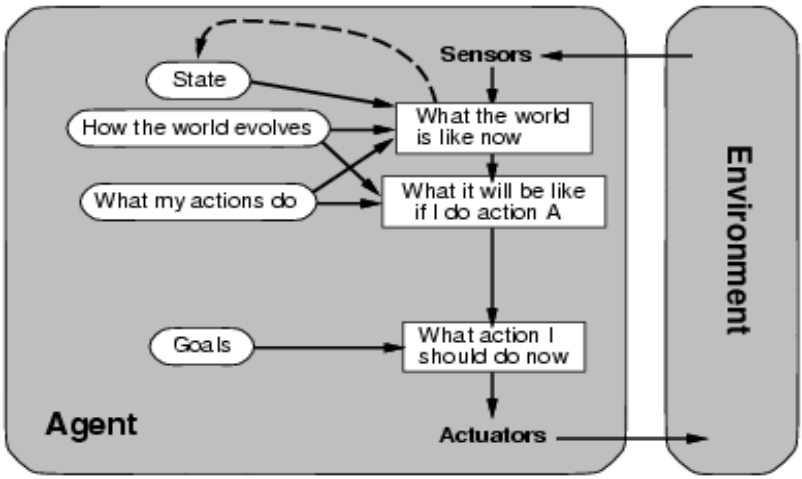


Figure 2.1: Intelligent Agent-Learning

2.2 What is a Robot?

A robot is a mechanical or virtual, artificial agent. It is usually an electromechanical system, which, by its appearance or movements, conveys a sense that it has intent or agency of its own. The word robot can refer to both physical and virtual software agents, but the latter are usually referred to as bots to differentiate [2].

While there is still discussion about which machines qualify as robots, a typical robot will have several, though not necessarily all of the following properties:

- Can sense its environment.
- Can manipulate things in its environment.
- Has some degree of intelligence or ability to make choices based on the environment or automatic control / preprogrammed sequence.
- Is programmable.
- Can move with one or more axes of rotation or translation.
- Can make dexterous coordinated movements

2.3 Embedded System

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions. It is usually *embedded* as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming.

2.3.1 Microcontroller

Basic Stamp 2 is a microcontroller board. A microcontroller is a small computing device with a built-in ALU, memory and other necessary stuff needed necessary for computing. The basic difference between a PC and microprocessor of a microcontroller is that it is capable of running independently, it does not need extra memory like the microprocessor, but the memory is small. Microcontroller is

designed to take input from the various sensors from the environment. It is less costly then a microprocessor and widely used in robotics all over the world.

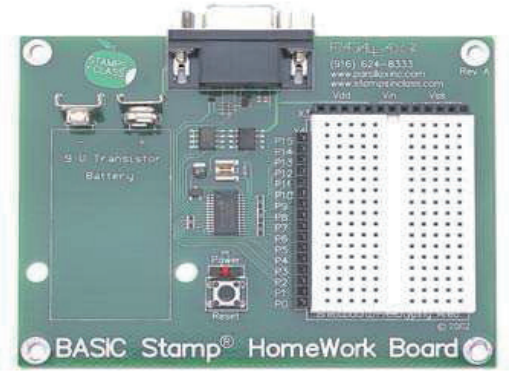


Figure 2.2: BS2 board from Parallax [4]

Basic Stamp 2 (BS2) [4] has a *PIC16c57* microcontroller built on it. It also has a small breadboard where small circuits can be tested. BS2 has a 9 pin serial port ready to communicate with the computer which is used for serial communication and for burning the necessary codes. It has got a 9 volt battery port with it. The microcontroller needs 5 volt to run so necessary circuits must convert 5 volt from 9 volt. The main shortcoming of a BS2 is the limited amount of code or memory it can store at one time i.e. 500 lines of code before its run out of memory.

2.3.2 Sensors

The word 'sensor' is derived from entire meaning 'to perceive' and to perceive the surround environment is exactly what we need for our goal based Robot to achieve its desired goals. Chapter 4 discusses about the sensors used throughout our Thesis.

2.4 Implementation Platform

The simulation software was implemented on Java SE.

2.4.1 Benefits of Java

We used Java SE in our project to build our simulation; though we checked our alternatives like MATLAB and .NET none of them could provide the ease of use as well the adaptability to any environment available. Suppose when installing on the laptop we decided to use Linux so the .NET solution would not be viable anymore (portability unavailable). Java too has a vast library with which we can communicate with the COM port as well as communicate with the Bluetooth enabled debugger.

Benefits of Java for this software [5]

- Object Oriented Programming.
- Ease of Debugging.
- Built-in Operating System Features.
- Software Libraries.
- Readily Available Supply of Related Books, Literature and other Resources.
- Portability.
- Learn Modern Software Development Skills.

2.4.2 JAVA COMM

The Java Communications API is a Java extension that facilitates developing platform-independent communications applications for technologies such as Smart Cards, embedded systems, and point-of-sale devices, financial services devices, fax, modems, display terminals, and robotic equipment [7].

The Java Communications API (also known as javax.comm) provides applications access to RS-232 hardware (serial ports) and limited access to IEEE-1284 (parallel ports), SPP mode.

2.5 Serial Communication

Serial communication is a popular means of transmitting data between a computer and a peripheral device such as a programmable instrument or even another computer. Serial communication uses a transmitter to send data, one bit at a time, over a single communication line to a receiver. You can use this method when data transfer rates are low or you must transfer data over long distances. Serial communication is popular because most computers have one or more serial ports, so no extra hardware is needed other than a cable to connect the instrument to the computer or two computers together [8].

Serial communication requires that you specify the following four parameters:

- The baud rate of the transmission
- The number of data bits encoding a character
- The sense of the optional parity bit
- The number of stop bits

3.1 Simulation Software

Any hardware needs software to work properly. It is the software that drives the hardware in a controlled way. In out implementation controlling the hardware with the software was kept at a minimum, the software was used only to simulate the information sent by the hardware and represent the information in an understandable way. Information from the hardware i.e. the shaft encoder and the compass is processed and plotted visually in a 2D interface. The software itself does not instruct the hardware to do anything except starting its action. The hardware at each and every point provides the software with positioning information such as real time displacement and telemetry data and the software processes this information and represents it visually.

3.2 Different parts of the simulation software

Computer software needs to be divided into parts to decrease its internal complexity and for the sake of simpler development process. Those parts must have their own responsibilities. In addition to that software parts must be interrelated to each other and maintain communication among the parts. The relationship must be well defined and communication must be precise. The simulation software is divided into two main parts. These parts have their individual responsibilities. These parts are listed below:

- Hardware Control
- Visual Representation (GUI)

The relationship among these three parts is shown by the following figure:

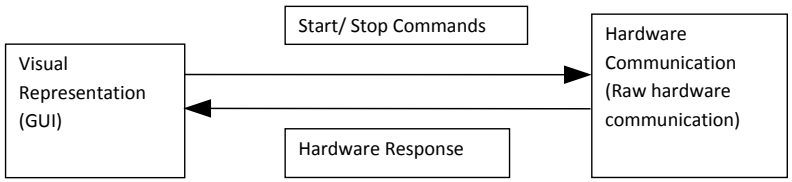


Figure 3.1: Simulating Software parts

From the figure it is understandable that the GUI part gets information about the hardware through Hardware Communication part. Since the software is a simulating one, it itself does not provide the hardware with any commands other that start and stop. The Hardware communication part communicates with the hardware and the GUI providing the GUI with data directly understandable by it.

3.2.1 Hardware Communication

This part deals with all the complexities of the hardware. It communicates with the hardware through COM port. It gets command from the GUI (i.e. Start/Stop) and notifies the hardware communication part which in turn notifies the hardware and vice versa. The primary responsibility of Hardware Communication is to maintain a communication between the computer and the robot's hardware. It also converts the received data from the hardware into understandable format for the GUI part.

3.2.2 Visual Representation (GUI)

The visual representation (GUI) part takes input from the hardware communication part and represents the information in a 2D area. The representation is

updated with every physical movement. The Interface is represented in the diagram below.

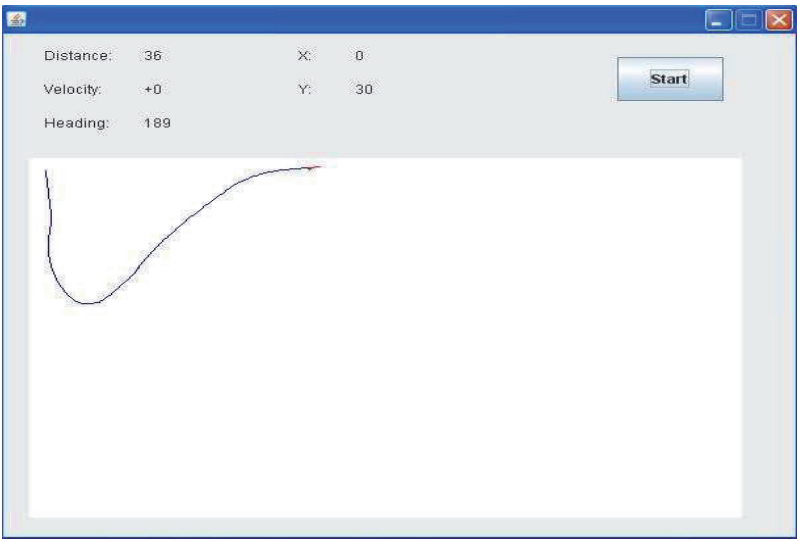


Figure 3.2: The Interface.

The interface represents the path taken by the robot and represents it as shown. Information about total distance moves, current velocity, and clockwise angle with respect to the north is also represented. The interface also represents current position of the robot within the 2D map using its X and Y coordinates.

3.3 Internal 2D Map

The robot must know its current location and direction at any point of time and also the path that the robot has been traveling. For this purpose the simulation software maintains an internal 2D map. The map contains the current position and direction of the robot. This map is viewed in the GUI. The information of map is

always supplied to the Hardware Communication and data processing part to let the user see the current status of the whole arena.

3.3.1 Internal 2D Map- working principle.

The 2D map itself is not intelligent like the robot; rather, the robot provides the 2D map with information about its displacement and orientation (compared with its initial direction).

The logic within the 2D map reacts only when the orientation data received from the robot changes, any displacement of the robot before that is stored within a variable. The displacement occurred in that (previous) direction is then drawn with in the map. This process goes on until the end of line is reached.

The data for the map is solely provided by a dedicated microcontroller whose sole purpose is to detect orientation and displacement of the robot only.

The figure below illustrates the logic-

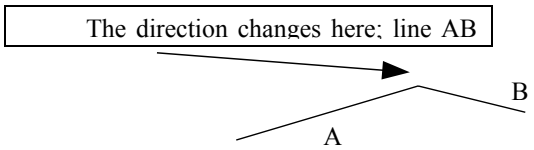


Figure 3.3: 2D map logic.

3.4 Development Language and Design Issues

To make the controlling software platform independent the entire software code is written in JAVA language. NetBeans 6.0 IDE (Integrated Development Environment) is used to manage the development process of the entire controlling software.

For hardware communication JAVA COM API [7] (Application Programming Interface) is used. For GUI JAVA Swing API is used.

3.4.1 Java Comm API

The Java Communications API (also known as javax.comm [7]) provides applications access to RS-232 hardware (serial ports). This API contains all the raw level communication with the serial port. All communications used with our micro controller [13] [14] will use RS-232 serial communication at 1200 baud, eight data bits, one stop bit, and no flow control[15][16].

We used the following classes from the Java Communications API for port access[17].

Java Class	Description
javax.comm.CommPortIdentifer	This is the main class for controlling access to communications ports. It provides a list of communications ports made available by the driver, assists with the opening of Communications ports, and assists with port ownership.
javax.comm.SerialPort	This is the standard RS-232 communications port provided by the comm.jar. Per the Java documentation, this class defines the minimum required functionality for serial communications ports.

3.4.2 Design issues

While designing the software the first problem we faced was how to communicate with the P.C., this issue was resolved by the java comm API. Designing the interface the interactivity required by it was also a problem, the interactivity requirement was slacked afterwards. The main problem we faced with the communication between the P.C. and the microcontroller, the kind of data to be sent and received by the platforms, our requirement of the simulation software was to receive data only so that the robot can perform solely, these issues was addressed by the shaft encoder and compass placed within the robot, these two sensors made the robot totally independent of any command and made the simulation software really a simulating one.

4.1 Sensors Synopsis

A sensor is to a robot like what eyes, ears are to a human. Without it a robot is handicapped. Most of the programming logic for sensors is located inside the microcontroller programs. The reason for this is simple: microcontrollers are fully fledged platform with low power requirements, requires less space thus mobile and is cheaper; these properties makes it feasible to use them dedicatedly to guide sensors. However, using Java programs to initiate and control the “sensing” from these sensors has many advantages

4.2 Infra Red Sensor

Infrared (**IR**) radiation is electromagnetic radiation of a wavelength longer than visible light, but shorter than radio waves. The name means "below red" (from the Latin *infra*, "below"), red being the color of visible light with the longest wavelength. Infrared radiation has wavelengths between about 750nm and 1mm [9].

4.2.1 Infrared Uses:

The uses of infrared include military, such as: target acquisition, surveillance, homing and tracking and non-military, such as thermal efficiency analysis, remote temperature sensing, short-ranged wireless communication, weather forecasting, object detention, distance measuring, etc [9].

4.2.2 IR for Line Following:

We use the infrared for line following. The concept is to generate an infrared wave continuously by an infrared emitter and two receiver wait to receive it.

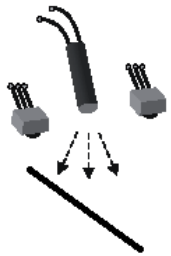


Figure 4.1: Line Following.

The line following code is illustrated below:

```
DO
  HIGH 0
  HIGH 2
  RCTIME 0, 1, time1
  RCTIME 2, 1, time2
  'DEBUG  ? time1
  'DEBUG  ? time2

  IF time1>30 AND time1<50 AND time2>60 AND time2<90 THEN
    GOTO endofline
  ENDIF

' Testing weather sensors are connected correctly if lit the sensors are
needed to be adjusted
  IF time1=0 OR time2=0 THEN
    HIGH 4
  ELSE
    LOW 4
  ENDIF

'if on the line move forward

  IF time1>70 AND time2>120 THEN
    HIGH direction
    PULSOUT 15,500
    PULSOUT 11,1000
```

```
          DEBUG "111111"
          r = -1
          'SHIFTOUT 7,8,MSBFIRST,[1\1]

      'white detected to the right move left
        ELSEIF time1>70 AND time2<120 THEN
        HIGH direction
        PULSOUT 15,500
        DEBUG "133333"
        r = 1
        'SHIFTOUT 7,8,MSBFIRST,[1\1]

      'white detected to the left move right
        ELSEIF time1<70 AND time2>120  THEN
         HIGH direction
         PULSOUT 11,1000
         DEBUG "122222"
         r = 0
         'SHIFTOUT 7,8,MSBFIRST,[1\1]
      'if white is detected by both the sensors decision is made basing on
previous decision
         ELSE

          IF r=0 THEN 'right
            HIGH direction
            PULSOUT 11, 1000
            r=0
            DEBUG "122222"
            'SHIFTOUT 7,8,MSBFIRST,[1\1]
          ELSEIF r=1 THEN   'left
            HIGH direction
            PULSOUT 15, 500
            r=1
            DEBUG "133333"
            'SHIFTOUT 7,8,MSBFIRST,[1\1]
          ELSE
            LOW direction
            'PULSOUT 15,850
            'PULSOUT 11, 650

            PULSOUT 15, 850
            PULSOUT 11, 650
            'SHIFTOUT 7,8,MSBFIRST,[0\1]

            'PULSOUT 15, 1000
            'PULSOUT 11, 500

          ENDIF

      ENDIF


      PAUSE 5
      'DEBUG CLS
      LOOP
```

The infrared emitter is always kept on, infrared light from the surface gets detected by the sensors and the amount of infrared detected is kept in the variables time1 and time2. If a particular sensor is above a black surface the amount of detection is more compared to the amount of detection received when it is over a white surface. In our implementation a black line is drawn over a white surface. If the sensor at the left detects white surface under it, if causes the robot to move right and if the sensor at the right detects it is over white surface it causes the robot to move left. If both the sensors detect that they are over black surface they cause the robot to move straight. In some instances both the sensors might detect that they are over a white surface if such a thing happens decision is taken based on the previous decision and if the previous decision does not result to a solution the robot moves backwards.

4.2.3 Components & Circuits:

IR LED: emits IR signal when driven at IR rate

IR detector: designed to send a "pin low" in the presence of an IR signal

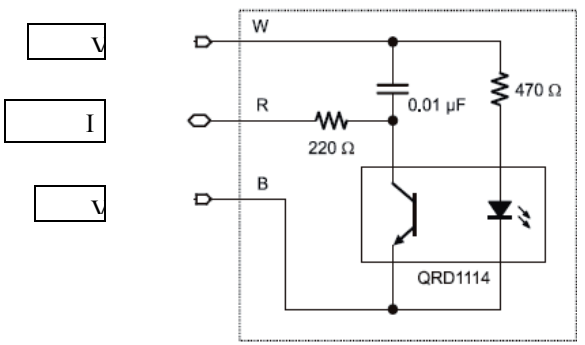


Figure 4.2: Interfacing IR components to the BASIC Stamp

4.2.3.1 Emitter

The emitter is connected directly with the supply i.e. the infrared emitter is always kept on.

4.2.3.2 Detector:

RCTIME command is used with the detector which allows us to understand the color of the surface under the considered detector and make a decision accordingly.

4.4 Electronic Compass

The Hitachi *HM55B* Compass Module is a dual-axis magnetic field sensor that adds a sense of direction to the path finding Robot. The sensing device on the Compass Module is a Hitachi HM55B chip [10]. An onboard regulator and resistor protection make the 3 volt HM55B chip compatible with 5 volt BASIC Stamp microcontroller supply and signal levels.

4.4.1 Working of an Electronic Compass

The Hitachi HM55B Compass Module has two axes, x and y. Each axis reports the strength of the magnetic field's component parallel to it. The x-axis reports (field strength) × cos(), and the y-axis reports the (field strength) × sin(). To resolve into a clockwise angle from north, use arctan(-y/x), which in PBASIC 2.5 is x ATN -y. The ATN command returns the angle in binary radians. To convert to degrees with PBASIC, we just apply */ 360 to the variable storing the binary radian measurement. [11]

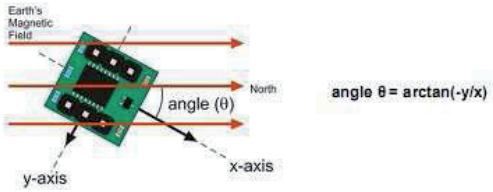


Figure 4.3: Magnetic Field Alignment with the compass [11]

4.4.2 Components & Circuits:

The figure 4.4 shows the Interfacing of the Hitachi *HM55B* compass components to the BASIC Stamp.

While changing the direction of movement (rotating) the compass is used to make it as accurate as possible

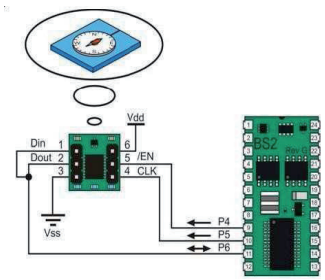


Figure 4.4: Interfacing the Compass [11]

The Figure 5.10 shows the alignment of the magnetic poles and the angle given by the compass. It is noted that 0 degrees is the magnetic North and 180 degrees the magnetic south.

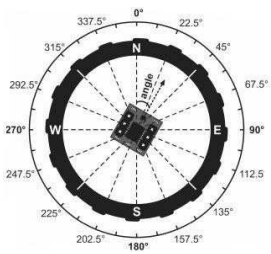


Figure 4.5: Angular representation of the Compass [11]

Snippet of the code used to detect orientation of the robot using compass.

```
DinDout   PIN   15
Clk   PIN       14
En    PIN       13

YOffset   CON     0
XOffset   CON     0

Reset     CON     %0000
Measure   CON     %1000
Report    CON     %1100
Ready   CON       %1100
NegMask   CON     %1111100000000000

x   VAR   Word
y   VAR   Word
status    VAR       Nib
angle   VAR        Word
dat VAR Word

compass:

  GOSUB GET_Compass_Axes

  angle = x ATN y
  angle = angle */ 360

  'DEBUG HOME,
   ' "x-axis N(-S) = ", SDEC x, CLREOL,
    'CR, "y-axis W(-E) = ", SDEC y, CLREOL,
    'CR, CR, DEC3 angle, " Degrees"

  GOTO getdis

GOTO compass


Get_Compass_Axes:

  HIGH En: LOW en
  SHIFTOUT DinDout, clk, MSBFIRST, [Reset\4]

  HIGH en: LOW en
  SHIFTOUT DinDout, clk, MSBFIRST, [Measure\4]
  status = 0

  DO
    HIGH En: LOW En
    SHIFTOUT DinDout, clk, MSBFIRST, [Report\4]
```

```
    SHIFTIN DinDout, clk, MSBPOST, [Status\4]
  LOOP UNTIL status = Ready
  SHIFTIN DinDout, clk, MSBPOST, [x\11, y\11]
  HIGH En

  IF (y.BIT10 = 1) THEN y = y |NegMask
  IF (x.BIT10 = 1) THEN x = x | NegMask

  y = y- YOffset
  x = x- XOffset
RETURN


getdis:

'PAUSE 90
'DEBUG CLS

SHIFTOUT 11,12, MSBFIRST,[128\8,1]
PAUSE 1

SHIFTOUT 11,12,MSBFIRST,[2\8]
SHIFTIN 11,12,MSBPRE,[dat]



'IF dat < 127 THEN
'DEBUG "y+",DEC dat
'ELSE
'dat = dat ^ 255 + 1
'DEBUG "y-",DEC dat
'ENDIF

'DEBUG CR

DEBUG  DEC3 dat
DEBUG DEC3 angle
PAUSE 50

RETURN
```

4.5 Sensor Positioning

The line following sensor arrangement is placed underneath the robot, at the middle front of it. Initially the robot i.e. the line following sensor needs to be placed over the line. The execution of the program causes the robot to follow the line up till

the end of line is reached. The algorithm causes the robot to stop once it reaches the end of line.

Snippet of the code used to cause the robot follow a line.

```
      DO

        HIGH 0
        HIGH 2
        RCTIME 0, 1, time1
        RCTIME 2, 1, time2
        'DEBUG  ? time1
        'DEBUG  ? time2

        'IF time1>30 AND time1<50 AND time2>60 AND time2<90 THEN
         ' GOTO endofline
        'ENDIF

      ' Testing weather sensors are connected correctly if lit the sensors are
needed to be adjusted
        IF time1=0 OR time2=0 THEN
          HIGH 4
        ELSE
          LOW 4
        ENDIF

      'if on the line move forward

        IF time1>70 AND time2>120 THEN
          'HIGH direction
          PULSOUT 15,500
          PULSOUT 11,1000
          'DEBUG "l11111"
          r = -1
          'SHIFTOUT 7,8,MSBFIRST,[1\1]

      'white detected to the right move left
        ELSEIF time1>70 AND time2<120 THEN
        'HIGH direction
        PULSOUT 15,500
        'DEBUG "l33333"
        r = 1
        'SHIFTOUT 7,8,MSBFIRST,[1\1]

      'white detected to the left move right
        ELSEIF time1<70 AND time2>120  THEN
         'HIGH direction
         PULSOUT 11,1000
         'DEBUG "l22222"
         r = 0
```

```
         'SHIFTOUT 7,8,MSBFIRST,[1\1]
      'if white is detected by both the sensors decision is made basing on
previous decision
         ELSE

          IF r=0 THEN 'right
            'HIGH direction
            PULSOUT 11, 1000
            r=0
            'DEBUG "122222"
            'SHIFTOUT 7,8,MSBFIRST,[1\1]
          ELSEIF r=1 THEN   'left
            'HIGH direction
            PULSOUT 15, 500
            r=1
            'DEBUG "133333"
            'SHIFTOUT 7,8,MSBFIRST,[1\1]
          ELSE
            'LOW direction
            'PULSOUT 15,850
            'PULSOUT 11, 650

            PULSOUT 15, 850
            PULSOUT 11, 650
            'SHIFTOUT 7,8,MSBFIRST,[0\1]

            'PULSOUT 15, 1000
            'PULSOUT 11, 500

          ENDIF

      ENDIF


      PAUSE 5
      'DEBUG CLS
      LOOP
```

4.5.1 Positioning and Code Concepts

One emitter and two detectors continuously scan the underneath of the sensor arrangement for the line. At any instant if one of the detectors looses the track of the line the robot moves in the opposite direction, i.e. if the sensor at the left looses the line it causes the robot to turn right and vice versa. If at any instant both the sensors loose the track of the line (which is possible at some instants) the algorithm causes the robot to move backwards until at least one sensor gets a track of the line.

5.1 Shaft Encoders

A shaft encoder is a component which translates the rotational movement of a shaft into an electrical wave form. Shaft encoders provide motion detection because they translate the rotary motion of a shaft into either a two or a three channel output (analog or digital). The encoder feedback signal is sent to the system that controls the speed, position, and the direction of rotation of the shaft. Shaft encoders are used in many closed-loop servo applications. Typical applications include printers, plotters, tape drive, positioning tables, automatic handlers, robotics, factory automation, medical equipment and high quality instrumentation. [12]

5.2 Encoder Classification

Rotary encoders can be classified in two ways: by detecting method and output signal. Categories for the detecting method include contact, optical, magnetic, and laser encoders, while for those for the output signal are incremental and absolute types.[12]

5.2.1 Absolute Encoders

Absolute encoder provides a binary "word" for each position. Each bit requires a separate optical channel. The resolution is equal to the number of output bits. Absolute encoders are rather complex and expensive products which constantly retain the correct position for one revolution. The main advantage is that the output signal is not af-

fected by a power shut-off. Absolute encoders are mainly used in robotic tools. Micro-Drives do not offer absolute encoders, including resolvers.[12]

5.2.1.1 Absolute rotary encoder

The absolute digital type produces a unique digital code for each distinct angle of the shaft. A metal sheet cut into a complex pattern is affixed to an insulating disc, which is rigidly fixed to the shaft. A row of sliding contacts is fixed to a stationary object so that each contact wipes against the metal sheet at a different distance from the shaft. As the disc rotates with the shaft, some of the contacts touch metal, while others fall in the gaps where the metal has been cut out. The metal sheet is connected to a source of electric current, and each contact is connected to a separate electrical sensor. The metal pattern is designed so that each possible position of the axle creates a unique binary code in which some of the contacts are connected to the current source (i.e. switched on) and others are not (i.e. switched off).

This code can be read by a controlling device, such as a microprocessor, to determine the angle of the shaft.

The absolute analog type produces a unique dual analog code that can be translated into an absolute angle of the shaft (by using a special algorithm). [13]

Figure 5.1: Absolute rotary encoder [13]

5.2.1.2 Standard binary encoding

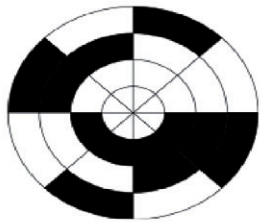


Figure 5.2: 3-bit Standard binary encoder [13]

Rotary encoder for angle-measuring devices marked in 3-bit binary. The inner ring corresponds to Contact 1 in the table. Black sectors are "on". Zero degrees is on the right-hand side, with angle increasing anticlockwise.

An example of a binary code, in an extremely simplified encoder with only three contacts, is shown below.[13]

Sector	Contact 1	Contact 1	Contact 1	Contact 1
1	Off	Off	Off	0° to 45°
2	Off	Off	On	45° to 90°
3	Off	On	Off	90° to 135°
4	Off	On	On	135° to 180°
5	On	Off	Off	180° to 225°
6	On	Off	On	225° to 270°
7	On	On	Off	270° to 315°
8	On	On	On	315° to 360°

Table 5.1: 3-Bit Binary Encoder.[13]

In general, where there are n contacts, the number of distinct positions of the shaft is 2n. In this example, n is 3, so there are 23 or 8 positions.

In the above example, the contacts produce a standard binary count as the disc rotates. However, this has the drawback that if the disc stops between two adjacent sectors, or the contacts are not perfectly aligned, it can be impossible to determine the angle of the shaft. To illustrate this problem, consider what happens when the shaft angle changes from 179.9° to 180.1° (from sector 4 to sector 5). At some instant, according to the above table, the contact pattern will change from off-on-on to on-off-off. However, this is not what happens in reality. In a practical device, the contacts are never perfectly aligned, and so each one will switch at a different moment. If contact 1 switches first, followed by contact 3 and then contact 2, for example, the actual sequence of codes will be

Off-on-on (starting position)

On-on-on (first, contact 1 switches on)

On-on-off (next, contact 3 switches off)

On-off-off (finally, contact 2 switches off)

Now look at the sectors corresponding to these codes in the table. In order, they are 4, 8, 7 and then 5. So, from the sequence of codes produced, the shaft appears to have jumped from sector 4 to sector 8, and then gone backwards to sector 7, then backwards again to sector 5, which is where we expected to find it. In many situations, this behavior is undesirable and could cause the system to fail. For example, if the encoder were used in a robot arm, the controller would think that the arm was in the wrong position, and try to correct the error by turning it through 180°, perhaps causing damage to the arm. [13]

5.2.1.3 Gray encoding

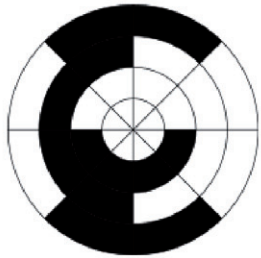


Figure 5.3: 3-bit Gray coding. [13]

Rotary encoder for angle-measuring devices marked in 3-bit binary-reflected Gray code (BRGC). The inner ring corresponds to Contact 1 in the table. Black sectors are "on". Zero degrees is on the right-hand side, with angle increasing anticlockwise.

To avoid the above problem, Gray encoding used. This is a system of binary counting in which two adjacent codes differ in only one position. For the three-contact example given above, the Gray-coded version would be as follows.[13]

Sector	Contact 1	Contact 1	Contact 1	Contact 1
1	Off	Off	Off	0° to 45°
2	Off	Off	On	45° to 90°
3	Off	On	Off	90° to 135°
4	Off	On	On	135° to 180°
5	On	Off	Off	180° to 225°
6	On	Off	On	225° to 270°
7	On	On	Off	270° to 315°
8	On	On	On	315° to 360°

Table 5.2: 3-Bit Gray Code Encoder.[13]

In this example, the transition from sector 4 to sector 5, like all other transitions, involves only one of the contacts changing its state from on to off or vice versa. This means that the sequence of incorrect codes shown in the previous illustration cannot happen here.

However, as far as most people are concerned, all that needs to be known is that Gray code is more secure than binary in encoder applications.[13]

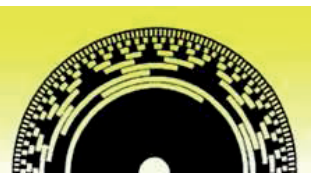


Figure 5.4: Gray code encoder.[14]

Absolute encoders fall into two groups according to their resolutions: encoders with a resolution of up to 360 degrees are called single turn encoders and those which can revolve many times (usually up to 4096 turns) are called multi-turn encoders. Normally a single turn encoder gives a 12 bit parallel output (4.096 different codes per revolution of a coded disc) while a multi-turn encoder uses the same single turn disc as the single turn encoder but it is coupled to a second disc via a 16: 1 reduction gearbox so that the second coded disc revolves once for every 16 turns of the first disc.

The second disc gives 4 bits output and if only two coded discs were used. The total 16 bits gives 65.536 different codes for 16 complete revolutions. ASM use a 24 bit absolute encoder which has 4 coded discs.

The electronic interface SSI or Synchronous Serial Interface was designed for use with absolute encoders. Generally, absolute encoders give a parallel output in

Gray code which allows only a one bit change between adjacent resolvable steps and reduces the maximum reading error to one step.[14]

This is all very desirable but if the parallel output was used it would mean at least 24 wires would be required to read the output. SSI however offers a much better solution whereby the parallel output is set into a shift register (parallel to serial converter) and then shifted out serially by pulses supplied by an external interface (SSI).[14]

The speed of the transmitted output is therefore dependent on the frequency of the pulses supplied so if fast update is required, the speed of the pulses would need to be adjusted accordingly.

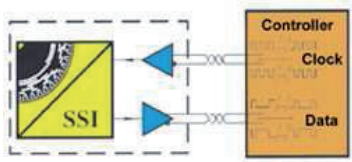


Figure 5.5: Microcontroller and Gray code encoder.[14]

5.2.1.4 Incremental Encoders

Incremental encoders provide a pulse for each increment of shaft movement. Usually this consists of two optical channels to enable the determination of the direction of rotation. The incremental encoder has a lower cost than the absolute encoder due to the limited number of channels is more reliable, and the encoded position is not limited to one revolution.[12]

Basic Operation of Optical Rotary Incremental Encoders Optical rotary incremental encoders consist of five main components:

- LED light source
- Rotating code disk
- Stationary mask
- Photo detector(s)
- Amplifying/squaring electronics

As the code disk rotates in front of the stationary mask, it shutters light from the LED. The light that passes through the mask is received by the photo detector, which produces pulses

in the form of a quasi-sine wave. The encoder electronics convert the sine wave into a square signal, ready for transmission to a counter.[15]

5.2.1.4.1 Conventional Code Disks

Conventional incremental code disks contain a fixed number of equally spaced opaque lines that produce a corresponding number of pulses per revolution. Each line count requires a unique code disk. The position and spacing of the lines on the disk requires a high degree of precision. Physical limitations determine the maximum number of lines that can be created on a code disk of a given size.[16]

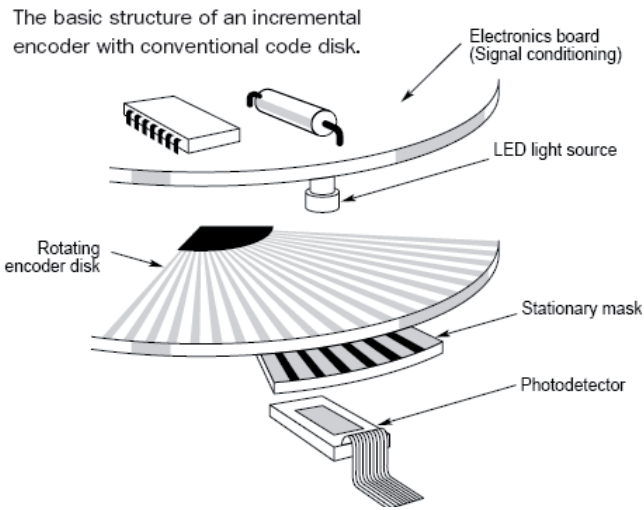


Figure 5.6: Basic structure of an incremental encoder.[16]

5.2.1.4.2 Tachometer Encoders

A single channel (e.g. A) incremental encoder, or tachometer, is used in systems that operate in only one direction and require simple velocity information. Velocity can be determined from the time interval between pulses, or by the number of pulses within a given time period. [16]

Incremental Encoders- Incremental rotary encoders are also known as quadrature encoders.

An incremental rotary encoder (also called relative rotary encoders) has two outputs called quadrature outputs. They can be either mechanical or optical. In the optical type there are two gray coded tracks, while the mechanical type has two contacts that are actuated by cams on the rotating shaft. The mechanical type requires debouncing and is typically used as digital potentiometers on equipment including consumer

devices. Most modern home and car stereos use mechanical rotary encoders for volume. Due to the fact the mechanical switches require debouncing, the mechanical type are limited in the rotational speeds they can handle. The incremental rotary encoder is the most widely used of all rotary encoders.[17]

The optical type is used when higher RPMs are encountered or a higher degree of precision is required.Incremental rotary encoders are used to track motion and can be used to determine position and velocity. This can be either linear or rotary motion. Because the direction can be determined, very accurate measurements can be made. They employ two outputs called A & B which are called quadrature outputs as they are 90 degrees out of phase. The state diagram:[17].

Gray Coding for Clockwise Rotation

Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Table 5.3: Gray Code for Clockwise Rotation.[17]

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

Table 5.4: Gray Code for Clockwise Rotation.[17]

If you were to draw this as a wave form, you would see that they are 90 degrees out of phase, which is all that the quadrature term means. These signals are decoded to

produce a count up pulse or a countdown pulse. For decoding in software, the A & B outputs are read by software, either via an interrupt on any edge or polling, and the above table is used to decode the direction. For example if the last value was 00 and the current value is 01, the device has moved one half step in the clockwise direction. The mechanical types would be debounced first by requiring that the same (valid) value be read a certain number of times before recognizing a state change.

Rotary sensors that have a single output are not encoders and cannot determine direction, but can sense RPM.

This same principle is used in old ball mouse to track whether mouse is moving to the right/left or forward/backwards

Optical Encoders – Consists of Light emitting elements (LED) and light receiving elements.

5.2.1.5 Magnetic Encoders

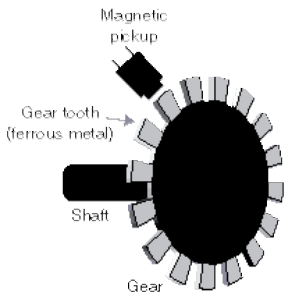


Figure 5.7: Basic structure of a magnetic encoder.[18]

A magnetic encoder consists of a rotating gear made of ferrous metal and a magnetic pick-up that contains a permanent magnet and the sensing element. The gear, which is mounted on the rotating shaft, has precisely machined teeth that provide the code pat-

tern. As the disk rotates, these teeth disturb the magnetic flux emitted by the permanent magnet, causing the flux field to expand and collapse. These changes in the field are sensed by the sensing element, which generates a corresponding digital or pulse signal output.

Two kinds of magnetic pick-ups exist:

- **Hall effect** -- pick-ups use a semiconducting sensing element that relies on the Hall effect to generate a pulse for every gear tooth that passes the pickup.
- **Variable reluctance** -- pick-ups use a simple coil of wire in the magnetic field. As the gear teeth pass by the pick-up and disturb the flux, they cause a change in the reluctance of the gear/magnet system. This induces a voltage pulse in the sensing coil that is proportional to the rate flux change.[18]

5.3 Shaft Encoder Using ADNS-2610 Chip.

Description: The ADNS-2610 is a new entry level, small form factor optical mouse sensor. It is used to implement a no mechanical tracking engine for computer mice. Unlike its predecessor, this new optical mouse sensor allows for more compact and affordable optical mice designs. It is based on optical navigation technology, which measures changes in position by optically acquiring sequential surface images (frames) and mathematically determining the direction and magnitude of movement. The sensor is housed in an 8-pin staggered dual inline package (DIP). It is designed for use with the HDNS- 2100 Lens, HLMP-ED80-xx000, and the HDNS-2200 LED Clip, providing an optical mouse solution that is compact and affordable. There are no moving parts, so precision optical alignment is not required, thereby facilitating high volume assembly. The output format is a two wire serial port. The current X and

Y information are available in registers accessed via the serial port. Resolution is 400 counts per inch (c.p.i) with rates of motion up to 12 inches per second (i.p.s).[19]

Theory of Operation: The ADNS-2610 is based on Optical Navigation Technology. It contains an Image Acquisition System (IAS), a Digital Signal Processor (DSP) and a two wire serial port. The IAS acquires microscopic surface images via the lens and illumination system provided by the HDNS- 2100, HDNS-2200, and HLMP-ED80-xx000. These images are processed by the DSP to determine the direction and distance of motion.[19]

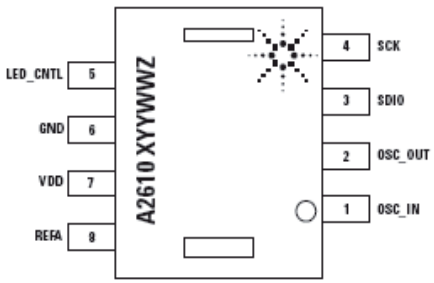


Figure 5.8: Mechanical Drawing (Top View)[19]

Pin Number	Pin	Description
1	OSC_IN	Oscillator input
2	OSC_OUT	Oscillator output
3	SDIO	Serial data (input and output)
4	SCK	Serial port clock (Input)
5	LED_CNTL	Digital Shutter Signal Out
6	GND	System Ground
7	VDD	5V DC Input
8	REFA	Internal reference

Table 5.4: Pin out of ADNS-2610 Optical Sensor [19]

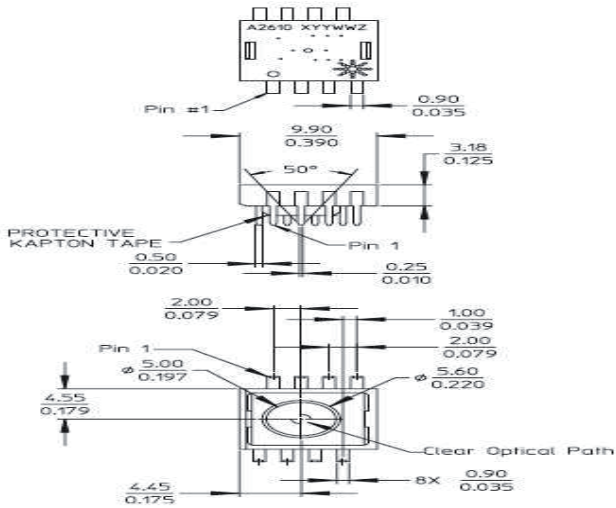


Figure 5.9: Package Outline Drawing [19]

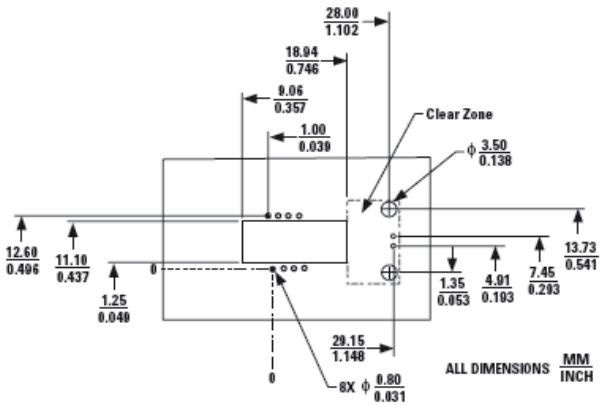


Figure 5.10: Recommended P.C.B. mechanical cutouts and spacing. [19]

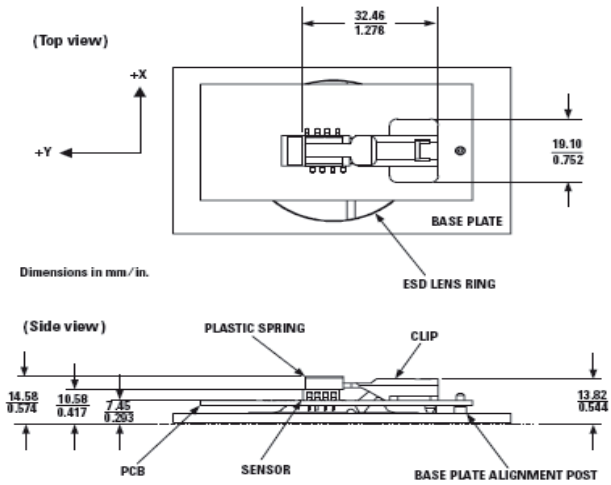


Figure 5.11: 2D assembly drawing of ADNS-2610 shown with the HLMP-ED80 (top and side view). [19]

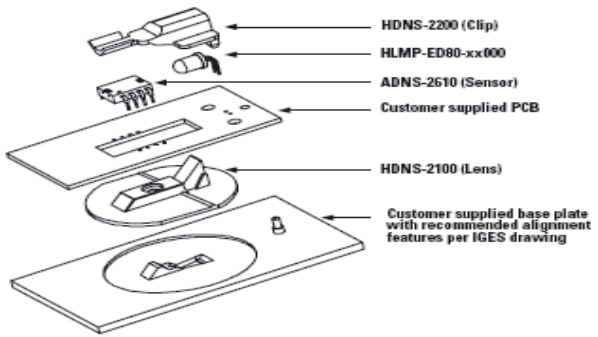


Figure 5.12: Exploded view drawing. [19]

5.3.1 Overview of Optical Mouse

Sensor Assembly:

NOTE: Pin 1 of optical mouse sensor should be inserted into the reference point of mechanical cutouts.

Figures 3 and 4 are shown with HDNS-2100, HDNS-2200 and HLMP-ED80-xx000.

The components shown in Figure 5 interlock as they are mounted onto defined features on the base plate.

The ADNS-2610 sensor is designed for mounting on a through hole PCB, looking down. There is an aperture stop and features on the package that align to the lens.

The HDNS-2100 lens provides optics for the imaging of the surface as well as illumination of the surface at the optimum angle. Features on the lens align it to the sensor, base plate, and clip with the LED. The lens also has a large round flange to provide a long creep age path for any ESD events that occur at the opening of the base plate.

The HDNS-2200 clip holds the LED in relation to the lens. The LED's leads must be formed first before inserting into the clip. Then, both LED and clip is loaded on the PCB. The clip interlocks the sensor to the lens, and through the lens to the alignment features on the base plate.

The HLMP-ED80-xx000 is recommended for illumination. If used with the bin table (as shown in Figure 8), sufficient illumination can be guaranteed. [19]

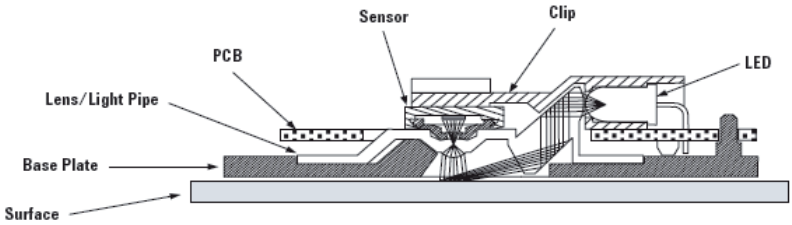


Figure 5.13: Sectional view of PCB assembly highlighting optical mouse components (optical mouse sensor, clip, lens, LED, PCB and base plate). [19]

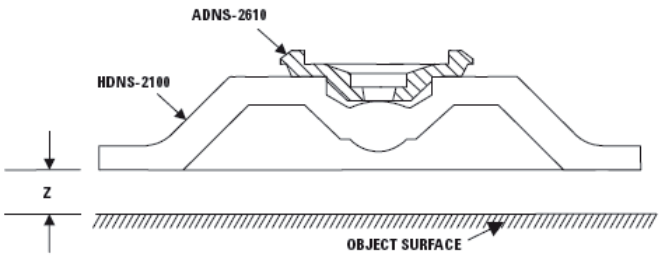


Figure 5.14: Distance from lens reference plane to surface. [19]

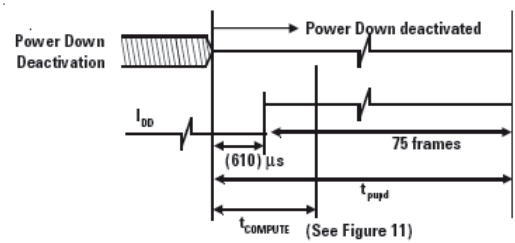


Figure 5.15: Power up timing mode. [19]

Synchronous Serial Port: The synchronous serial port is used to set and read parameters in the ADNS-2610, and also to read out the motion information.

The port is a two wire, half duplex port. The host microcontroller always initiates communication; the ADNS-2610 never initiates data transfers.[19]

SCK: The serial port clock. It is always generated by the master (the microcontroller).[19]

SDIO: The data line.[19]

Write Operation: Write operations, where data is going from the microcontroller to the ADNS-2610, is always initiated by the microcontroller and consists of two bytes. The first byte contains the address (seven bits) and has a "1" as its MSB to indicate data direction. The second byte contains the data.[19]

The transfer is synchronized by **SCK**. The microcontroller changes **SDIO** on falling edges of **SCK**. The ADNS-2610 reads **SDIO** on rising edges of **SCK.[19]**

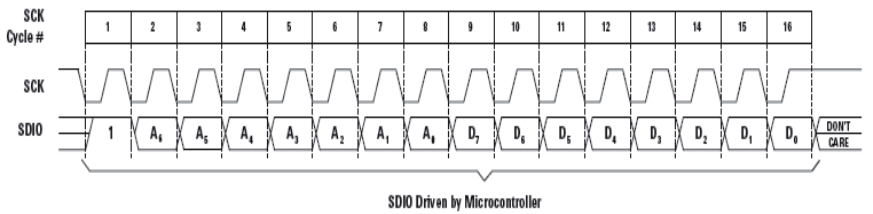


Figure 5.16: Write operation.[19]

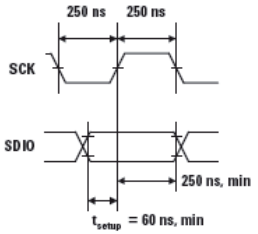


Figure 5.17: SDIO setup and hold times SCK pulse width. [19]

Read Operation: A read operation, meaning data that is going from the ADNS-2610 to the microcontroller, is always initiated by the microcontroller and consists of two bytes. The first byte that contains the address is written by the microcontroller and has a "0" as its MSB to indicate data direction. The second byte contains the data and is driven by the ADNS-2610. The transfer is synchronized by **SCK**. **SDIO** is changed on falling edges of **SCK** and read on every rising edge of **SCK**. The microcontroller must go to a High-Z state after the last address data bit. The ADNS-2610 will go to the High-Z state after the last data bit. Another thing to note during a read operation; SCK needs to be delayed after the last address data bit to ensure that the ADNS-2610 has at least 100 μs to prepare the requested data. This is shown in the timing diagrams below (See Figures 21 to 23). [19]

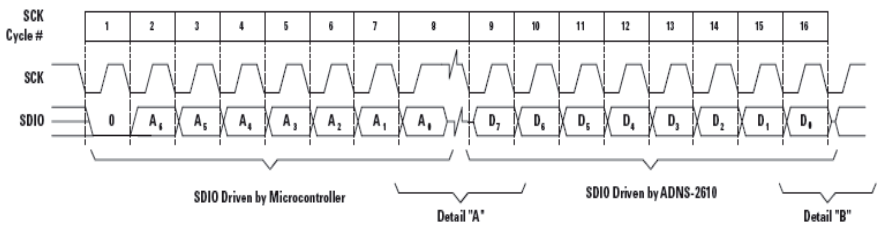


Figure 5.18: Read operation. [19]

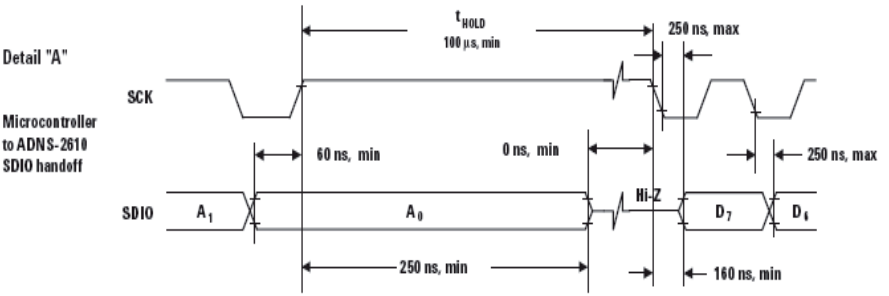


Figure 5.19: Microcontroller to ADNS-2610 SDIO handoff. [19]

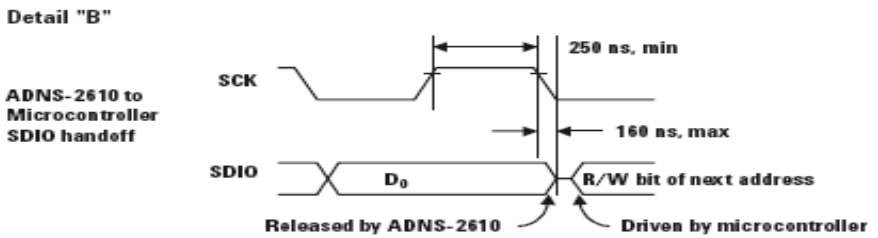


Figure 5.20: ADNS-2610 to microcontroller SDIO handoff. [19]

NOTE:

The 250 ns high state of SCK is the minimum data hold time of the ADNS-2610. Since the falling edge of SCK is actually the start of the next read or write command, the ADNS-2610 will hold the state of D0 on the SDIO line until the falling edge of SCK. In both write and read operations, SCK is driven by the microcontroller. [19]

Required Timing between Read and Write Commands (tsxx): There are minimum timing requirements between read and write commands on the serial port.

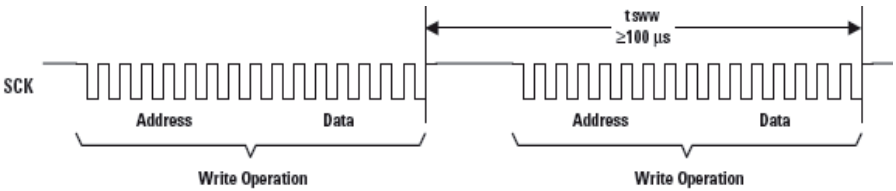


Figure 5.21: Timing between two write commands. [19]

If the rising edge of the SCK for the last data bit of the second write command occurs before the 100 microsecond required delay, then the first write command may not complete correctly.

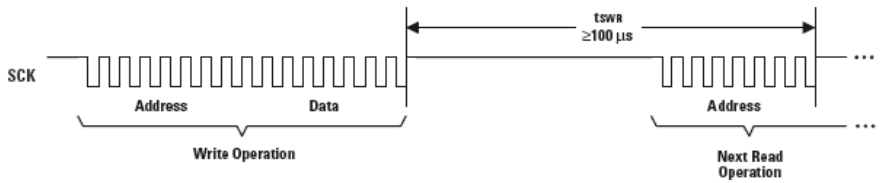


Figure 5.22: Timing between write and read commands. [19]

If the rising edge of SCK for the last address bit of the read command occurs before the 100 microsecond required delay, then the write command may not complete correctly.

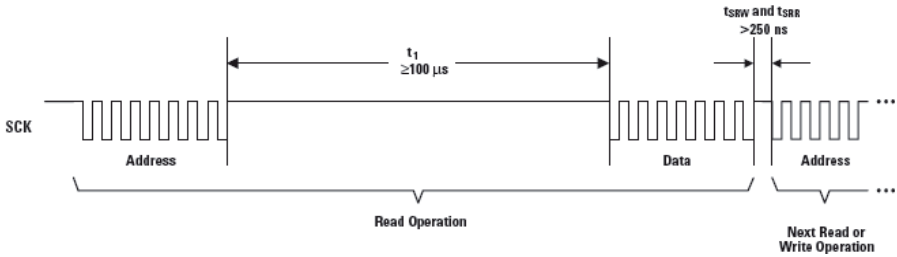


Figure 5.23: Timing between read and either write or subsequent read commands. [19]

The falling edge of SCK for the first address bit of either the read or write command must be at least 250ns after the last SCK rising edge of the last data bit of the previous read operation. [19]

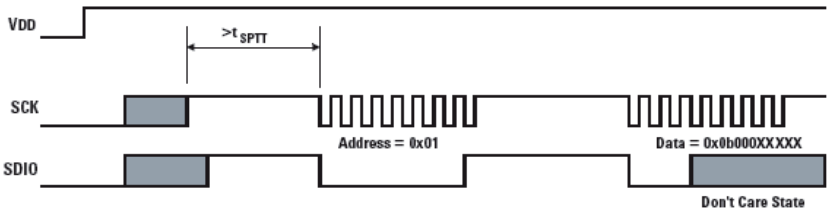


Figure 5.24: Soft reset serial port timer sequence. [19]

Serial Port Timer Timeout

Soft Reset

ADNS-2610 may also be given the reset command at any time via the serial I/O port.

The proper way to perform soft reset on ADNS-2610 is:

1. The microcontroller starts the transaction by sending a write operation containing the address of the configuration register and the data value of 0x80. Since the reset bit is set, ADNS-2610 will reset and any other bits written into the configuration register at this time is properly written into the Configuration Register. After the chip has been reset, very quickly, the ADNS-2610 will clear the reset bit so there is no need for the microcontroller to re-write the Configuration Register to reset it.

2. The digital section is now ready to go. It takes 3 frames for the analog section to settle. [19]

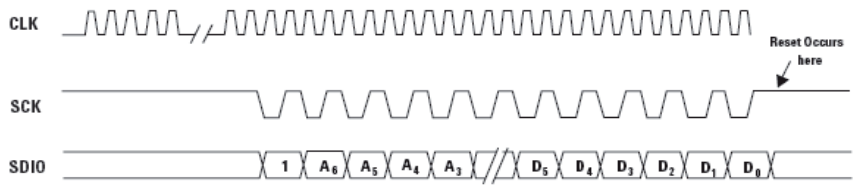


Figure 5.25: ADNS-2610 soft reset sequence timing. [19]

Soft reset will occur when writing 0x80 to the configuration register.

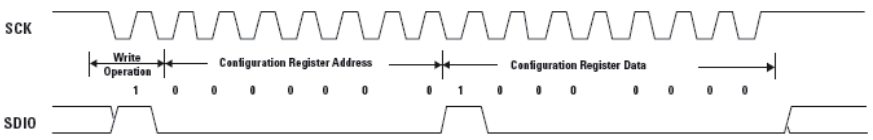


Figure 5.26: Soft reset configuration register writing operation. [19]

Programming Guide Registers:

The ADNS-2610 can be programmed through registers, via the serial port, and configuration and motion data can be read from these registers.

Register	**Address**	**Notes**
Configuration	0x00	Reset, Power Down, Forced Awake, etc
Status	0x01	Product ID, Mouse state of Asleep or Awake
Delta Y	0x02	Y Movement
Delta X	0x03	X Movement
SQUAL	0x04	Measure of the number of features visible by the sensor
Maximum Pixel	0x05	
Minimum Pixel	0x06	
Pixel Sum	0x07	
Pixel Data	0x08	Actual picture of surface
Shutter Upper	0x09	
Shutter Lower	0x0A	
Inverse Product	0x11	Inverse Product ID

Table 5.5: Registers types and address [19]

Configuration Access: Read/Write
Address:0x00 Reset Value:0x00

Bit	7	6	5	4	3	2	1	0
Field	C7	C6	C5	C4	C3	C2	C1	C0

Table 5.6: Configuration register. [19]

Data Type: Bit field

USAGE: The Configuration register allows the user to change the configuration of the sensor. Shown below are the bits, their default values, and optional values.

Field Name	Description
C 7	7 Reset 0 =No effect 1 =Reset the part
C 6	6 Power down 0 =Normal operation 1 =power down all analog circuitry
C 5 – C 1	Reserved
C 0	Forced Awake Mode 0 =Normal, fall asleep after one second of no movement (1500 frames/s) 1 =Always awake

Table 5.7: Configuration register bits. [19]

Status Address: 0x01

Access: Read Reset Value: 0x01

Bit	7	6	5	4	3	2	1	0
Field	ID 2	ID 1	ID 0	Reserved	Reserved	Reserved	Reserved	Awake

Table 5.8: Status register. [19]

Data Type: Bit Field

USAGE: Status information and type of mouse sensor, current state of the mouse.

Field Name	Description
ID 2 -ID 0	Product ID (000 for ADNS-2610)
Reserved	Reserved for future
Awake	Mouse State 0 =Asleep 1 =Awake

Table 5.9: Status register bits. [19]

Delta Y Address: 0x02

Access: Read Reset Value: 0x00

Bit	7	6	5	4	3	2	1	0
Field	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Table 5.10: Delta Y register. [19]

Data Type: Eight bit 2 ’s complement number.

USAGE: Y movement is counted since last report. Absolute value is determined by resolution. Reading clears the register.

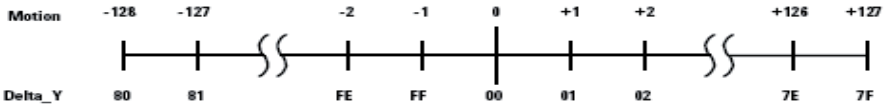


Figure 5.27: Ranges of value of Delta Y.

Delta X Address: 0x03

Access: Read Reset Value: 0x00

Bit	7	6	5	4	3	2	1	0
Field	X7	X6	X5	X4	X3	X2	X1	X0

Table 5.11: Delta X register. [19]

Data Type: Eight bit 2 ’s complement number.

USAGE: X movement is counted since last report. Absolute value is determined by resolution. Reading clears the register.

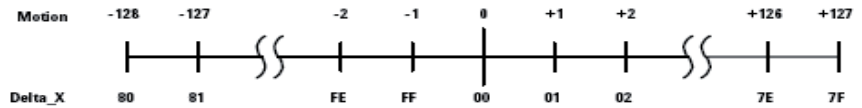


Figure 5.28: Ranges of value of Delta X.

5.4 Implementation of encoders within this project

5.4.1 Optical incremental encoder.

These concepts and the requirement of the project forced us to build an optical rotary encoder from the scratch. Initially binary code was used, he figure below illustrates;

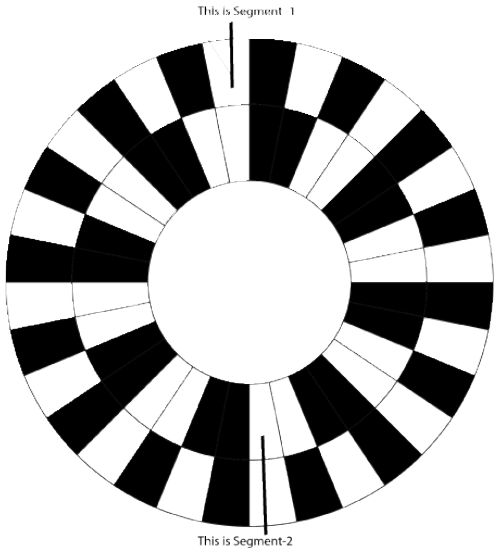


Figure 5.29: A 2 Bit binary encoder disk.

The use of two bit binary code solution was the initial obvious one, but it resulted into catastrophe. At any instant the distance moved and the direction of motion is determined by the changes of bits in segment-1 and segment-2. The table below illustrates:

Here Black is considered to be 0 and white is considered to be 1.

Segment-1	Segment-2	Result
Black	Black	00
Black	White	01
White	Black	10
White	White	11

Table 5.4: 2-Bit Binary Code.

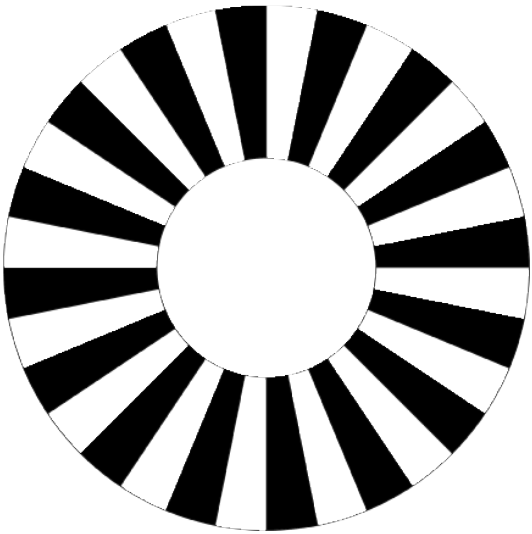


Figure 5.30: A 1 Bit binary encoder disk.

At any instant changing from black-black to black-white (i.e. 00 to 01) is considered one step forward (about 0.5 cm) forward and the reverse is considered as 0.5 cm backward. The order continues every four steps i.e. 00-01 is 0.5 cm forward, 01-10 is 0.5 cm forward, 10-11 is 0.5 cm forward, and 11-00 is 0.5 cm forward similarly 01-00 is 0.5 cm backward, 10-01 is 0.5 cm backward, 11-10 is 0.5 cm backward, and 00-11 is 0.5 cm backward. The problem with this solution is already illustrated in Standard binary encoding section. The problem tends to be unsolvable with the low Tec sensors being used. The solution was gray code encoding but it had to be implemented in an intelligent manner. The gray code encoder had codes at two sides of the disk similar to the binary encoder. The figure illustrates:

One bit binary was used at each side of the disk; the binary codes were 90° out of phase, this solution worked like gray code. Every time a single change occurred, as a result the problem with simultaneous changes was totally gone. The arrangement is shown in the figure below:

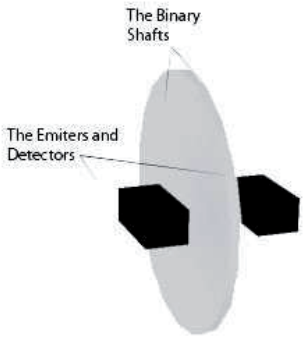


Figure 5.31: Arrangement of the binary disks and the sensors.

Two pairs of emitters and detectors are used in the total arrangement. A pair of detectors detects emission of infrared light from the surface of the shaft and correspondingly determines the current state of the shaft, i.e. if the detector is over a black strip it responds with a “0” and if the detector is over a white strip it responds with a 1.

Since the two shafts are 90° out of phase this arrangement works as a two bit gray code encoder, the next figure illustrates the working principle:

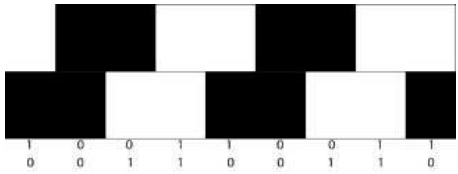


Figure 5.32: Illustration of the gray code.

The changes illustrates the direction and magnitude of distance moved.

In this arrangement also Black is considered to be 0 and white is considered to be 1. At any instant a single change occurs, for example “10” to “00” is considered one step forward (i.e. 0.5cm forward), again “00” to “01” is considered one step forward (i.e. 0.5cm forward), and

“01” to “11” is considered one step forward (i.e. 0.5cm forward), this cycle continues. For backward motion the reverse is true, i.e. “00” to “10” is considered one step forward (i.e. 0.5cm forward), again “01” to “00” is considered one step forward (i.e. 0.5cm forward), and “11” to “01” is considered one step forward (i.e. 0.5cm forward), this cycle continues. This arrangement works flawlessly with a precision of 0.5cm, enough for the ongoing project.

Another implementation of the shaft encoder has been also done using ADNS-2610 optical navigation chip. This chip works flawlessly to produce different types of movement, velocity as well as acceleration related data in a relatively low cost. The precision of this arrangement was almost absolute.

6.1 Locomotion

Locomotion is the ability to move from place to place. While adding this feature to the robot we worry about two things beforehand.

- Mobility
- Localization

Mobility captures the physical mechanics of the vehicle, the interaction of the vehicle with the terrain and the effect of control of the vehicle on the terrain. The maneuver ability of a mobile robot is the combination of the mobility available based on the sliding constraints plus additional freedom contributed by steering.

Localization or position determination provides estimates of the location, altitude, velocity and acceleration of the vehicle. This ability can often precede independently from the system components.

Our wheeled mobile robot consists of a structure supported by two wheels, one or more of which can be steered or driven at a given time. For modeling path traversal, only the constraints imposed by the wheels are important. There is a castor wheel that helps it pivot better. Basically the motion of the robot is limited to

- Move forward.
- Move right.
- Move left.
- Move back.

We found that that using these basic movements we could move in every possible way by performing a series of steps of the above-mentioned steps. Though the maneuver ability of the robot can be evolved as needed the requirement of our project did not demand that much maneuver ability.

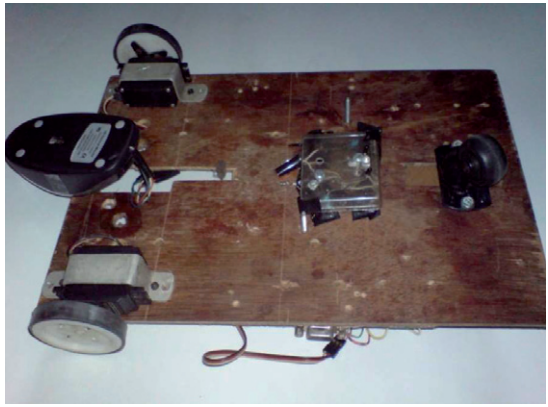


Figure 6.1: Wheel arrangement in the Robot

Both the side wheels are connected to two separate servo Motors.

6.2 Servo Motors

The robot is fitted with two continuous rotation servo motors [20] made by Parallax. Unlike normal servo's these have an angle of rotation of 360 degrees which makes it ideal for usage as a geared motor [21].

A little bit about servo motors:

- Servo Motors are activated by a series of pulses.
- The series of pulses are very brief high signals 1ms to 2ms in duration and about 20ms apart.

- Series of pulses is called a *TRAIN* __|¯¯|_____20ms_____|¯¯|
- At this rate of (2ms + 20 ms)= 22ms, Ref: (22/1000)=45 pulses per second
- 45 pulses will be sent to the servo every second.
- Mostly servos have 3 wires as shown in Fig.
    1. 1 Positive
    2. 1 Negative
3. 1 Pulse Input

Figure 6.2: Continuous Servo Motor by Parallax

Controlling Servo is determined by each of the following: 1.Where to Go - Length of Pulse will send the servo motor to a position.

2. Repetition for sending the pulse determines how long the servo motor will hold that position.

6.2.1 Components & Circuits

The servos are connected to a BS2 board in a manner similar to the figure shown below

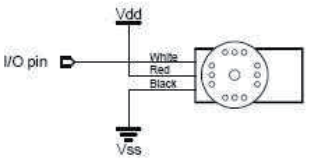


Figure 6.3: Pin configuration of a Servo[23]

As already mentioned we needed to have 4 types of movement in our code

Snippets of Code used to control the servo motors so as to move forward, right, left, and backward.

```
'FORWARD
 PULSOUT 11, 1000
 PULSOUT 15,500

'RIGHT
 PULSOUT 15,500

'LEFT
 PULSOUT 11, 1000

'BACKWORD
 PULSOUT 11,500
 PULSOUT 15, 1000
```

7.1 Discussion

In this thesis work we have developed software to demonstrate the navigation system. The ultimate goal of the navigation system is to plot the accurate movement of the robot. The thesis also included a simple way for the robot to follow a line from a starting point to an ending point.

7.2 Problems & Limitations

Even though we tried our best to fix as many issues as possible within the robot it still has many issues, which are listed below.

- The grid size is fixed but in real life there is no such grid system. The movement should be more arbitrary.

- Use more accurate and powerful motors i.e. stepper motors as they provide better results in terms of accuracy and to carry the onboard laptop.

- Still needs to have a laptop on board of the robot to do the processing.

- We didn't incorporate any communications among the two microcontroller even though we had planned to do, out implementation did

not demand or need the communication between the two microcontrollers, all the data was sent to the main platform i.e. the pc which alone acted as a communicating platform for the microcontrollers. One microcontroller was meant to be used solely for localization calculations.

7.3 Future work

There is much more work that can be done to improve on this project. First of all the whole guidance and roaming system needs to incorporate the randomness of the environment surrounding the robot hence a new navigational system needs to be implemented but at the same time out goal reaching algorithm can be incorporated into it.

Mobile 2D mapping can be implemented to view a graphical representation of the robot movement.

More improvement can be incorporated into the tweaking of the sensors such they perform more accurately.

Real time movement simulation is possible by using more a master slave based micro controller system. The localization incorporated can also be used to incorporate with the simulator. The vision needs to be incorporated to the working of the whole system.

Bibliography

[1] Wikipedia. “Intelligent agent”. Available at http://en.wikipedia.org/wiki/Intelligent_agent, last accessed on December 8, 2007

[2] Wikipedia. “Robotics and Robots”. Available at http://en.wikipedia.org/wiki/Robot, last accessed on December 10, 2007

[3] Wikipedia. “Microcontrollers”. Available at http://en.wikipedia.org/wiki/Microcontroller, last accessed on December 15, 2007

[4] Parallax Inc. “What’s a microcontroller Student Guide”, Available at http://www.parallax.com/dl/docs/books/edu/wamv2_2.pdf, last accessed on December 10, 2007

[5] RidgeSoft. “Benefits of Java”. Available at http://www.ridgesoft.com/javabenefits.htm, last accessed on December 15 2007

[6] Klingsheim, André. “J2ME Bluetooth Programming.” Master’s Thesis, Submitted to University of Bergon, 2004

[7] Sun.com. “The Java Communications API”. Available http://java.sun.com/products/javacomm/, last accessed on December 12 2007

[8] Connexions.“Serial Port Communication”. Available at http://cnx.org/content/m12293/latest/, last accessed on December 12, 2007

[9] Standford. “InfraRed Sensors”. Available at http://ccrma.stanford.edu/~gary/controllers/ir.html, last accessed on December 10, 2007

[10] Parallax Inc. “Hitachi HM55B Compass Module”. Available at http://www.parallax.com/detail.asp?product_id=29123, last accessed on December 3, 2007

[11] Parallax Inc. “Parallax Hitachi HM55B Compass Documentation.” Available at http://www.parallax.com/dl/docs/prod/compshop/HM55BModDocs.pdf, last accessed on December 3, 2007

[12] MicroMo Electronics of Clearwater, FL. “Encoder Types Tutorial” Available at http://www.faulhaber-group.com/n378900/n.html, last accessed on December 4, 2007

[13] Wikipedia. “Rotary encoder- Absolute rotary encoder”. Available at http://en-.wikipedia.org/wiki/Rotary_encoder, last accessed on December 3, 2007

[14] Copidate Technical Publicity “Introduction to Absolute Encoders and SSI”. Available at http://www.sensorland.com/HowPage021.html, last accessed on December 3, 2007

[15] SICK Stegmann “Incremental Encoders- Basic Operation of Optical Rotary Incremental Encoders”. Available at http://www.sick.es/sus/products/products/incremental/en.html, last accessed on December 3, 2007

[16] SICK Stegmann “Incremental Encoders- intro%20to%20inc%20encoders-coretech.pdf”. Available at http://www.sick.com/sus/products/products/incremental/en.toolboxpar.0002.file.tmp/intro%20to%20inc%20encoders-coretech.pdf , last accessed on December 3, 2007

[17] Wikipedia. “Rotary encoder- Incremental rotary encoder”. Available at

http://en.wikipedia.org/wiki/Quadrature_encoder, last accessed on December 3, 2007

[18] National Instruments “Magnetic Encoder Fundamentals”. Available at http://zone.ni.com/devzone/cda/tut/p/id/4500, last accessed on December 3, 2007

[19] Agilent Technologies “Agilent ADNS-2610 Optical Mouse Sensor Data Sheet-5988-9774EN.pdf”. Available at

http://cp.literature.agilent.com/litweb/pdf/5988-9774EN.pdf, last accessed on December 3, 2007

[20] Stringer, Kathi. “Robotics, Circuits, Mechanics, Servo Motors, Motion Controls and etc”. Available at http://www.toddlertime.com/robotics/servo.htm, last accessed on December 16 2007

[21] Seattle Robotics Society. “What’s a Servo?”. Available at http://www.seattlerobotics.org/guide/servos.html, last accessed on December 19, 2007

[22] Parallax Inc. “What’s a Microcontroller? Student Guide VERSION 2.2”. Available at

http://www.parallax.com/dl/docs/books/edu/wamv2_2.pdf , last accessed on December 19, 2007

[23] Parallax Inc. “Parallax Continuous Rotation Servos”. Available at http://

www.parallax.com/detail.asp?product_id=900-00008, last accessed on December 19, 2007